Haaga-Helia
ammattikorkeakoulu Oy

# Rakennusautomaation ohjelmointi pöytätietokoneella

Nikolai Muittari

Tietojenkäsittelyn koulutusohjelma

Haaga-Helia
ammattikorkeakoulu Oy

**Tiivistelmä**

29.11.2022

**Tekijä(t)**
Nikolai Muittari

**Koulutusohjelma**
Tietojenkäsittelyn koulutusohjelma

**Opinnäytetyön otsikko**

Rakennusautomaation ohjelmointi pöytätietokoneella

**Sivu- ja liitesivumäärä**
61 + 0

**Opinnäytetyön otsikko englanniksi**
Desktop application development for building automation systems

Opinnäytetyön tarkoituksena oli suunnitella ja kehittää Windows-pohjainen tietokonesovellus, joka tuottaa erityiset ohjeet ja ohjelmatiedostot Modbus-protokollan mukaista viestintää varten Fidelix-aseman keskusyksikön ja muiden valmistajien laitteiden välillä. Tässä opinnäytetyössä kehitetty sovellus on osa ohjelmistopakettia, joka on tarkoitettu rakennusautomaatiojärjestelmien projektien luomiseen.

Opinnäytetyössä keskitytään Windows-sovellusten kehittämiseen ja arkkitehtuuriin, yksityiseen Modbus-protokollan integrointiin. Sovelluksen tuloksia ohjelmatiedostoina ja datana on tarkoitus käyttää yhdessä toisen Fidelix-automaatio-ohjelmiston kanssa.

Opinnäytetyössä käytetään Java-ohjelmointikieltä, JavaFX-graafista käyttöliittymäkehystä ja Eclipse-kehitysympäristöä. Kehitystyön aikana tutustutaan tarkemmin vastaaviin työkaluihin ja tekniikoihin.

The purpose of this thesis was to design and develop a Windows operating system computer application which creates special instructions and program files for Modbus protocol communication between Fidelix substation CPU and third-party devices. The application developed in this thesis will be a part of software package dedicated to creating a project for building automation systems.

The thesis is focused on Windows application development and architecture, detailed Modbus protocol integration. Application's results as program files and data shall be used with another Fidelix automation software.

Java programming language, JavaFX graphical user interface library and Eclipse development environment are interned to be used in this thesis. During development, corresponding tools and techniques will be studied in detail.

**Asiasanat**
Sovelluskehitys, JavaFX, automaatiojärjestelmä, taloautomaatio (BMS), rajapinta, tiedonsiirto, Modbus RTU / TCP-IP

**Sisällys**

# 1. Johdanto

Nykyiset kotiautomaatiojärjestelmät ovat monimutkaisia laitteistojen ja ohjelmistojen yhdistelmiä, joissa monet kerrokset ovat vuorovaikutuksessa keskenään. Niiden virheetön toiminta on kriittinen vaatimus jokaisessa rakennuksessa.
Lähes jokaisessa järjestelmässä on älykkäitä laitteita, kuten ilmanvaihtimia, ilmastointilaitteita jne., jotka kommunikoivat keskenään käyttämällä vakiomuotoisia viestintäprotokollia ja -käytäntöjä. Niiden yhteenliittyminen ja koordinointi on tärkeää kustannusten ja työajan säästämiseksi ja vaikuttaa lopulta koko järjestelmän tehokkuuteen.

Tutkimuksessa pyritään kokoamaan yhteen eri älylaitteiden valmistajien erikoisuudet, niiden tiedonvaihtomenetelmät sekä löytämään yhteisiä tapoja ja lähestymistapoja. Tämän tiedon pohjalta kehitetään sovellus, jolla liitännät ja tiedonsiirrot voidaan yhdistää ja koordinoida.

Lopputuloksena on sovellus, joka auttaa taloautomaatiohankkeen projektipäällikköä (toteuttajaa) tekemään älylaitteen ja automaatiojärjestelmän väliset yhteydet nopeammin ja helpommin. Saadaan selkeä toiminnallinen kaavio, joka auttaa ymmärtämään tiedonsiirtoja ja kehittämään protokollia. Älylaitteiden kehittäjät voivat hyötyä sovelluksesta optimoimalla omia ohjelmiaan.

**Tavoitteet:**
Saada vastauksia kysymyksiin / selvittää asioita:
Onko mahdollista rakentaa monikäyttöinen ja helppokäyttöinen sovellus, joka integroi eri Modbus-laiteliitännät, luo valmiita IEC-ohjelmia rakennusautomaatioprojektin hyödyksi.
Millaisia ominaisuuksia ja toimintoja sovelluksessa olisi?
Miten MVC-ohjelmointitapaa ja -tekniikoita käytetään helpon ja joustavan sovelluksen rakentamiseen.
Miten UI-suunnittelu (käyttöliittymä) rakennetaan?

## 2. Käsitteet

| | |
|---|---|
| BMS | Building Management System<br>Rakennusautomaatiojärjestelmä, RAU-järjestelmä |
| PLC | Programmable Logic Controller<br>Vapaasti ohjelmoittava logiikka |
| Ala-asema, Alakeskus<br>Valvonta-alakeskus<br>(VAK) | Ohjaava yksikkö rakennusautomaatiojärjestelmässä<br>Koostu yleensä vapaasti ohjelmoitavasta logiikasta, I/O moduuleista ja apulaitteista. |
| I/O-moduuli | (Input/Output) Tulo-lähtö pisteistä koostuva piirikortti |
| KNX | Avoin kommunikointistandardi rakennusautomaatioon |
| LonWorks | LonWorks eli Local Operating Network on avoin standardi (ISO/IEC 14908) verkkoalustoille, jotka on luotu erityisesti ohjaussovellusten tarpeisiin. |
| BACnet<br>IP, MS/TP | Rakennusautomaatio- ja ohjausverkkojen (BAC) kommunikointiprotokolla. |
| Modbus RTU / Modbus TCP-IP | Kommunikointi protokolla käytetty automaatiojärjestelmissä |
| IEC 61131-3 (IEC) | Ohjelmoitavia logiikkaa (PLC) koskeva avoin kansainvälinen standardi IEC 61131. Käsittelee PLC:n ohjausohjelman perusohjelmistoarkkitehtuuria ja ohjelmointikieliä. |
| HMI<br>päätelaite | Human-machine interface<br>Käyttäjän rajapinta laitteella tai järjestelmässä. |
| PID-säädin | Proportionaali-integraali-derivoiva-säädin |
| DDC | Direct digital control<br>Suora digitaalinen ohjaus |
| BIM | Building information modeling<br>Rakentamisen tietomallintaminen |
| SCADA. | Supervisory control and data acquisition<br>Valvomo-ohjelmisto: graafinen käyttöliittymä automaatiojärjestelmiin |

# 3. Automaation perusteet

Tässä osassa pyrimme määrittelemään automaation järjestelmien rakentamisen alana ja analysoimaan automaatiojärjestelmän historiaa, komponentteja ja luomisprosessia.

> Automaatio, jolla yleisesti tarkoitetaan itsestään toimivaa, on viime vuosikymmeninä saanut monenlaisia merkityksiä. Tekniikan yhteydessä puhutaan muun muassa instrumenttitekniikasta, mittaus- ja säätötekniikasta, servotekniikasta sekä logiikkaohjauksista. Yleismuodossa automaattisella tarkoitetaan tietosanakirjan mukaan itsestään, ilman ohjausta tapahtuvaa tai toimivaa. (Keinänen, ym., 2007, 7.)

Sana ”automaatio” tulee kreikan kielen sanasta "automatos". joka tarkoittaa itsetoimivaa. Automaatio on kaikenlaista työtä ja toimintaa helpottava prosessi, joka on levinnyt monille ihmiselämän aloille. Lähes kaikissa valmistukseen liittyvissä toiminnoissa ja monissa palveluihin liittyvissä toiminnoissa automaatiota käytetään rutiinitehtävien siirtämiseen ihmisiltä teknisille tai ohjelmistojärjestelmille.

> Automaatio voidaan yleisesti ottaen määritellä prosessiksi, jossa noudatetaan ennalta määriteltyä toimintojen sarjaa vähäisellä tai olemattomalla ihmistyövoimalla käyttäen erikoislaitteita ja -laitteita, jotka suorittavat ja ohjaavat valmistusprosesseja. (Gupta, ym., 2017, 2)

## 3.1. Automaation historia

### 3.1.1. Muinaiset ajat

Ajatus sellaisten koneiden ja mekanismien luomisesta, jotka toimisivat ilman ihmisen väliintuloa, juontaa juurensa antiikin ajoilta Kreikassa ja Egyptissä. Jo 3. vuosisadalla eaa. kreikkalainen filosofi Aristoteles haaveili automaattisista tai itsetoimivista laitteista. Ensimmäiset ihmisen tekemät automaattiset toimintalaitteet luotiin ja niitä käytettiin uskonnollisiin tai viihdetarkoituksiin. Ensimmäinen maininta automaatiolaitteesta on peräisin Egyptistä ensimmäiseltä vuosisadalta jKr. Matemaatikko ja insinööri Heron Aleksandrialainen loi ensimmäisen myyntiautomaatin. Hän rakensi myös koneet temppelin ovien avaamista varten ja vesiurut.

> Yksi tärkeä virstanpylväs automaattisten järjestelmien historiassa oli säätölaitteen keksiminen, kuten Ktesiboksin-vesikellossa (kreikkalainen keksijä 283–247 eKr) (Keinänen, ym., 7.).

### 3.1.2. Mekaaninen automaatio

Vesipyörän keksiminen, myllyjen tuulimyllymoottori ja mekaaninen kello, jossa on säätökoneisto, olivat automaatiojärjestelmien peruspilareita ennen höyrykoneen keksimistä. Kaikki keskiajan automaatiolaitteet perustuivat tavalla tai toisella näihin keksintöihin, joissa oli mekaaninen palaute. Niiden merkitys sivilisaatiolle on valtava ja tärkeä.

Höyrykoneen keksimisen jälkeen James Wattin keskipakoisessa höyrykoneen säätimessä (1784) käytettiin negatiivista takaisinkytkentää koneen nopeuden säätämiseksi. Tämä oli lähtökohta monien sellaisten teollisuuslaitteiden rakentamiselle, joissa oli höyrykoneet, jotka pystyivät toimimaan automaattisesti tai minimaalisen vähän ihmisen puuttuessa asiaan. Esimerkiksi höyrykäyttöisten kangaspuiden käyttö johti tuotannon valtavaan kasvuun.

Prosessiautomaatiossa on tärkeää huomata, että hollantilainen insinööri Cornelius Drebbel keksi maailman ensimmäisen säätimen, jolla kananmunahautomoita voitiin lämmittää tasaisessa lämpötilassa. Hänen kehittämäänsä takaisinkytkentäohjattua laitetta käytettiin 1970-luvulle asti. (Bosch, 2020)

Säätimet näyttivät tasoittavan tietä ohjausperiaatteiden ja säätimien keksintöjen tulvalle, joka jatkui 1900-luvun puoliväliin asti. Ei ole sattumaa, että höyrykone oli ensimmäinen ohjaustekniikan ja -teorian sovelluskohde, sillä se ei pystynyt toimimaan itsenäisesti eikä sillä ollut "itsesäätöä". Sen epäsuotuisat dynaamiset ominaisuudet johtivat usein siihen, että siihen kytketty säädin ei toiminut suunnittelijan odottamalla tavalla, "heilutti" konetta tai ei pystynyt ohjaamaan sitä lainkaan. Tämä kaikki johti luonnollisesti teoreettiseen tutkimukseen.

### 3.1.3. Sähköinen aikakausi

Teollisuusautomaation kehitys kiihtyi nopeasti 1920-luvulla, kun tehtaat alkoivat käyttää relelogiikkaa eli sähköistettiin. Sähkökeskusvoimaloiden yleistyminen yhdessä uusien korkeapaineisten kattiloiden, sähköasemien ja höyryturbiinien käytön kanssa johti siihen, että mittalaitteiden ja ohjauslaitteiden kysyntä kasvoi.

Tuotantolaitokset alkoivat siirtyä käyttämään sähkömoottoreita, ja yhä harvemmat laitokset käyttivät edelleen höyrykoneita. Tämän siirtymävaiheen aikana teollisuusyritykset lisäsivät tuotantoaan noin kolmanneksella. Tämä johtui siitä, että sähkömoottoreiden hyötysuhde oli paljon korkeampi kuin höyrykoneiden, ja ne vaativat vähemmän huoltoa.

Tehokkaan toiminnan varmistamiseksi valvomoista oli lähetettävä erilaisia signaaleja laitoksen työntekijöille, jotta he voisivat tehdä muutoksia manuaalisesti, kuten avata tai sulkea venttiilejä ja kytkeä kytkimiä päälle tai pois päältä. Tämä on prosessinohjauksen tyyppi, joka tunnetaan nimellä "on-off". 1930-luvulla teollisuuteen ilmestyi säätimiä, jotka tekivät laskennallisia muutoksia vastauksena poikkeamiin asetusarvosta.

Prosessiautomaation ja erityisesti lämpötilan säädön kannalta tärkein innovaatio oli Nikolai Minorskin keksitty vuonna 1925 PID-säätimen (Proportionaali-integraali-derivoiva-säädin), joka on alun perin laivojen ohjauksessa käytetty säätöpiirimekanismi. Samaa periaatetta sovellettiin kuitenkin pian ensimmäisen analogisen laitteen luomiseen lämmitysjärjestelmien lämpötilan säätöön. Näiden ohjaimien avulla sähköisillä ja myöhemmin elektronisilla laitteilla voitiin ohjata prosesseja veden lämmityksestä rakettien suuntaamiseen.

### 3.1.4. Elektroniikka automaatiossa

Mikroprosessorin keksiminen 1970-luvulla johti tietokonelaitteistojen hintojen merkittävään laskuun ja mahdollisti digitaalisten ohjausjärjestelmien nopean kasvun kaikilla teollisuuden aloilla. Tähän päivään asti tietotekniikan jatkuva kehittyminen on edelleen teollisuusautomaation kehityksen moottori. Digitaalisten tietokoneiden ansiosta teollisuusyrityksillä on nyt ohjaimet, jotka pystyvät suorittamaan monimutkaisempia tehtäviä nopeammin ja tehokkaammin. Digitaali-analogi- ja analogia-digitaalimuunninten kehittyminen on mahdollistanut vanhentuneiden mekaanisten, pneumaattisen ja sähköisten ohjausmenetelmien täydellisen poistamisen ja korvaamisen ohjelmistomenetelmillä. Tietoväylien ja tietoliikenneverkkojen laajamittainen käyttöönotto on mahdollistanut järjestelmien yhdistämisen ja valtavien laitosten säätelyn yhdestä ainoasta valvontakeskuksesta käsin, jolloin tuotanto- ja ympäristöparametrit voidaan ottaa huomioon tuotannon lisäämiseksi ja resurssien säästämiseksi.

Suoran digitaalisen ohjauksen (DDC, Direct digital control) käyttöönotto on mullistanut markkinat rakennusautomaation alalla. Sen käyttö on laajentunut nopeasti, mikä on lisännyt huomattavasti rakennusautomaation ohjausjärjestelmien markkinoita. Vanhat pneumaattiset ja analogiset laitteet ja ohjausjärjestelmät on korvattu.

Seuraava virstanpylväs oli rakennusten tietomallintamisen (BIM, Building information modeling) kehittäminen. Näiden ohjelmistopakettien avulla voitiin kerätä ja analysoida kaikki rakennuksen toimintaa koskevat olennaiset tiedot. BIM-järjestelmät olivat ensimmäiset ohjelmistotyökalut, joilla pystyttiin mallintamaan, integroimaan ja keräämään kaikki olennaiset rakennustiedot. Ne mahdollistivat jatkuvan tietomallinnuksen koko rakennuksen elinkaaren ajan - mikä on edellytys nykyaikaisille rakennusautomaatiojärjestelmille.

### 3.1.5. Verkot ja tekoäly

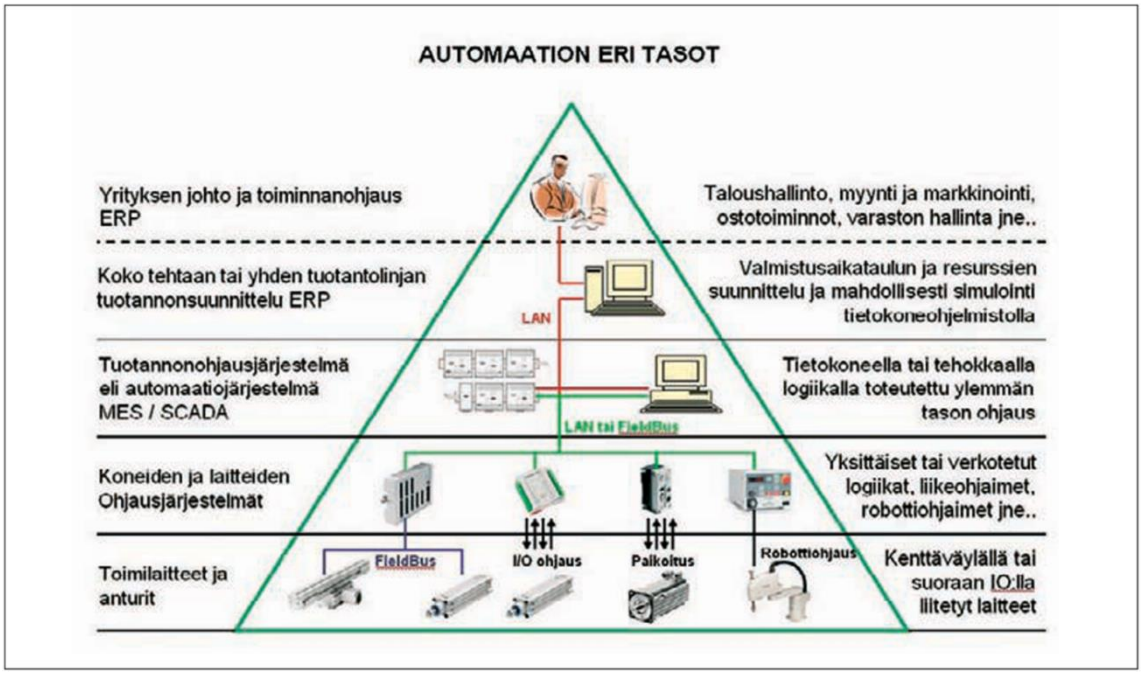
Paikallisten ja maailmanlaajuisten tietoverkkojen nopea kehitys on ollut loogista jatkoa tietoväylien käytölle. Eri automaatiolaitteiden valmistajat kehittävät sekä omia että avoimia standardeja automaatiojärjestelmiä varten. Yhä tehokkaammat prosessorit ja halvemmat tietokonelaitteiden hinnat mahdollistavat sen, että automaatiojärjestelmiä voidaan käyttää lähes missä tahansa, missä on tarvetta. Tiedonsiirtonopeudet ja mahdollisuus käyttää internetiä langattomien verkkojen kautta ovat synnyttäneet käsitteen "esineiden internet", jossa jokaisella laitteella on oma ohjelmoitava ohjain ja joka on vuorovaikutuksessa käyttäjän lisäksi myös muiden järjestelmään integroitujen laitteiden kanssa.

Ohjelmistojen kehittäminen mahdollistaa aiemmin vain fiktiossa esiintyneiden "tekoälymenetelmien" soveltamisen, jotka analysoivat järjestelmässä olevia tietoja ja pystyvät päättämään yhdestä tai toisesta toimenpiteestä optimointi- tai onnettomuuksien ehkäisytehtävien ratkaisemiseksi.

## 3.2. Laitteisto ja ohjelmisto

Kaikissa nykyaikaisissa automaatio- tai ohjausjärjestelmissä on useita elektronisia, sähköisiä, sähkömekaanisia tai pneumaattisia laitteita, jotka ovat vuorovaikutuksessa keskenään ja ympäristön kanssa. Kuten missä tahansa monimutkaisessa järjestelmässä,

signaalit ja vaikutukset välittyvät yksinkertaisemmilta tasoilta monimutkaisemmille tasoille. Näin ollen automaatiojärjestelmissä on useita tasoja, joilla tietoa ja vaikutteita välitetään ja muunnetaan.

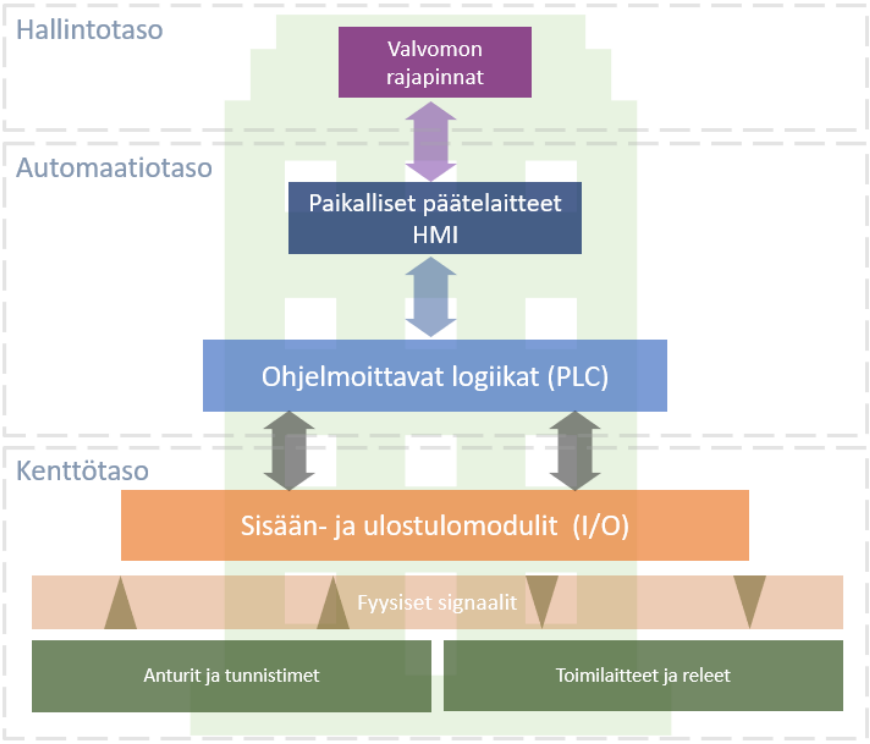



Kuva 1. Tuotantoautomaation rakenne (Keinänen, ym., 2007, 209)

Rakennusautomaatiojärjestelmä eroaa teollisuusautomaatiosta siinä, että se on pitkälti sidoksissa ihmisten elämän ja viihtyvyyden ylläpitämiseen.

> Rakennusautomaatiolla tarkoitetaan rakennusten lämmitys-, ilmanvaihto-, valaistus-, hälytys-, ja valvontajärjestelmien eli talotekniikan automaattista ohjausta ja valvontaa. Rakennusautomaatiojärjestelmällä on usein toiminnallisia yhteyksiä kiinteistönhallinta- ja turvajärjestelmiin, kuten kulunvalvonta, palonilmaisu ja savunpoistojärjestelmät, jotka ovat usein integroitu osaksi rakennusautomaatiojärjestelmää. (Härkönen ym., 2012)

Tyypillinen rakennusautomaatiojärjestelmä koostuu antureista, sähkömekaanisista toimilaitteista, fyysisen signaalinmuuntoyksiköistä, ohjelmoitavista logiikkoista ja käyttöliittymästä (HMI, Human-machine interface). Nämä laitteet muodostavat eri kerroksia, jotka suorittavat erilaisia tehtäviä ja kommunikoivat keskenään tietoväylien välityksellä.




Kuva2. Automaatiojärjestelmän tasot

### 3.2.1. Toimilaitteet ja anturit

Toimilaitteita ja antureita (mittalaitteita, tunnistimia) käytetään automaatiojärjestelmän kenttätasolla. Myös paikalliset säätimet ja parametrien asetuslaitteet kuuluvat usein tälle tasolle.

> Mittalaitteita ovat muun muassa lämpötila-, kosteus- ja paineanturit. Toimilaitteita ovat muun muassa prosesseja säätävät venttiilimoottorit, peltimoottorit ja taajuusmuuttajat. Kenttätasolla voi olla itsenäisiä säätimiä, kuten huonesäätimet, jotka ohjaavat paikallisesti tilan olosuhteita. (Härkönen ym., 2012, s. 95)

Anturit toimivat automaatiojärjestelmän aistieliminä ja antavat ajantasaista tietoa järjestelmien ja prosessien nykytilasta. Erilaiset toimilaitevaihtoehdot toimivat manipulaattoreina, joiden avulla automaatiojärjestelmä voi vaikuttaa prosesseihin tietyn tuloksen saavuttamiseksi tai onnettomuuksien estämiseksi. Esimerkiksi lämpötila-anturilta saatu arvo lähetetään PLC logiikkaan, ja sen ohjelma määrittää kulman, jolla lämminvesiventtiilin moottoriventtiiliä käännettävä huoneen lämpötilan muuttamiseksi.

Toinen esimerkki on paikallisen säätimen käyttö: käyttäjän asettama haluttu lämpötila-arvo lähetetään paikalliseen säätimeen, joka on periaatteessa pieni ohjelmoittava logiikka

(PLC), joka päättää itse, avataanko lämminvesiventtiili halutun lämpötilan saavuttamiseksi. Paikallinen logiikka kommunikoi ylemmän tason logiikan kanssa ja voi toimia sen suorassa ohjauksessa.

Kenttätasoon kuuluvat myös analogia-digitaali- ja digitaalianalogia muunninmoduulit (I/O-moduulit) sekä protokolla- ja dataväylämuuntimet. Niiden tehtävänä on kääntää ympäröivän maailman tiedot muotoon, jota ohjelmoitavat logiikat ymmärtävät, ja siirtää ne nopeasti ja vääristymättömästi.

Esimerkiksi lämpötilatiedot lähetetään seuraavasti:

- mittaamalla anturin (termistorin) sähköinen resistanssi ohmeina ja muuntamalla arvo numeeriseen muotoon (tavuiksi) käyttämällä asianmukaista taulukkoa (resistanssi - lämpötila).
- Numeroarvo siirretään muistirekisteriin, josta ohjelmoitava ohjain vastaanottaa sen tietoväylän kautta.
- Säädin muuntaa numeerisen arvon tavuista celsiusasteiksi asennetun anturityypin mukaan.

Tietoväylämuuntimet toimivat protokollan (ja/tai fyysisen portin) muuntamisjärjestelmän mukaisesti. Tällainen laite purkaa vastaanotetut tiedot, kirjoittaa ne väliaikaiseen muistiin, muuntaa ne sitten eri muotoon standardien mukaisilla taulukoilla ja kirjoittaa ne rekistereihin, joista ohjelmoitava ohjain saa ne ymmärrettävässä muodossa.

### 3.2.2. Logiikat (PLC)

Tyypillisessä automaatiojärjestelmässä ohjaus tapahtuu ohjausyksiköillä, eli ohjelmoitavilla logiikkaohjaimilla, jotka usein ovat teollisuustietokoneita (tai ohjelmoitavia mikroprosessoreita), joissa on useita ulkoisia liitäntöjä, käyttöjärjestelmä ja sisäinen muisti. Ohjelmiston on mahdollistettava jatkuva yhteydenpito kenttälaitteiden kanssa järjestelmää koskevien tietojen saamiseksi. Nämä tiedot siirretään ohjaimen sisäiseen ohjelmaan.

Ohjelma sisältää yleensä seuraavat lohkot:

- vastaanottaa tietoja kenttätason laitteista
- tietojenkäsittely ja päätöksenteko
- komentojen ja signaalien lähettäminen kenttälaitteille ja järjestelmän muille osille, kuten muille ohjelmoitaville logiikoille, korkeamman tason ohjaus- ja valvontalaitteille ja käyttöliittymille.

Nykyaikaisista automaatiojärjestelmistä on tullut hyvin monimutkaisia, joten ohjausyksikön on suoritettava suuri määrä toimintoja ja niissä on oltava erilaisia liitäntöjä, jotta ne ovat yhteensopivia monenlaisten laitteiden kanssa. Ohjausyksikössä on myös oltava riittävän suuri muistikapasiteetti ohjelmien, mittaustulosten ja historiatietojen tallentamista varten. Nykyaikaiset ohjausyksiköt voivat vaihtaa tietoja ja ohjata tuhansia kenttälaitteita, ne ovat yhteensopivia kymmenien eri liitäntöjen kanssa ja tallentavat useiden vuosien järjestelmän tilatiedot pitkäaikaismuistiin.

### 3.2.3. Valvomot

Automaatiojärjestelmän ylimmällä tasolla ovat ohjausjärjestelmät tai valvomot. Nämä yksiköt kommunikoivat alemman tason ohjelmoitavien logiikoiden kanssa ja keräävät kaikki tiedot niiltä. Tiedonkeruu tapahtuu yleensä nopeiden tietoverkkojen kautta. Valvomot on varustettu suuritehoisilla tietokoneilla, joissa on paljon muistia ja erikoisohjelmistoja. Käyttäjäystävällinen graafinen käyttöliittymä on olennaisen tärkeä valvontajärjestelmän tehokkaan toiminnan kannalta. Valvomossa operaattori voi saada ajantasaista tietoa kymmeniltä tai jopa sadoilta ohjelmoitavilta ohjaimilta, reagoida hälytyksiin ja toimia manuaalisesti automaatiokomponenteilla tarpeen mukaan. Mittaustietojen ja muiden parametrien keräämisen avulla voidaan analysoida järjestelmän kaikkien osien toimintaa ja tehdä optimointi- tai korjauspäätöksiä.

Valvomotietokoneiden ohjelmat on suunniteltu yhteensopiviksi monien protokollien ja tietoverkkojen kanssa, jotta kommunikaatio eri valmistajien erityyppisten ohjelmoitavien logiikoiden kanssa voi tapahtua saumattomasti.

## 3.3. Automaatiojärjestelmien toteutus

Tässä osassa tarkastelemme rakennusautomaatioprojektin toteuttamisen päävaiheita. Tarkastelemme toteutuksen vain teknistä puolta, vaikka hanke kokonaisena sisältää muun muassa rahoituksen, rakennuttajan hyväksymisen, urakoitsijoiden valinnan ja asiakirjojen laatimisen vaiheet.

Eri projektien sisältö, tehtävät ja toteutustavat voivat vaihdella eri tapauksessa, mutta perusmenetelmät ja lähestymistavat ovat samat:

- suunnittelu;
- laitteiden ja ohjelmistojen valinta, ohjelmointi;
- asennus ja käyttöönotto;
- testaus ja ylläpito;

### 3.3.1. Määrittely ja suunnittelu

Rakennusautomaatiojärjestelmä on nykyään kiinteä osa lähes jokaista rakennusta. Rakennusautomaatiohankkeen tavoitteet ja vaatimukset alkavat jo arkkitehtisuunnitteluvaiheessa, kun uuden rakennuksen (tai peruskorjatun rakennuksen) perusparametrit määritellään.

Näihin vaatimuksiin kuuluvat pääasiassa:

- lämmitys ja ilmanvaihto;
- valaistus- ja kulkujärjestelmät;
- teknisten järjestelmien hallinta;
- resurssikirjanpitojärjestelmä;

Automaatiojärjestelmää varten laaditaan kutakin vaatimusta varten luettelo toiminnoista ja vaikutuksista, jotka pitävät rakennusjärjestelmät tietyssä tilassa. Näiden toimintojen ja rakennukseen valittujen laitteiden (ilmanvaihtokoneet, mittarit, lisälaitteet) perusteella laaditaan automaatiojärjestelmäsuunnitelma.

Rakennusautomaatiojärjestelmä suunnitellaan usein valmiiksi insinööritoimistoissa, jotka saavat toimeksiannon rakennuttajalta yhdessä rakennusasiakirjojen kanssa.

Suunnittelussa otetaan huomioon lukuisat rakennuksen teknisiin järjestelmiin vaikuttavat tekijät sekä lämmityksen, ilmanvaihdon ja muiden rakennusosien määritellyt ominaisuudet.

Suunnitteluprosessissa otetaan huomioon putkistojen, kanavien, sähkökaappien ja muiden rakennuksen apujärjestelmien sijainti rakennusasiakirjojen perusteella. Nämä tekijät määrittävät automaatiojärjestelmän johdotusrakenteen, anturien ja toimilaitteiden valinnan sekä ohjelmoitavien logiikoiden (ala-asemien) määrän ja sijainnin.

Esimerkiksi sähkömekaanisen toimilaitteen valinta lämmityksen ohjaukseen riippuu putken halkaisijasta ja venttiilin koosta. Monikerroksisessa rakennuksessa on perusteltua asentaa useita I/O-moduuleja eri kerroksiin ja liittää ne ohjelmoitavaan logiikan (ala-asemaan) tietoväylän kautta kaapeloinnin pituuden lyhentämiseksi.

Suunnittelijoiden on otettava huomioon eri automaatiolaitteiden ominaisuudet, rajoitukset ja yhteensopivuus. On varmistava esimerkiksi, että sisäänrakennetut ilmastointilaitteet tukevat samoja väylä- ja tiedonsiirtoprotokollia kuin suunniteltu ohjelmoitava logiikka.

Suunnittelija laatii myös yksityiskohtaisen ja kattavan kuvauksen järjestelmän toiminnasta (toimintaselostus) automaation näkökulmasta järjestelmän oikeaa ohjelmointia varten. Suunnittelijoiden työn tuloksena pitäisi olla piirustukset ja kuvaus toiminnasta, joka noudattaa standartteja ja jonka avulla projekti-insinööri voi toteuttaa kaikki vaatimukset.

### 3.3.2. Toteutus ja testaus

Kun suunnitteluasiakirjat on saatu, projekti-insinööri (tai insinööriryhmä) aloittaa toteuttamisen. Laitteen valinta riippuu usein kokemuksesta tietystä valmistajasta ja laitteiden saatavuudesta. Valmisteluvaiheessa voidaan tehdä muutoksia tai mukautuksia, joista sovitaan sekä suunnittelutoimiston että asiakkaan kanssa, koska ne voivat lisätä projektin kustannuksia.

Projekti-insinööri tekee yhteistyötä saman rakennushankkeen muiden tiimien, kuten sähkö-, ilmanvaihto- ja vesijärjestelmän-teknikoiden, kanssa asentaakseen ja liittääkseen anturit, toimilaitteet ja muut automaatiojärjestelmän osat oikea-aikaisesti.

Jo ennen laitteiden asentamista insinööri alkaa valmistella kaikkia tarvittavia ohjelmia ja liitäntöjä, jotta ne täyttävät hankeasiakirjojen vaatimukset. Ohjaimen ohjelmointi ja alustava suorituskyvyn testaus voidaan suorittaa urakoitsijan toimistoissa ja testauslaboratorioissa. Tässä vaiheessa voidaan havaita mahdolliset virheet ja puutteet, jotka olisivat halvempaa korjata ennen laitteiden asennusta paikan päällä.

Valitut laitteet viedään työmaalle ja asennetaan suunnitelluille paikoille. Kaapelijärjestelmä testataan ja kenttätasolaitteiden moitteeton toiminta alustavasti tarkistetaan kuin mahdollista. Logiikoiden (sähköasemien) ohjelman luominen tehdään yleensä ennen koko järjestelmän käyttöönottoa, vaikka tarvittavat korjaukset tai muutokset voidaan tehdä "lennossa", jos ohjainten ohjelmisto sen sallii.

Käyttöönotto suoritetaan, kun ohjelmoitava logiikka (sähköasema) pystyy suorittamaan toimintaselostuksessa vaaditut toiminnot ja kaikki anturit, toimilaitteet ja muut apulaitteet on kytketty. Koekäyttöprosessi on erittäin tärkeä kaikissa projekteissa, koska sen avulla voidaan havaita mahdolliset virheet ja ristiriidat ennen kuin järjestelmä voidaan ottaa normaaliin käyttöön. Koekäytöt tehdään yleensä yhteistyössä muiden rakennusjärjestelmien asiantuntijoiden kanssa, jotta voidaan ehkäistä hätätilanteita asianomaisissa rakennusjärjestelmissä.

Koekäytön ja mahdollisten virheiden korjaamisen jälkeen projekti siirtyy testausvaiheeseen. Asiakkaan edustaja ja/tai erityisasiantuntija suorittaa toimintakokeen, jonka aikana tarkistetaan kaikki käytettävissä olevat automaatiojärjestelmän toiminnot ja sen reagointi ulkoisiin vaikutuksiin, sekä normaali- että hätätilanteissa.

Toimintakokeen aikana tarkistetaan lämmitys-, ja ilmanvaihtojärjestelmien perustoimintatilat sekä niiden vakaus ja toimintaselostuksen mukaisuus.

Pysyvä käyttöönotto on mahdollista vasta, kun kaikki virheet ja ristiriidat on korjattu ja ohjelmoitava ohjain (sähköasema) toimii perustoiminnoissaan. Toimintakoe on projektin tärkein vaihe; kun se on hyväksytty, insinööri siirtyy viimeisiin vaiheisiin käynnistykseen, luovutukseen ja ylläpitoon.

### 3.3.3. Käyttöönotto ja ylläpito

Automaatiojärjestelmä otetaan usein käyttöön jo ennen rakennuksen avaamista, jotta kaikkien järjestelmien toiminta ja vakaus ehditään tarkistaa. Koekäytön aikana testattua järjestelmää tarvitsee harvoin korjata tai muuttaa. Tässä vaiheessa projekti-insinööri laatii dokumentaation ja siirtää järjestelmän asiakkaalle. Käyttöönoton aikana voi kuitenkin tulla tilanteita, joissa järjestelmään on tehtävä lisäyksiä tai muutoksia, jolloin näiden muutosten jälkeen ei testata koko järjestelmää, vaan ainoastaan niitä osia, joihin muutokset on tehty.

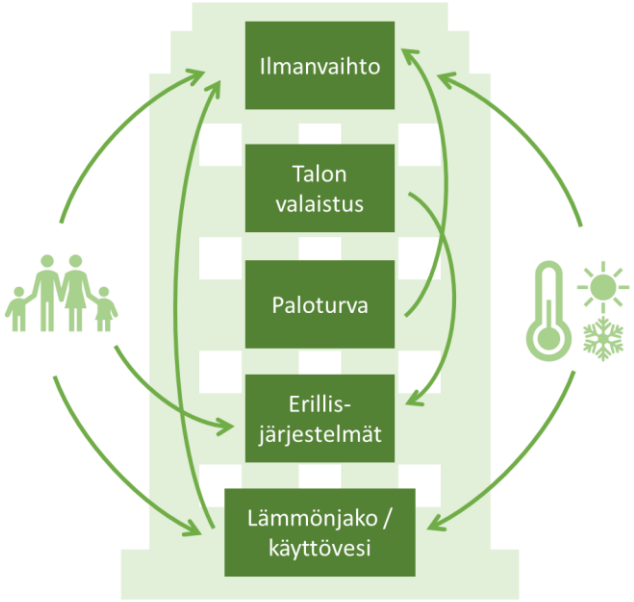
Käyttöönottovaiheessa automaatiojärjestelmä liitetään valvomoon tai muihin hallintatason järjestelmiin. Järjestelmän tiedot siirretään ylemmälle tasolle, ja niiden tallennus ja eheys testataan jatkuvan toiminnan varmistamiseksi.

Kun järjestelmä on luovutettu asiakkaalle, projekti-insinöörin on tarkkailtava järjestelmää jonkin aikaa, säädettävä sen parametreja tarvittaessa ja tehtävä korjauksia, jos jokin elementti tai laite meni rikki. Käytäntö osoittaa, että tämä todennäköisyys on suuri ensimmäisten viikkojen kuluttua normaalin toiminnan aikana. Projektin myöhempi tukeminen voi siirtyä muille asiantuntijoille tai jopa toiselle yritykselle. Yksi projekti-insinöörin tehtävistä on tässä tapauksessa säilyttää ja siirtää dokumentaatio ja ohjelman lähdekoodit hankkeen päivityksiä tai muutoksia varten.

# 4. Älykäs rakennus. Konseptin tekniset, taloudelliset, toiminnalliset ja ympäristönäkökohdat.

Kun suunnitellaan ja rakennetaan asuinrakennuksia, toimistorakennuksia, ostoskeskuksia tai sairaaloita, sijoittaja tai omistaja ja sopimussuunnittelu- ja rakennusliikkeet sekä rakennuksen johtoryhmä tekevät yhteistyötä luodakseen täydellisen suunnittelupaketin

rakentamista varten. Tähän sisältyy rakennuksen arkkitehtoninen suunnittelu, tilojen sijoittelu, rakennusvaiheet sekä teknisten järjestelmien kokoonpano ja hallinta. Kun otetaan huomioon, että teknisten järjestelmien kustannusten osuus rakennuksen kokonaiskustannuksista on 30–50 prosenttia, tämän kysymyksen periaatteellinen ja oikea-aikainen ratkaisu vaikuttaa paitsi rakennuskustannuksiin tulevaisuudessa myös rakennuksen järjestelmien nykyisiin huolto- ja korjauskustannuksiin, kuukausittaisten yleishyödyllisten palveluiden maksujen määrään ja rakennuksessa työskentelevien ja asuvien ihmisten mukavuustasoon. Näin ollen rakennusten järjestelmien asianmukainen valvonta ja seuranta on erittäin tärkeää käytännössä ja taloudellisesti.




Kuva 3. Älykäs talon peruskaavio

Seuraavassa esitellään joitakin "älykkään rakennuksen" käsitteen teknisiä, toiminnallisia, taloudellisia ja ympäristöön liittyviä näkökohtia ja kerrotaan rakennusautomaatiojärjestelmien perusperiaatteista, järjestelmien välisten yhteyksien arkkitehtuurista ja siitä, miten ne liittyvät toisiinsa.

## 4.1. Rakennusautomaatiojärjestelmä

Rakennusautomaatiojärjestelmä (RAU-järjestelmä, Building Management System BMS) on elektroninen laitteiston ja ohjelmistojen kokoelma, joka mittaa, havaitsee, säätää ja

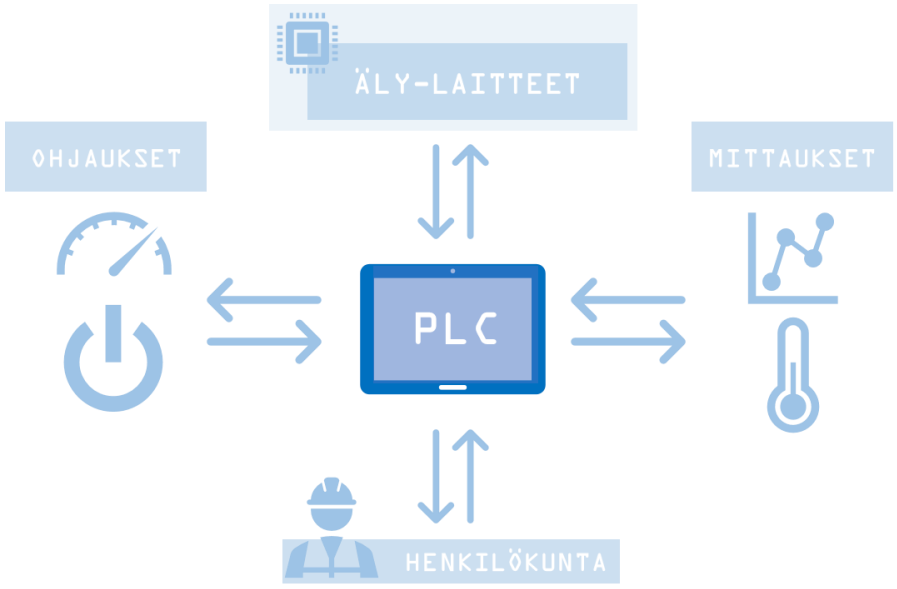
ohjaa (vaikuttaa) rakennuksessa kiertäviä tietoja (signaaleja). Nykyaikaisessa rakennuksessa on kymmeniä tai useampia elämää ylläpitäviä järjestelmiä, jotka eroavat toisistaan paitsi toiminnaltaan ja toiminnoiltaan myös toimintatavoiltaan: sähköiset, mekaaniset, elektroniset, hydrauliset ja niin edelleen. Valmistaja toimittaa kunkin järjestelmän yleensä laitekokonaisuutena, josta voidaan muodostaa kokonaisratkaisu, jossa on oma ohjaus- ja valvontajärjestelmä. Jotta voidaan varmistaa, että kaikki nämä erilaiset tekniset järjestelmät toimivat yhdessä, kommunikoivat keskenään ja että niitä ohjataan yhdestä valvomosta, älykkään rakennuksen tärkein lenkki on rakennusautomaatiojärjestelmä.

Suurissa projekteissa RAU-järjestelmä pystyy valvomaan laitteita keskitetysti ja hallitsemaan seuraavia teknisiä järjestelmiä ja kokonaisuuksia:

- Lämmitysjärjestelmä;
- Kuuman ja kylmän veden syöttöjärjestelmä;
- Viemäröinti- ja viemäröintijärjestelmät;
- Sähkönjakelujärjestelmä;
- Valaistusjärjestelmät (huone-, käytävä-, julkisivu- ja hätävalaistusjärjestelmät);
- Ilmanvaihtojärjestelmä;
- Ilmankäsittely-, puhdistus- ja kostutusjärjestelmä;
- Jäähdytysjärjestelmä
- Ilmastointi- ja ilmastointijärjestelmä;
- Ilmastointijärjestelmä; Ilmastointi- ja ilmastoinnin säätöjärjestelmä; Kaasun havaitsemis- ja valvontajärjestelmä;
- Resurssikirjanpito- ja valvontajärjestelmä (vesi, lämpö, sähkö);
- Palohälytys- ja turvajärjestelmä;
- Palontorjunta- ja sammutusjärjestelmä;
- Kulunvalvonta- ja hallintajärjestelmä;
- Pysäköintialueiden hallintajärjestelmä;
- Meteorologinen järjestelmä;

Nykyiset RAU-järjestelmät sisältävät yleensä yhden tai muutaman vapaasti ohjelmoitavan logiikkayksikön (PLC), jotka ohjaavat rakennuksen vaikutuksia.

RAU-järjestelmässä PLC sisältää kuitenkin keskusyksikön lisäksi usein erillisiä laitteita, joilla on omat PLC-yksikkönsä. Ne ohjaavat esimerkiksi ilmanvaihtokoneita (IV-yksiköt), lämpöpumppuja, sähkömittareita jne. RAU PLC-yksikön ja koneen PLC:n välinen kommunikointi tapahtuu tietoverkkojen kautta standardiprotokollaa käyttäen.




Kuva 4. RAU-järjestelmän rakenne

## 4.2. Älykäs rakennus -konseptin taloudelliset näkökohdat

Rakennuksen hallintajärjestelmän käyttö lisää rakennuksen suunnittelun kokonaiskustannuksia, ja se riippuu rakennuksen koosta ja teknisten järjestelmien toimintaa koskevista teknisistä vaatimuksista. Samalla RAU-järjestelmän ja resursseja säästävien laitteiden käyttö tarjoaa monia etuja. Surina:an (2007, s. 132) mukaan ne ovat seuraavat:
Rakennusautomaatio vähentää lisähyödykkeiden rakentamisesta ja sähkökaapeloinnista aiheutuvia kustannuksia erityisesti kaupungin keskeisillä alueilla, joilla kunnat rajoittavat rakennusten omistajien energiankulutusta. Rakennusautomaatio vähentää viallisten laitteiden korjaus- ja vaihtokustannuksia ja pidentää laitteiden käyttöikää seuraamalla jatkuvasti teknisten järjestelmien parametreja ja tekemällä oikea-aikaisia säätöjä, kun havaitaan poikkeamia järjestelmän normaaleista parametreista. Rakennusautomaatio vähentää kuukausimaksuja (vesi, lämpö, viemäröinti, sähkö) käyttämällä järjestelmiä mahdollisimman taloudellisessa tilassa ja siirtymällä automaattisesti rakennustekniikan päiväkäyttötilasta yökäyttötilaan, jolloin valaistus ja ilmastointilaitteet kytkeytyvät automaattisesti pois päältä ja lämpötilaa alennetaan niissä tiloissa, joiden henkilökunta on poistunut

rakennuksesta. Rakennusautomaatio vähentää huoltokustannuksia, koska useimmat järjestelmät toimivat automaattisesti, mikä vähentää henkilöstön huolimattomuudesta tai käyttäjän virheestä johtuvien kalliiden laitteiden korjaus- tai vaihtokustannuksia.

Älykkäiden rakennusjärjestelmien lisäkustannusten poistaminen, kun laajennetaan hyötyjärjestelmien määrää ja päivitetään niitä hyödyntämällä rakennuksen hallintajärjestelmän avoimen arkkitehtuurin ominaisuuksia.

## 4.3. Älykäs rakennus -konseptin toiminnalliset näkökohdat

Älykkään rakennuksen omistaja voi odottaa seuraavia hyötyjä sen lisäksi, että rakennuksen hallinnan ja ohjauksen maksimaalisen automatisoinnin ansiosta rakennuksen järjestelmiä käyttävien henkilöiden määrä vähenee huomattavasti.

> Teknisten järjestelmien käyttöajan pidentäminen laitteiden optimaalisten toimintaolosuhteiden automaattisen ylläpidon ansiosta. Hätätilanteessa laitteiden toimintaa valvovilla operaattoreilla on täydelliset tiedot kunkin järjestelmän toiminnasta ja RAU-suositukset, joiden perusteella he voivat valita parhaan ja turvallisimman tavan selvitä tilanteesta. Samaan aikaan rakennusautomaatiojärjestelmä huolehtii suurimmasta osasta tehtävistä. Laitehäiriön sattuessa BMS ilmoittaa hyvissä ajoin laitteen toiminnasta vastaaville huoltopalveluille sekä päähuoltopalvelulle ja siihen liittyville yksiköille. Laitteiden ja teknisten järjestelmien huoltokustannukset ovat minimaaliset; kun kaikkien järjestelmien parametreja seurataan ympäri vuorokauden ja kun huoltomiehiä kutsutaan ajoissa paikalle, laitteita korjataan harvemmin. RAU-järjestelmä tallentaa kaikki automaation ja järjestelmän käyttäjien toimet, joten tapahtuneita prosesseja voidaan analysoida optimoimiseksi ja ongelmien ehkäisemiseksi.
> (Surina 2007, s. 133)

Älykkään rakennuksen takaisinmaksuaika riippuu pääasiassa automaatiojärjestelmän monimutkaisuudesta, resurssikustannuksista ja rakennuksen vuokrasta.

## 4.4. Älykkään rakennuskonseptin ympäristönäkökohdat

Energiansäästölaitteiden, älykkäiden ohjausjärjestelmien ja ympäristöystävällisten teknologioiden käyttö viihtyisän ympäristön ylläpitämiseksi älykkäässä rakennuksessa tekee sen mahdolliseksi:

- Luo terveellinen ja ympäristöystävällinen työympäristö yrityksen työntekijöille tai asukkaille;

- Vähentää rakennuksen kuluttamia resursseja optimoimalla ja poistamalla hukkakulutuksen automaattisesti.
- Saadaan ajoissa tietoa energian ja resurssien ylikulutuksesta ja epäoptimaalisesta käytöstä.

# 5. Rakennusautomaation rakennus ja rajapinnat

RAU-järjestelmät rakennetaan käyttämällä ala-asemia (valvonta alakeskus, VAK), jotka perustuvat nykyaikaisiin ohjelmoitaviin logiikkaan, jotka tukevat useimpia kommunikointistandardeja. RAU-järjestelmän valvomo usein perustuu palvelimiin, joissa on SCADA-ohjelmisto (SCADA, Supervisory control and data acquisition). Rakennuksen teknisiin osajärjestelmiin voi kuulua erilaisia älykkäitä laitteita, joilla on erityyppisiä rajapintalähtöjä. Ainoa vaatimus laitteille on "avoimen" tiedonsiirtoprotokollan saatavuus, jonka useimmat valmistajat nykyisin tarjoavat.

## 5.1. RAU-järjestelmän liitännät

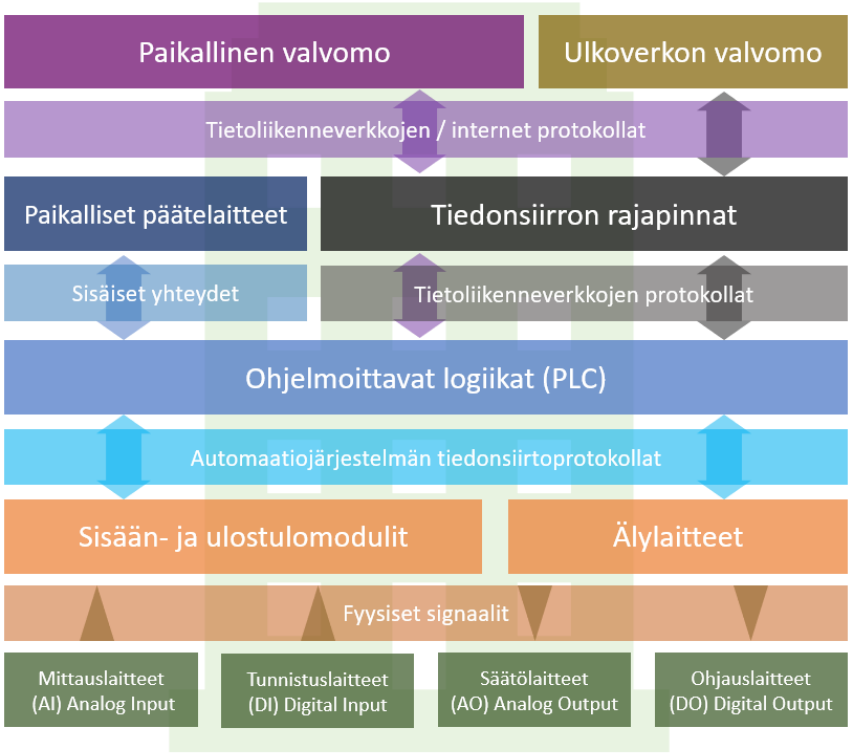
Vapaat ohjelmoitavat logiikkaohjaimet (PLC), jotka käyttävät sekä avoimia että omia protokollia ja tiedonsiirtostandardeja, kuten Modbus, BACnet, M-Bus, KNX, ProfiNet, LonWorks, valvovat ja ohjaavat rakennuksessa olevien teknisten järjestelmien toimintaa sekä vaihtavat tietoja RAU-järjestelmän muiden PLC:n kanssa. Kerättyjen tietojen perusteella PLC:t lähettävät itsenäisesti ohjauskomentoja muille teknisen järjestelmän ohjaimille niiden algoritmien mukaisesti, joilla ne reagoivat tapahtumiin tavanomaisissa tai hätätilanteissa. Mittaustiedot ja käsitelty tiedot voidaan siirtää yhdistettyinä signaaleina käyttäjän tietoliittymään. Liitäntöjen kaksisuuntaisen arkkitehtuurin ansiosta sekä PLC:ille että päätelaitteille voidaan antaa käsikomentoja, joiden avulla järjestelmän toimintaan voidaan tarvittaessa puuttua nopeasti.

Tämä RAU-järjestelmän arkkitehtuuri mahdollistaa:

- Valvoa ilmanvaihdon, lämmityksen, veden, valaistuksen, turvajärjestelmien ja muiden järjestelmien toimintaa automaattisessa tilassa ja varmistaa, että henkilökunnalle taataan mahdollisimman miellyttävät olosuhteet kussakin huoneessa lämpötilan, kosteuden ja valaistuksen osalta;

- Saada puolueetonta tietoa kaikkien järjestelmien toiminnasta ja kunnosta ja ilmoittaa viipymättä lähettämöille tarpeesta puuttua heidän työhönsä tai kutsua paikalle huoltoasiantuntijoita, jos jonkin järjestelmän parametrit poikkeavat normaaliarvoista;
- Mittaamalla asiaankuuluvia parametreja rakennuksessa ja järjestelmien työkuormaindikaattoreita voidaan muuttaa laitteiden ja järjestelmien toimintatapoja, varmistaa niiden tehokas käyttö ja säästää energiavaroja;
- Asettaa optimaalinen ohjaustapa teknisille laitteille, jotta voidaan vähentää rakennuksen teknisten järjestelmien kuluttamien energiavarojen (lämmin ja kylmä vesi, kaukolämpö, sähkö, ilmanvaihto jne.) käyttökustannuksia.
- Varmistaa keskitetty valvonta ja hallinta poikkeus- tai hätätilanteissa:
- Paikallistaa hätätilanteet ajoissa;
- Tehdä nopeita päätöksiä hätätilanteissa ja muissa kuin tavanomaisissa tilanteissa (tulipalo, tulva, vesi- tai kaasuvuoto, luvaton pääsy).
- Tilastojen pitäminen mittauksista ja laiteparametreista, jotta järjestelmien toiminnasta voidaan tehdä objektiivinen analyysi, joka perustuu tehtyjä päätöksiä ja laitteiden toiminnan tehokkuutta koskeviin dokumentoituihin tietoihin;

Laitteiston ja siirtojärjestelmien osalta arkkitehtuuri koostuu kolmesta pääkerroksesta:
- Paikalliset (pääte-) laitteet (mittaus- ja käyttölaitteet)
- Ohjelmoitavat logiikat
- Käyttäjän käyttöliittymä




Kuva 5. RAU-järjestelmän liitännät

Tasojen väliseen tiedonsiirtoon käytetään erilaisia fyysisiä signaaleja, välineitä ja protokollia. Tarkastellaan rakennusautomaatiojärjestelmää sen päätasojen ja -elementtien sisäisen ja niiden välisen viestinnän kannalta.

### 5.1.1. Paikallislaitteiden liitynnät

Paikallisella laitetasolla nämä ovat fyysisiä signaaleja, jotka siirretään suoraan johtoja pitkin: jännitteiden, virtojen ja vastusten mittaaminen tai ohjaaminen. Näissä tapauksissa esimerkiksi paineenmittauslaite muuntaa ilmanvaihtokanavassa olevan ilmanpaineen voiman jännitteeksi. Jännitelähtö on vakioalueella 0–10 Volttia DC, ja se vaihtelee laitteen mittausalueen mukaan. Jos laitteen mittaama enimmäispaine on 2500 Pascalia, ilmakanavan sisäinen paine 1250 Pascalilla antaa 5.0 Voltin lähtöjännitteen. Tämä jännite muunnetaan numeeriseksi arvoksi I/O-moduulin tasolla tai älylaitteessa.

I/O-moduulin numeerinen arvo siirretään yleensä ohjelmoitavaan ohjaimeen sisäisten tiedonsiirtoverkkojen kautta käyttäen suljettua tai avointa protokollaa. Mittaustulosten tulkinta tapahtuu yleensä ohjelmoitavassa ohjaimessa tietyntyyppisen mittauslaitteen ja sen mittausalueiden vaatimustenmukaisuustaulukoiden mukaisesti. Älykäs laite tulkitsee

yleensä mitatun fyysisen signaalin sisäisen ohjelman mukaisesti ja antaa ohjelmoitavalle ohjaimelle vakiomuotoisen numeerisen arvon tietoverkkoon.

Toimilaitteille on tyypillistä päinvastainen järjestys: ohjelmoitavasta ohjaimesta tuleva ymmärrettävä signaali, esimerkiksi käsky asettaa ilmanvaihtokanavan portin avautuminen 60 prosenttiin maksimitasosta, lähetetään I/O-moduuliin numeerisena arvona, minkä jälkeen moduuli muuntaa tämän arvon lähtöjännitteeksi (6.0 Voltin tasajännite), joka johdotetaan toimilaitteeseen, jonka minimiaukko on määritelty 0.0 Voltin tulojännitteellä ja maksimiaukko 10.0 Voltin jännitteellä. Näin ollen portti asetetaan 60 prosenttiin enimmäistasosta. Älykäs laite tulkitsee tämän numeerisen arvon signaalitasoksi sisäisen ohjelmansa mukaisesti ja lähettää ohjausjännitteen toimilaitteelle.

### 5.1.2. Ohjelmoittavan logiikan liitynnät

Ohjelmoitavien logiikoiden, moduulien ja älykkäiden laitteiden väliseen tiedonsiirtoon käytettävien protokollien perhe koostuu sekä suljetuista (omistusoikeudellisista) että avoimista protokollista, jotka toimivat eri fyysisen kerroksen siirtoporttien kautta. Tähän tarkoitukseen käytetään tyypillisesti ”hidas” nopeuksia siirtoprotokollia, jotka ovat erittäin luotettavia, eikä välitettävän verkon ja välitettävän median laadusta tarvitse huolehtia.

Esimerkiksi fyysisen kerroksen RS485-siirtoportti käyttää kaksijohtimista linkkiä, jonka pituus on jopa 1200 metriä, ja sen avulla voidaan liittää jopa 256 orjalaitetta yhteen Master-laitteeseen. Tiedonsiirtonopeus voi olla jopa 1 megabitti sekunnissa. Koska suureissa kohteissa, kuten rakennukset ja kerrostalot, prosessit liikkuvat suhteellisen hitaasti, useimmissa rakennusautomaatiojärjestelmissä jopa 100 kilobitin sekuntinopeus riittää mittausten keräämiseen ja muutosten tekemiseen. Näin ollen tällaisten verkkojen asennuksen helppous, alhaiset kustannukset ja vikasietoisuus nousevat etusijalle. Fyysisen kerroksen protokollat, kuten RS485, M-Bus ja Ethernet, täyttävät korkeimmat vaatimukset kaikissa näissä suhteissa, ja niitä käytetään laajalti rakennusautomaatiossa.

Nykyaikaisissa projekteissa on usein mahdollista, että käytössä on useita fyysisiä siirtoportteja ja viestintäprotokollia samanaikaisesti. Integroiduilla logiikoilla varustettujen

älykkäiden laitteiden ja antureiden on siksi oltava yhteydessä rakennusautomaatiojärjestelmään ja vaihdettava sen kanssa mittaustuloksia, ohjauskomentoja ja muita signaaleja. Tällaiset projektit edellyttävät ohjelmoitavilta logiikoilta suurta joustavuutta ja yhteensopivuutta eri viestintästandardien kanssa.

Fyysisten kerrosten päällä toimivien tietoliikenneprotokollien valinta riippuu usein yksittäisestä asennetuista laitteista. Jos esimerkiksi rakennuksessa on älykkäitä M-Bus-lämpötila-antureita ja ilmankäsittelykone, jossa on Modbus RTU -tiedonsiirtoportti, on välttämätöntä, että PLC:ssä on nämä portit ohjelmoittava. Näiden porttien ja protokollien avulla ohjelmoitava logiikka vastaanottaa lämpötilan mittaustulokset M-Bus-portista, käsittelee ne (esim. laskemalla keskilämpötilan) ja lähettää ohjaussignaalin (tuloilman lämpötilan asetusarvon) ilmanvaihtokoneelle RS485-portissa Modbus RTU -protokollan kautta. Tässä tapauksessa ohjelmoitava logiikka ei toimi ainoastaan valvonta- ja ohjaustoimintona vaan myös tietojen ja viestintäprotokollan muuntimena. Suurissa ja monimutkaisissa rakennusautomaatioprojekteissa voi esiintyä 3–7 tällaista muunnosta.

Järjestelmäkomponenttien välisten protokollien ja tiedonsiirtoporttien oikea valinta on siis erittäin tärkeää nopean, tehokkaan ja kustannustehokkaan projektin toteuttamisen kannalta.

PLC:n tasolla käytetään yleisesti rakenteelliseen kaapelointiin perustuvia standardoituja tiedonsiirtorakenteita, kuten kierretty pariliitoskaapelit tai Ethernet, keskinäiseen ja muiden laitteiden väliseen viestintään. Tällaiset verkot mahdollistavat sellaisen teknisen infrastruktuurin luomisen, joka on hyvin avoin teknisten järjestelmien skaalautuvuuden ja nopean päivittämisen kannalta. Näin ollen on järkevää järjestää sisäiset verkot TCP/IP-protokollapinoon perustuvien laajalle levinneiden tietokoneverkkojen, kuten LAN/WAN-verkkojen, pohjalta. Standardoitujen verkostointimenetelmien sekä standardoitujen laitteiden ja materiaalien käyttö mahdollistaa monimutkaisten projektien toteuttamisen erittäin tehokkaasti.

PLC:n välinen sekä niiden ja ylemmän tason lähetysjärjestelmien välinen tiedonsiirto on suoritettava suhteellisen suurella nopeudella, koska loppulaitteista, älylaitteista ja muista ohjaimista kerättyjen tietojen määrä on paljon suurempi kuin edellisellä tasolla. Tästä

syystä on perusteltua käyttää tietokoneverkkotekniikkaa, joka on suunniteltu suurten tietomäärien nopeaan siirtoon.

Monissa rakennusautomaatioverkoissa on monia TCP/IP-pinon protokollia: HTTP, FTP, UDP sekä erikoistuneet Modbus TCP-IP, BACnet TCP-IP ja BACnet TCP-IP. Kaikki nämä protokollat perustuvat fyysiseen kerrokseen Ethernet LAN (harvemmin WLAN), ja niiden toteuttaminen on suhteellisen helppoa.

### 5.1.3. Käyttäjän liitynnät

Järjestelmän kolmantena kerroksena voi olla tiedonsiirto ohjelmoitavista logiikoista lähetysjärjestelmiin ja operaattorin käyttöliittymiin. Tämä valinnainen mutta hyvin yleisesti käytetty kerros voi koostua sekä erityisistä (esim. suljetut, suojatut tietoliikenneprotokollat paikallisen ohjaimen näyttöön) että yleisistä protokollista ja tietoliikenteen rakentamismenetelmistä. Tiedonsiirto tällä tasolla voi olla nopeudeltaan ja turvallisuudeltaan vieläkin vaativampaa, koska siihen voi liittyä ulkoisia ja julkisia verkkoja ja tiedot voivat olla luonteeltaan arkaluonteisia. Tällä tasolla voidaan myös siirtää suuria määriä tilastollisia mittaustietoja, mikä edellyttää suurta verkon kaistanleveyttä ja luotettavia tiedonsiirtoprotokollia.

Nykyaikaisissa automaatiohankkeissa käytetään perinteisten sisäisten valvomoiden ja tietokantojen ohella laajalti pilvipalvelimia tilastojen siirtämiseen tietokantoihin ja lähettämissovellusten käyttämiseen. Luotettavat viestintäjärjestelmät Internetin kanssa ovat usein olennainen osa rakennusautomaatiohankkeita. Näiden elementtien yhteenliittymiseen on nykyään useita eri tapoja: perinteinen LAN/WAN/WLAN, langaton Internetyhteys matkaviestinverkkojen kautta, optiset tietoverkot jne. Tämän kerroksen protokollien joukosta löytyy sekä pitkään käytetty OPC Unified Architecture (OPC UA) että suhteellisen uusia protokollia, kuten BACnet.

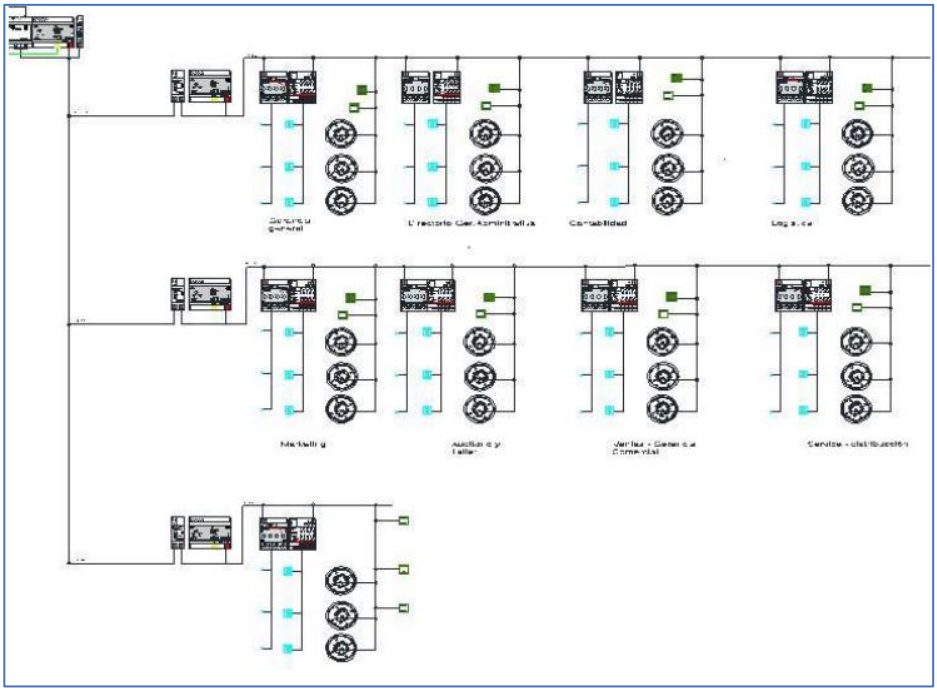
## 5.2. Tiedonsiirtoprotokollat

Seuraavassa tarkastelemme rakennusautomaatiojärjestelmien yleisimpiä viestintäprotokollia ja vertailemme niiden parametreja ja ominaisuuksia.

### 5.2.1. KNX

KNX on yksi yleisimmistä ratkaisuista, joita käytetään keskisuurissa ja suurissa koti-, toimisto- ja liikerakennusten automaatiojärjestelmissä. KNX:n tärkeimpiä ominaisuuksia ovat eri valmistajien tuotteiden taattu yhteensopivuus, yhtenäinen ohjelmistotyökalu (Engineering Tool Software, ETS) projektin suunnittelua, suunnittelua ja toteutusta varten.

KNX-projektien peruskonfigurointia voidaan pitää hajautettuna - tietojenvaihto laitteiden välillä tapahtuu suoraan ilman erillistä erikoistunutta ohjainta. Tällä lähestymistavalla on sekä etuja että haittoja, ja sitä on tarkasteltava hankkeen erityistehtävien yhteydessä. Tällä tavoin on esimerkiksi mahdollista toteuttaa itsenäinen segmentti, joka ohjaa talon valaistusta ohjelmoitujen kohtausten ja algoritmien perusteella. On kuitenkin tärkeää tiedostaa, että laitteet itsessään ovat suhteellisen yksinkertaisia, ja jos tarvitaan monimutkaisempia vuorovaikutusalgoritmeja, on asennettava lisäohjain. (Chamorro-Atalaya 2020)

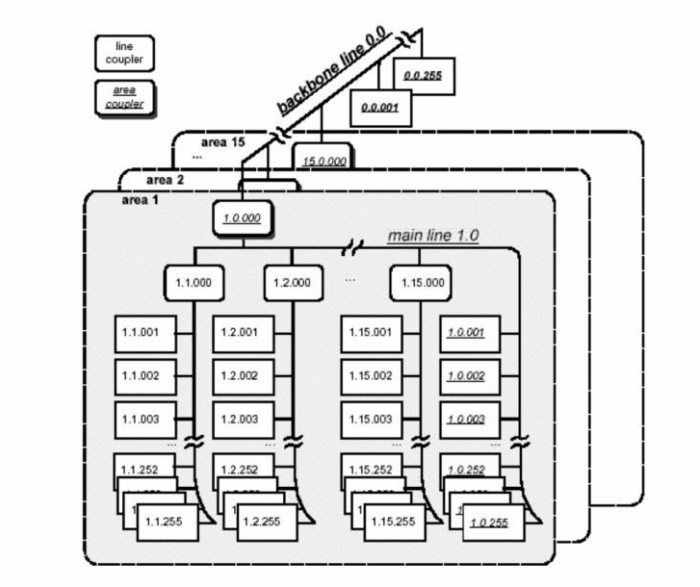


Kuva 6. KNX verkon rakenne (Chamorro-Atalaya, 2020, 1189)

Laitteen osoitteen määräyksessä käytetään yleensä kenttärivi-laite -järjestelmää. Osoitekentän koko on 16 bittiä. Tässä tapauksessa omat osoitteet on kirjoitettava kuhunkin laitteeseen järjestelmän ohjelmointivaiheessa ETS:n ohjelmiston kautta. Tämä toiminto

edellyttää fyysistä pääsyä (yleensä painamalla rungossa olevaa painiketta), ja kun osoite on asetettu, kaikki toiminnot voidaan suorittaa etänä. Näitä osoitteita voidaan sitten muuttaa. Viimeisimpiin sukupolviin on lisätty yksilölliset sarjanumerot, jotka ovat kätevämpiä ohjelmointia varten, ja lisäsuojaus laitteen tietojen etälukemista ja -kirjoittamista vastaan (4 tavun koodin tarkistus).

Ryhmäosoitteet ovat järjestelmän tärkeitä loogisia elementtejä. Ne ovat toiminnon mukaan koottuja laitteita. Anturi/sensori (esim. painike) voi lähettää komentoja vain yhteen ryhmään, kun taas toimilaitteet (esim. rele) voivat vastaanottaa tietoja useista ryhmistä samanaikaisesti. Huomaa, että kaikilla ryhmän laitteilla on oltava samat tietotyypit. Et voi esimerkiksi yhdistää binäärisignaalin lähettämistä himmennyskytkimestä. Usein on kuitenkin niin, että sama laite voi lähettää tai vastaanottaa erityyppisiä tietoja, mikä voi auttaa tässä tilanteessa. Esimerkiksi himmennin voi tarjota käyttöliittymän useisiin ryhmäkohteisiin ja ymmärtää on/off-, himmennin ylös/alas ja prosenttiosuuden asetusarvon himmennyskomentoja. (KNX Association 2013)




Kuva 7. KNX topologian kaavio (KNX Association, 2013, 13)

Tällaisen järjestelmän avulla voidaan yksinkertaistaa laiteryhmän ohjausta lähettämällä yksi viesti ryhmäosoitteeseen yksittäisten osoitteiden sijasta. Ryhmäosoitteiden enimmäismäärää koskevat rajoitukset ovat yleensä yksilöllisiä, ja ne on määritetty laitteiden teknisissä tiedoissa. Rakenteen yksinkertaistamiseksi ryhmäosoitteet voidaan jakaa

tiettyihin luokkiin. Esimerkiksi "lattia-huone-valaistus" -järjestelmän mukaan. Ryhmäkentän koko on myös 16 bittiä.

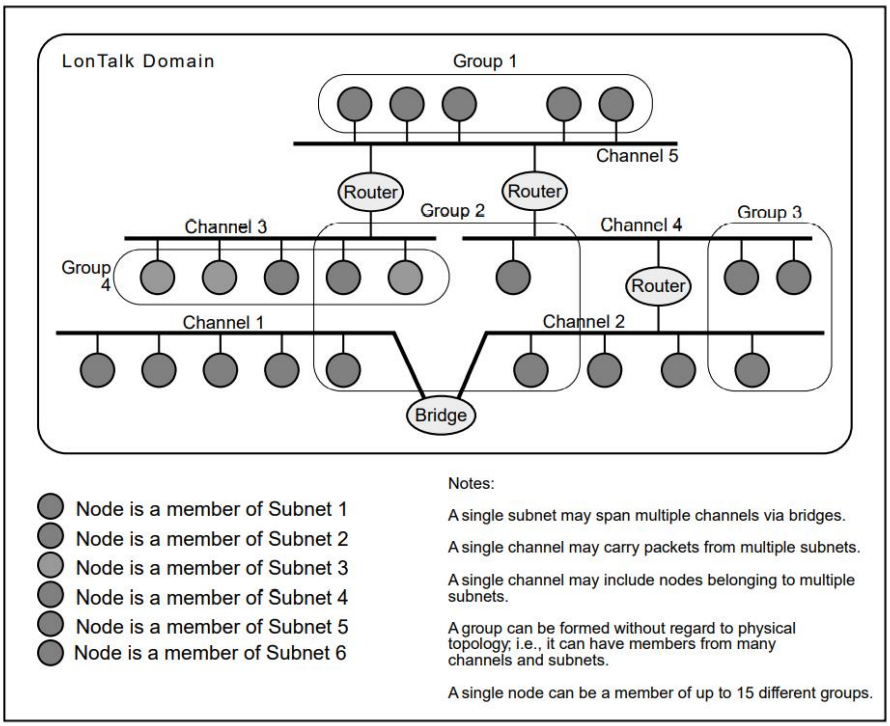
Erillinen laiteluokka ovat ohjaimet. Ne on varustettu omalla prosessorilla, KNX-väyläsovittimella ja niissä voi olla myös muita liitäntöjä. Täällä ei ole rajoituksia, vaan kehittäjä määrittelee kaiken. Nämä laitteet pystyvät moniin lisätoimintoihin, kuten ajastin- ja aikatoimintoihin, kohtauksiin, loogisten olosuhteiden (esim. anturin tilan) tarkistamiseen sekä vuorovaikutukseen ulkoisten laitteiden ja muiden järjestelmien kanssa.

KNX-protokollalla on seuraavat edut:
- Protokolla on avoin, eikä se vaadi omia laitteistoja ja ohjelmistoja.
- Se mahdollistaa laitteiden yhdistämisen useilla eri tavoilla ja rakenteellisten aliverkkojen ja sektorien järjestämisen.
- Se voi siirtää virtaa laitteisiin suoraan tietoväylän kautta (rajoituksin).

Protokollalla on seuraavat haitat:
- Laitteiden osoitteisuus ja ohjaus on monimutkaista ja rajoitettua.
- Sen tiedonsiirtonopeus on rajoitettu.
- Pienemmissä verkoissa käytetään suhteellisen kalliita laitteita.
- Verkkosuunnittelu on tehtävä huolellisesti, ja muutoskustannukset ovat suhteellisen korkeat.

### 5.2.2. LonTalk (LonWorks)

Yhdysvaltalainen Echelon-yritys kehitti LON (Local Operating Networks) -hajautetun ohjausjärjestelmätekniikan yleisiin teollisuussovelluksiin. LonWorks ei ole pelkkä hajautettu ohjausjärjestelmä vaan pikemminkin järjestelmä, jossa on hajautettua älykkyyttä, jossa laitteet voivat itsenäisesti käsitellä tapahtumia ja jossa on sulautettuja ohjelmistoja. LonWorks-tekniikan optimaalinen sovellus on rakennusautomaation kenttätason verkko, jossa on pitkät tiedonsiirtolinjat ja suhteellisen vähän segmenttien sisäistä liikennettä. (Echelon 1994)




Kuva 8. LonTalk verkon rakenne (Echelon. 1994, 14)

LON-verkoissa on useita erityispiirteitä:

- Tekniikka tukee erilaisten signaalinsiirtovälineiden käyttöä sekä langallisena (kuparikierretty pariliitos, valokuitu) että radiotaajuus-, infrapuna- tai voimajohtosiirtona.
- LonWorks-verkon laitteet ovat samanarvoisia. ”Master-” ja ”Slave”-laitteita ei ole jaettu.
- Tekniikka toteuttaa tapahtumapohjaisen logiikan: Toisin kuin keskitetyissä ohjausjärjestelmissä, joissa pääosa verkkoliikenteestä syntyy orjalaitteiden kyselystä, LON-verkoissa liikenne perustuu paketteihin, jotka välittävät tietoa ympäristöparametrien muutoksista (tapahtumista).
- Kierrettyyn pariliitäntään perustuvat ohjausverkot (noin 85 % kaikista LonWorks-kanavista) tukevat erilaisia topologioita, myös vapaita.
- Jokaisessa laitteessa on sulautettu ohjelmisto, ja ohjelmat suoritetaan tapahtumapohjaisesti.

Neuron-siru on keskeinen osa LonWorks-tekniikkaa. Jokaisessa LonWorks-verkon solmussa on oltava tämä komponentti laitteistossaan. Jokaisella Neuron-sirulla on yksilöllinen 48-bittinen tunnistuskoodi, joka tallennetaan Neuronin haihtumattomaan muistiin sen valmistuksen yhteydessä (nimeltään Neuron ID). LonTalk-protokolla on tarkoitettu

käytettäväksi ohjaussovelluksissa, ei datakeskeisissä sovelluksissa. Tämä protokolla käyttää kaikkia ISO-standardimallin 7 kerrosta verkon tiedonsiirtoon. Tämä protokolla on toteutettu Neuron-sirulla, mutta se voidaan toteuttaa myös muulla vastaavalla prosessorilla.

LonTalk-protokollalla on seuraavat edut:
- Laitteiden osoite- ja hallintatapa on riittävän yksinkertainen, jotta sekä yksinkertaiset että monimutkaiset verkot voidaan järjestää.
- Se pystyy yhdistämään laitteita useilla eri tavoilla ja järjestämään jäsenneltyjä aliverkkoja ja sektoreita.
- Ei rajoitteita tiedonsiirron fyysisellä tasolla.

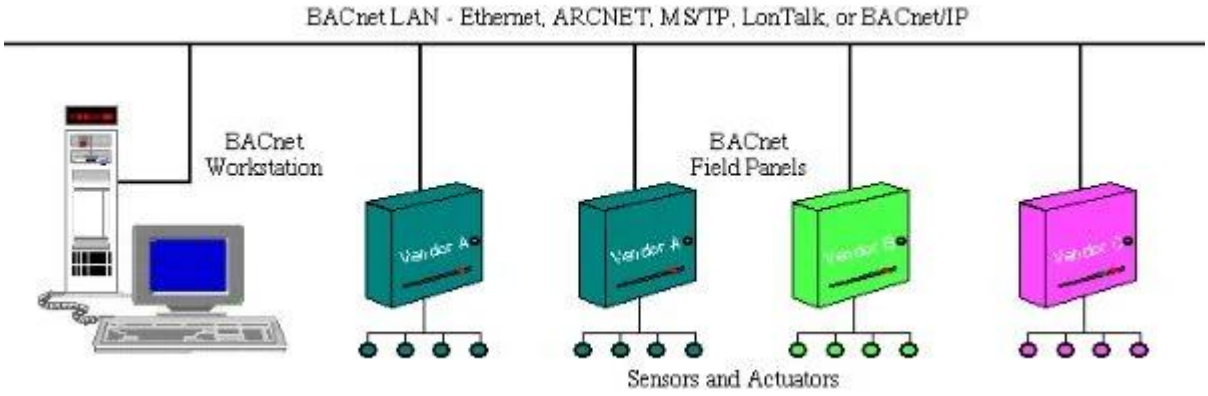
Protokollalla on seuraavat haitat:
- Protokolla on suljettu, ja se edellyttää sekä suojattua laitteistoa että ohjelmistoa.
- Pienissä verkoissa käytetään suhteellisen kalliita laitteita.
- Verkkosuunnittelu on tehtävä huolellisesti, ja muutoskustannukset ovat suhteellisen korkeat.

### 5.2.3. BACnet

BACnet on lyhenne sanoista Building Automation Control network (rakennusautomaatio-ohjausverkko), ja se on ASHRAE:n kehittämä rakennusautomaation viestintäprotokolla (ANSI/ASHRAE-standardi 135-2001), joka on äskettäin päivitetty standardiksi ISO 16484-5. BACnetin päätarkoituksena on standardoida eri valmistajien rakennusautomaatiolaitteiden yhteistoimivuus, mikä mahdollistaa tiedonvaihdon ja yhteistyön laitteiden välillä.

BACnet-laitteet ovat fyysisesti samankaltaisia kuin muut tavalliset rakennusautomaatiolaitteet, jotka sinun pitäisi tuntea, mutta niiden fyysinen muoto ei ole pääpaino, koska BACnet on vain joukko sääntöjä rakennusautomaatiojärjestelmien laitteiden välistä viestintää varten. Näiden laitteiden mikroprosessorit ovat ohjelmoitavia, mikä tarkoittaa, että ne pystyvät "ymmärtämään" toisiaan ja täyttämään BACnet-protokollan vaatimukset.

Koska BACnet on rakennettu asiakas-palvelin-malliin, tämän protokollan viestejä kutsutaan palvelupyynnöiksi. Asiakaskone lähettää palvelupyynnön palvelinkoneelle, joka sitten itse tarjoaa palvelun ja raportoi sen tulokset asiakkaalle. Tällä hetkellä BACnet tukee 35 viestityyppiä, jotka on jaettu viiteen ryhmään (tai luokkaan). Yksi luokka sisältää esimerkiksi viestejä, joilla voidaan käyttää ja hallita edellä kuvattujen objektien ominaisuuksia. Yleisin palvelupyyntö on "ReadProperty". Tämä viesti kehottaa palvelinta hakemaan halutun objektin pyydetyn ominaisuuden ja lähettämään sen arvon takaisin asiakkaalle. Muut viestiluokat käsittelevät hälytyksiä ja tapahtumia, tiedostojen lataamista ja lataamista, etäobjektien hallintaa ja virtuaalipäätetoimintoja. (Niazi, 2020)




Kuva 9. BACnet verkon rakenne (Niazi, M.A., 2020)

BACnet voi käyttää erilaisia fyysisiä välineitä viestien lähettämiseen. Ethernet nopeimpana verkkoyhteysmenetelmänä: 10–100 Mbit/s. Seuraavana on ARCNET, jonka nopeus on 2,5 Mbit/s. Ethernet ja ARCNET voivat molemmat käyttää erilaisia fyysisiä välineitä: koaksiaalikaapelia, kierrettyä parikaapelia tai jopa valokuitukaapelia. Laitteille, joiden tiedonsiirtonopeusvaatimukset ovat alhaisemmat, BACnet määrittelee MS/TP-verkot (Master-Slave/token-passing), jotka pystyvät toimimaan jopa 1 Mbs:n nopeudella. BACnet voi käyttää myös puhelinlinjoja tai EIA-232:ta valinta- tai pisteestä pisteeseen -tiedonsiirtoprotokollia (PTP). Tärkeintä on, että BACnet-viestit voidaan periaatteessa toimittaa minkä tahansa olemassa olevan verkkotekniikan kautta.

BACnet-protokollalla on seuraavat edut:

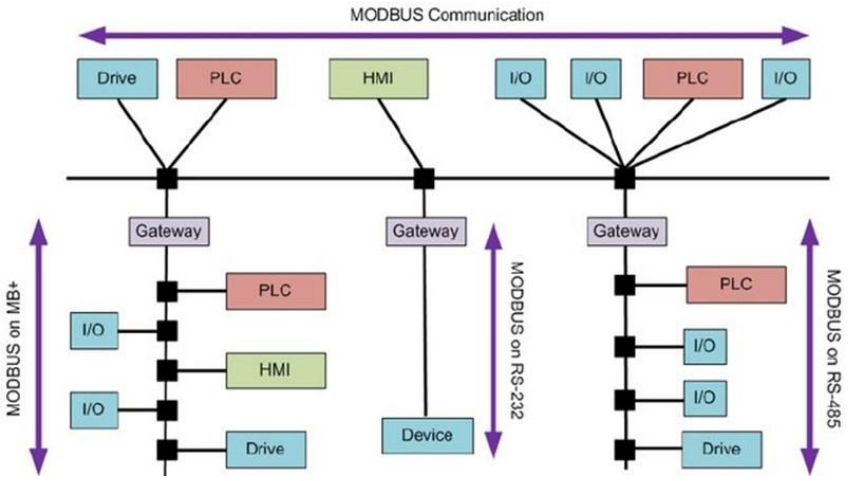
- Laitteiden osoite- ja hallintatapa on riittävän yksinkertainen, jotta sekä yksinkertaiset että monimutkaiset verkot voidaan järjestää.

- Se pystyy yhdistämään laitteita useilla eri tavoilla ja järjestämään jäsenneltyjä aliverkkoja ja sektoreita.
- Kykenee yhdistämään toisistaan poikkeavia laitteita, aliverkkoja ja suuria verkkoja rakenteeksi, joka jakaa tietoverkon resursseja muiden järjestelmien kanssa (ei BACnet).
- Sillä ei ole merkittäviä rajoituksia fyysisen tiedonsiirron tasolla.

Protokollalla on seuraavat haitat:
- Valmistajat tukevat protokollaa suhteellisen hitaasti ja laitteiden määrä on rajallinen.
- Melko monimutkainen ohjelmistototeutus protokollan tukemiseksi

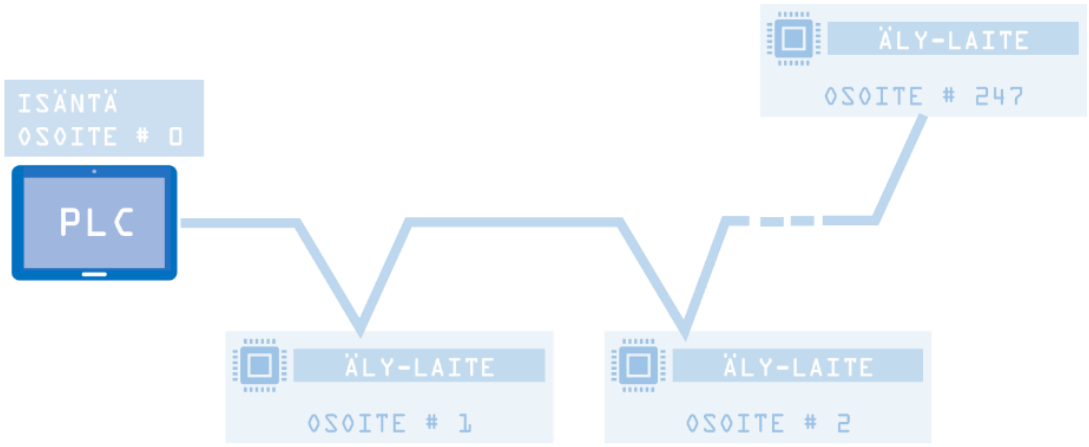
### 5.2.4. Modbus

Modbus-protokolla on 7-tasoinen OSI-viestintämalli, ja se tarkoittaa, että viestien tuottaminen voi tapahtua suoraan käyttäjäsovelluksessa, ja kaikkien muiden tasojen (esitystapa, istunto, kuljetus, verkko, datayhteys) oletetaan olevan jo valmiita ja toimivia. Tämä tekee käyttäjän roolista helpon ja vaatii vain oikean komennon antamista.




Kuva 10. Modbus verkon rakenne (Modbus Organization. 2012, 3)

Modbus-protokolla toimii "Master-Slave"-muodossa (Client-Server), mikä tarkoittaa, että Master-laite (isäntä) on ainoa tiedonsiirron aloittaja, ja kaikki muut laitteet kuuntelevat linjalla lähetettyjä viestejä ja aloittavat vastauksen lähettämisen vasta, kun Master-laite kytketään pois päältä. Tällainen muoto varmistaa, että kysely tavoittaa orjan ja vastaus tavoittaa isännän. Synkronointikysymykset hoidetaan vain isännän puolelta.




Kuva 11. Modbus RTU verkon rakenne (RS485)

Kaikki kyselyt voi käynnistää vain asiakaslaite: protokolla ei salli viestien lähettämistä palvelinlaitteista ilman asiakkaan edeltävää kyselyä.
Modbus-tietopaketti sisältää pysyvän PDU:n (Protocol Data Unit), joka on yhteinen kaikille protokollatoteutuksille ja joka koostuu toimintokoodista ja datasta. Lisäksi voi olla useita erityisiä kenttiä, jotka vaihtelevat verkon fyysisen kerroksen mukaan - yleisimmin palvelimen laiteosoite ja tarkistussumma virheiden havaitsemista varten. Kun otetaan huomioon lisäkentät, koko Modbus-pakettia kutsutaan ADU:ksi (Application Data Unit). Modbus on OSI:n (Open Systems Interconnection model) kerroksen 7 sovellusprotokolla. Se on riippumaton alikerroksista, ja sitä voidaan käyttää yhdessä muiden protokollien, kuten Ethernet TCP/IP:n tai UDP/IP:n, kanssa, ja siinä voidaan käyttää RS-232-, RS-422-, RS-485-, valokuitu-, radio- ja muita sarjaliitäntöjä fyysisenä välineenä signaalien siirtämiseen.

Modbus ASCII. Protokollan muunnelma, joka toimii myös RS-232/RS-485:n päällä, mutta käyttää ASCII-merkkejä viestien koodaamiseen. Modbus RTU (etäpääteyksikkö). Tämä on muunnelma protokollasta, joka verkon fyysisenä kerroksena käyttää yleisimmin RS-485-sarjaliitäntöjä ja harvemmin RS-232- ja RS-422-liitäntöjä. Pohjimmiltaan kaikki nämä liitännät määrittelevät kierrettyyn pariin perustuvan tiedonsiirron. (Modbus Organization. 2012)

Modbus TCP. Tämä on ModBus-järjestelmän toteutus Ethernet-verkoissa. Se toimii TCP/IP-pinon päällä. Toisin kuin Modbus RTU:ssa ja ASCII:ssä, Modbus TCP:ssä

yhteys tiettyyn laitteeseen muodostetaan TCP/IP:n avulla. Tämän vuoksi Modbus-paketissa oleva osoite jätetään useimmiten huomiotta, eikä lähetysviestejä käytetä. Osoitetta saatetaan kuitenkin tarvita, jos yhteys muodostetaan yhdyskäytävään, joka puolestaan muodostaa ulostulon RS485-verkkoon - jotta voidaan edelleen kommunikoida jo Modbus-kieltä käyttävien laitteiden kanssa.

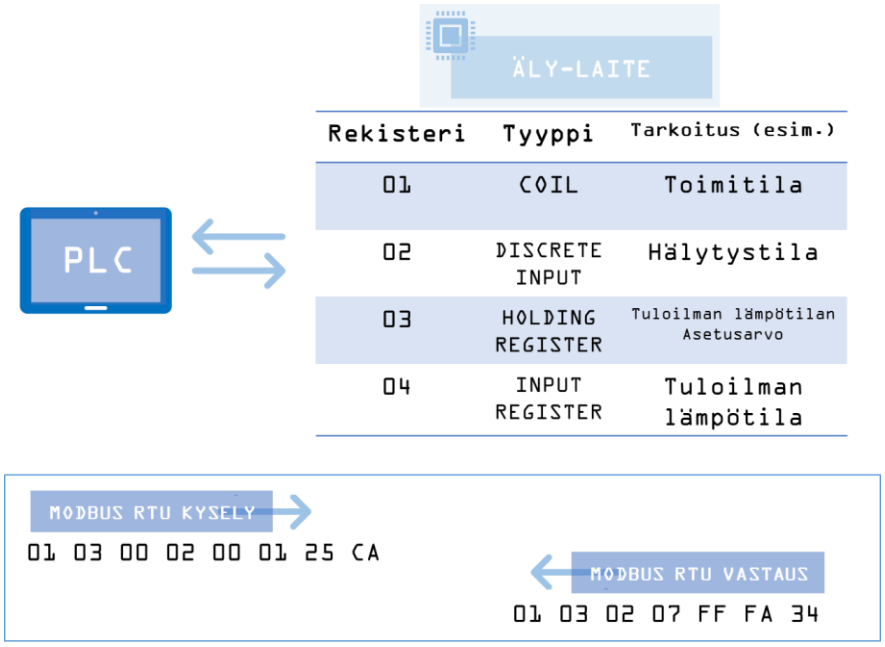
TCP/IP tarjoaa myös pakettien eheydenvalvonnan, joten Modbus-toteutusta ei tarvita.
Modbus RTU-sanoma (ADU Frame) sisältää:
[Slave ID] [Function code] [Data] [CRC] [CRC].
Suomen käännös:
[Slave-osoite 1..247] [Toimintakoodi / rekisterityyppi] [Pyydetyt tiedot / rekisterien alku ja lukumäärä] [Virheentarkistustiedot].

Isäntäkoneen (Modbus Master) pyyntö sisältää esim.
01 03 00 02 00 02 00 02 00 01 25 CA.

```
slave id (1 byte)
|  function code (read holding registers)
|  |      address of first register to read (2 bytes)
|  |      |     number of registers to read (1 byte)
|  |      |     |  checksum (2 bytes)
|  |      |     |  |
01 03 00 02 00 01 25 CA
```

Orjalta (Modbus Slave) tuleva vastaus:
01 03 02 07 FF FA 34

```
slave id (repeats own id)
|  function code (repeats requested code)
|  |   number of bytes of data (2)
|  |   |     the value of the register (0x07FF)
|  |   |     |     checksum
|  |   |     |     |
01 03 02 07 FF FA 34
```




Kuva 12. Modbus tiedon tyypit

Protokollan kuvauksen mukaan voidaan laatia kyselyviesti, joka orjalaitteelle lähetettynä voi lukea tai kirjoittaa mitä tahansa tietoja, jos ne ovat saatavilla orjalaitteesta. Edellä esitettyjen tietojen perusteella on mahdollista määrittää tekniset vaatimukset sellaisen sovelluksen kehittämiseksi tietokoneelle, joka pystyy muotoilemaan päteviä viestejä ja saamaan vastaukset selkokielellä ihmiselle.

Modbus-protokollalla on seuraavat edut:

- - Protokolla on avoin ja monipuolinen, ja tuettujen valmistajien ja laitteiden määrä on valtava.
- - Laitteiden osoitus- ja ohjausmenetelmä on varsin yksinkertainen, ja sen avulla yksinkertaiset verkot voidaan järjestää helposti ja edullisesti.
- - Helppo diagnostiikka ja virheenkorjaus
- - Korkea luotettavuus ja vähäiset resurssivaatimukset
- Protokollaan liittyy seuraavia haittoja
- - Ei sisäänrakennettua tietojen salausta
- - Master/Slave-viestintä rajoittaa järjestelmän laitteiden toimintaa. Ei mahdollisuutta järjestää sektoreita ja aliverkkoja.
- - Käytettävissä olevien tietojen rajallinen määrä ja rajallinen tietojen resoluutio.

### 5.3. Protokollan valinta

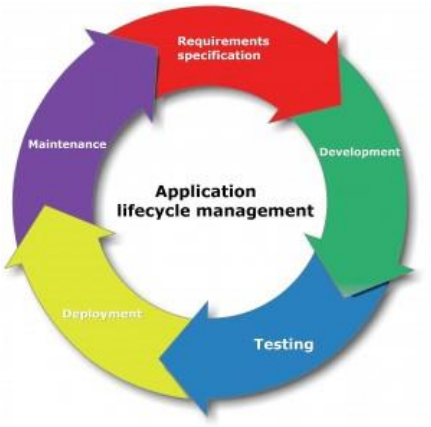
Muiden yritysten tavoin myös Fidelix Oy joutui kehitysvaiheen alussa valitsemaan, mitä siirtoprotokollaa se tukee. Rakennusautomaatiohankkeet, joihin keskityttiin, koostuvat yleensä useista eri valmistajien tuottamista älykkäistä laitteista, joilla on eri käyttötarkoitukset. Näiden laitteiden määrä yhdessä projektissa on pieni, ja ohjelmoitavan ohjaimen on kommunikoitava korkeintaan muutaman kymmenen laitteen kanssa. Älykkäiden laitteiden kustannuksilla, tietoverkon käyttöönoton helppoudella sekä käynnistyksen ja käyttöönoton helppoudella on suuri merkitys rakennusautomaatiohankkeissa.

Suljetut protokollat, kuten LonTalk tai ProfiNet, soveltuvat huonosti tällaisiin heterogeenisiin ja epätyypillisiin hankkeisiin, koska laiteluettelo on suhteellisen suppea ja valmistajille asetetut vaatimukset ovat vaatimattomia. Erikoistuneet protokollat, kuten KNX, rajoittavat mahdollisten laitevaihtoehtojen valikoimaa vakiovarusteita laajemmaksi ja edellyttävät suhteellisen monimutkaista verkon suunnittelua. Modbus-protokolla ja sitä tukevat laitteet täyttävät useimmat rakennusautomaatioprojektien vaatimukset. Tästä syystä yritys on valinnut Modbus-protokollan ohjelmoitavien ohjaintensa ensisijaiseksi protokollaksi ja käyttää Modbus RTU:n ja Modbus TCPIP:n muunnelmia. Muiden protokollien hylkääminen voidaan kompensoida asentamalla muuntimia ja yhdyskäytäviä, jotka voidaan liittää ohjainohjelmaan universaalia Modbus-protokollaa käyttäen. Käyttöliittymien valmistelu Modbus RTU / TCPIP käyttöliittymiä varten on siksi olennaisen tärkeää automaatioprojektien tehokkaan toiminnan kannalta.

## 6. Sovelluksen kehitysprosessi ALM-mallin näkökulmalla

### 6.1. Projektisuunnitelma

Yksi käytetyimmistä kehitysmalleista ALM perustuu jatkuvaan (kiertävään) prosessiin, ja se tarjoaa nopean ja tehokkaan lähestymistavan keskisuurille sovelluksille.




Kuva 13 Kaavio ALM-mallista (PractiTest, 2020)

ALM on yleisesti ottaen tuotantomalli, joten sen avulla kehitetään sovelluksia. Sitä käytettiin hyvin laajasti eri projekteissa ja yrityksissä.
Vaatimusmäärittelyvaiheessa määriteltiin ja kerättiin tulevan sovelluksen vaatimukset, perusominaisuudet, käytetyt teknologiat ja alustavat käyttöliittymäsuunnitelmat. Esimerkiksi valitaan ohjelmointikieli ja -ympäristö, valitaan tiimit (asiantuntijat), suunnitellaan grafiikka ja käyttöliittymä (developer interface).

Kehitysvaiheessa tehdään itse koodaus ja luodaan käyttöliittymät. Kaikki tiimit/kehittäjät työskentelevät omilla aloillaan, ja sovellus kokonaisuudessaan (toiminnalliset lohkot, käyttöliittymä, dokumentaatio jne.) luodaan tuotteena. Tässä vaiheessa voidaan lisätä/poistaa mukautettuja ominaisuuksia ja muuttaa rajapintoja ennen sovelluksen julkaisemista.

Testausvaiheessa sovelluksen ominaisuudet ja käyttöliittymät tarkistetaan sen varmistamiseksi, että ne vastaavat alkuperäistä suunnittelua ja täyttävät ensimmäisessä vaiheessa asetetut vaatimukset. Testaukseen on sisällyttävä sekä toiminnallinen että käyttöliittymätestaus. Tässä vaiheessa korjataan useimmat mahdolliset virheet ja poistetaan toimimattomat lohkot/ominaisuudet, jotta sovellus voidaan julkaista. Asennus/poisto eri koneille/käyttöjärjestelmiin on myös välttämätön osa testausta.

Käyttöönottovaiheessa sovellus valmistellaan myyntiin (julkaisu) ja asennetaan asiakkaan järjestelmiin. Monimutkaisten sovellusten asennus voi vaatia eritasoista tukea kehittäjältä. Loppukäyttäjien asennukset tehdään kuitenkin yleensä ilman tukea. Ylläpitovaiheessa

kehittäjä kerää kaikki vikailmoitukset ja asiakkailta saadun palautteen, jonka perusteella sovellusta korjataan ja parannetaan. Korjaukset ja päivitykset ovat olennainen osa tätä vaihetta. Automatisoitu sovelluksen seuranta voi olla tarpeen.

Kun palaute ja vikailmoitukset on kerätty, kehittäjän voi olla tarpeen luoda alaversio, johon on lisätty ominaisuuksia ja parannuksia asiakkaan tarpeiden mukaan. Tästä eteenpäin aloitetaan uusi ALM-sykli, joka etenee samalla tavalla.

### 6.2. Projektin osien toteutus

Tämän tutkimuksen kohteena oleva sovellus on tarkoitettu pääasiassa automaatioinsinööreille ja kunnossapitohenkilöstölle. Sovelluksen pitäisi pystyä muodostamaan ohjelma ja luettelo parametreista, jotka kommunikoivat Modbus-laitteen kanssa. Ohjelma ja parametrit käynnistetään Fidelix PLC -ala-asemalla (FX-sarja), ja niiden on täytettävä IEC -61131-3-standardien mukaiset vaatimukset.

Sovelluksen perusvaatimukset pyritään määrittelemään:

- Modbus-parametrien ja -ominaisuuksien selkeä valintaprosessi.
- Tarvittavien Modbus-rekisterien, -tyyppien ja -toimintojen nopea valinta.
- Parametrien ja rekisteriluettelon luominen ja tallentaminen tulevaa käyttöä varten.
- IEC-kielisen ohjelman luominen parametrien ja rekisteriluettelon perusteella.
- Ohjelman tallennus ja arkistointi Modbus-laitekohtaisesti (arkistointi).
- XML-tiedostojen käsittely (tuonti/vienti).
- Lisävaatimukset (oletettavasti tulevat seuraaviin versioihin):
- Yhteys Modbus-protokollan kautta kenttälaitteeseen käyttöliittymän testausta varten.
- Suora yhteys PLC-asemaan ohjelman lataamista ja seurantaa varten.

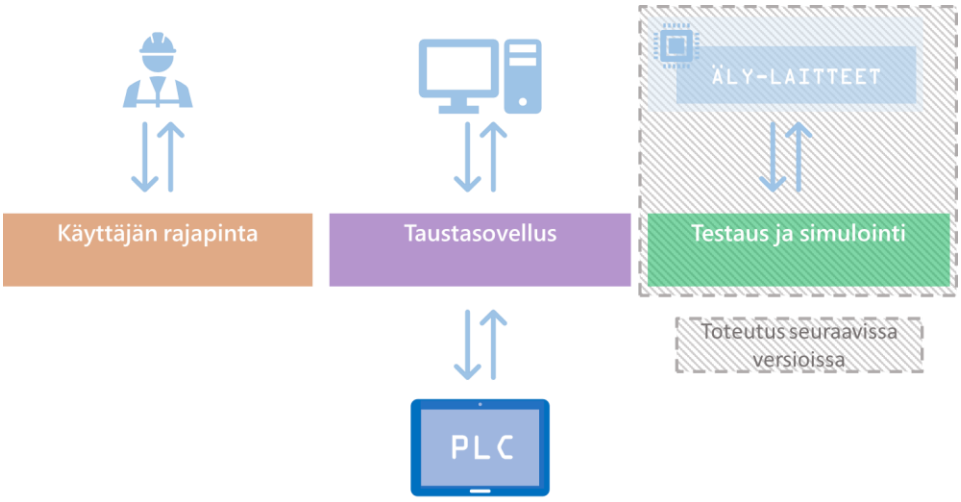
## 7. Toteutus

### 7.1. Toteutuksen suunnitelma

Sovelluksen toteutus Tehty Java-kielillä ja käyttäen JavaFX UI (käyttöliittymä) -kirjastoa. Ohjelmointiympäristönä on käytetty Eclipse SDK:ta (Software development kit), joka

on avoin, luotettava ja jousitettu työkalu, jossa on kaikki tarvittavat ohjelmointiominaisuudet.

Sovelluksen yleinen rakenne voidaan kuvitella seuraavanlaiseksi:




Kuva 14. Sovelluksen toiminnallinen rakenne

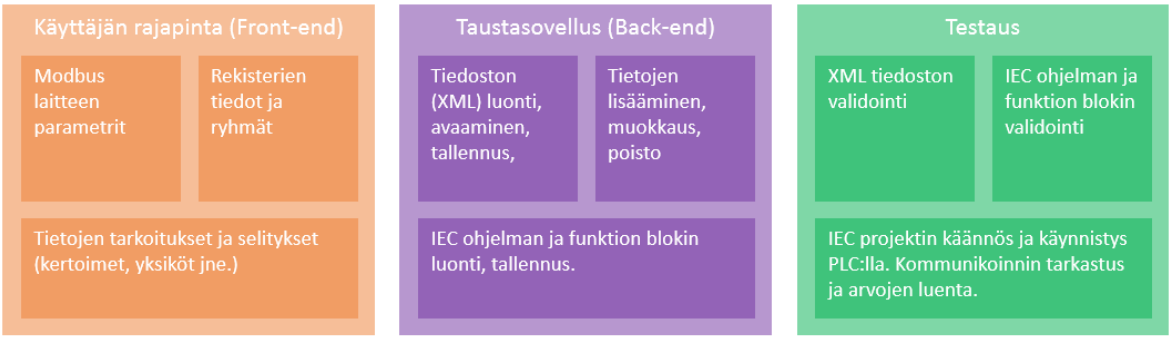
## 7.2. Sovelluksen perustoiminnot

Tarkastellaan kutakin perustoimintoa erikseen:

- Käyttöliittymä koostuu Modbus-yhteyden parametroinnista ja rekisterikokoelman muodostamisesta. Valinta tehdään taulukossa, josta käyttäjä valitsee uudelleen tarvittavat rekisterit ja antaa selitykset ja lisäparametrit. Tiedot siirretään taustasovellukseen käsiteltäväksi.
- Taustasovellus suorittaa seuraavat prosessit peräkkäin:
- Luodaan Modbus-yhteyden parametreista XML-tiedosto, joka sisältää kaikkien rekisterien tiedot ja jota voidaan käyttää arkistointiin.
- Luo IEC 61131-3 -ohjelmointikielen kutsuohjelman (.ST-tiedosto), joka sisältää kaikki PLC-laitteen Modbus-rekisterit (FX-sarjan PLC-yhteensopiva). Ohjelma on tarkoitettu OpenPCS-ohjelmointiympäristöön (Infoteam GmbH).
- Luodaan IEC 61131-3-ohjelmointikielen funktiolohko, joka sisältää kaikki Modbus-rekisteritietojen noudot sarjaportista ja niiden kirjoitukset sarjaporttiin. Toimilohko on FX-sarjan PLC-yhteensopiva. Toimilohko on tarkoitettu OpenPCS-ohjelmointiympäristöön.
- XML- ja .ST-tiedostojen tallennus ja lukeminen. Kansioiden luominen tarvittaessa.

- Testaus- ja simulointilohkot suorittavat seuraavat prosessit.
  - Rekisteritietojen validointi,
  - Virheellisten rekisteriparametrien tunnistaminen.
  - XML-tiedoston validointi

Sovelluslohkot voidaan esittää seuraavasti:




Kuva 15. Sovelluksen lohkorakenne

Sovelluksen jokainen lohko toteutetaan ja testataan erikseen ja rinnakkain. Näin varmistetaan, että lohkot eivät vaikuta toisen lohkon prosessiin, ja virheelliset tapaukset voidaan poistaa ja korjata nopeammin. Mahdolliset virheet tai viat korjataan jokaisessa lohkossa, jolloin lopputuloksena on vaatimusten mukaisesti toimiva ohjelma.

Seuraava lähestymistapa on yhdistää lohkot, siirtää tietoja niiden välillä ja testata ne yhdessä. Toteuta tiedonsiirto siten, että jokaisesta muutoksesta saadaan loki-/virheraportti ja/tai tiedosto. Tämä helpottaisi yhteistä testausta ja tekisi siitä luotettavampaa.

Yhteisessä testausvaiheessa tapahtuu virallinen versionhallinta eli ns. alfaversion julkaisu (käyttöönotto), joka esitellään tilaajalle ja kohderyhmälle. Käännetty ja testattu sovellus olisi esiteltävä loppukäyttäjien testiryhmälle, ja siitä olisi kerättävä palaute ja mahdolliset vikailmoitukset. Näiden perusteella tehdään alustava vaa'an ja ominaisuuksien hienosäätö ja aloitetaan seuraava kehitysvaihe.

Julkaisun jälkeen sovelluksen ylläpito aloitetaan keräämällä asiakaspalautetta, parannusehdotuksia ja toimintalokeja. Kaikki nämä tiedot analysoidaan, ja projektipäällikkö/analyytikko tekee päätökset niiden toteuttamisen ajoituksesta ja resursoinnista.

Kun korjausten ja parannusten määrä on kohtuullinen uudelle versiolle, versio nimetään uudelleen ja päivitys (tai kokonaan uusi sovellus) julkaistaan uudelleen. Seuraava ALM-kierros aloitetaan, kun uudet vaatimukset ja ominaisuudet on määritelty, projektin resursointi ja aikataulutus on tehty.

### 7.3. Tuotos

Sovellus on tarkoitettu Fidelix FX-sarjan PLC-ohjelmoijalle, joka tarvitsee yhteyden Modbus-laitteeseen. FX PLC toimisi tässä tapauksessa Modbus RTU (TCP/IP) "Master"-laitteena (päälaite), joka lähettää, ja kolmannen osapuolen Modbus-laite ("Slave") vastaanottaa viestit ja vastaa omalla sisäisellä ohjelmallaan.

Modbus-"Slave"-laitteen rekisterit on yleensä kuvattu käsikirjassa, ja tämä rekisteriluettelo on sovelluksen lähtötietona. Käyttäjän toiveena on saada IEC-ohjelma, jossa projektiin liittyvien rekisterien lukeminen ja kirjoittaminen on toteutettu. Usein Modbus-laitteen rekisteriluettelo sisältää kymmenestä tuhanteen erilaista rekisteriä, joista osaa ei käytetä projekteissa. Käyttäjä poimii rekisteriluettelosta yleensä tarvittavat rivit ja muodostaa niistä oman projektikohtaisen luettelonsa.
Sovellus alkaa siis esimerkiksi käyttäjälle räätälöidystä rekisteriluettelosta:
Modbus funktio (0x02 Read Discrete Inputs).

- 10001 Releтila Bitti 0–1 0. Pois päältä 1. Käytössä (ei tarvita projektissa).
- 10002 PIR-anturin tila (välitön) Bitti 0–1 0. Ei havaitsemista 1. Havaitseminen
- 10003 PIR-anturin tila (ohjaus) Bitti 0–1 0. Pois päältä 1. Päällä

Tulorekisterit (Modbus-toiminto 0x04 Lue tulorekisterit)

- 30001 CO2-mittaus S16 400...10000 400...10000 ppm 400...10000 ppm
- 30002 Lämpötilan mittaus S16 0...500 0.0...50.0 °C
- 30003 Kosteuden mittaus S16 0...100 0...100 % rH
- 30004 Y1-lähtöjännite U16 0...1000 0.00...10.00 V (Ei tarvita projektissa)

- 30005 Lähtöjännite Y2 U16 0...1000 0.00...10.00 V (Ei tarvita projektissa)

(Produal 2022)

Sovelluksen tuloksena käyttäjä saa IEC-ohjelmat, jotka sisältävät mainitut rekisterit (funktiolohko ja sen kutsu) ja jotka ovat yhteensopivia IEC 61131-3 InfoTeam Open-PCS -ohjelmointiympäristön kanssa.

Kehitysprosessin tuloksena syntyy työpöytätietokonesovellus, joka tarjoaa käyttäjälle graafisen käyttöliittymän, jossa on seuraavat toteutukset:

- Modbus-rekisteriparametrien valinta ja hallinta (rekisterin tyyppi, numero, tekstikuvaus, lukuarvon kerroin, version aikaleima).
- Valittujen rekisterien välimuistiin tallentaminen ja konfiguraatioluettelon näyttäminen.
- Välimuistissa olevien rekisterien muokkaaminen ja poistaminen
- Luettelon tallentaminen tiedostoksi muodossa, joka sisältää kunkin rekisterin kaikki parametrit, ja kyseinen tiedosto luetaan myöhemmin. Tallennetut tiedostot voidaan sekä säilyttää tietovarastossa että jakaa muiden kanssa heidän käytettäväkseen
- Luettelon tallentaminen kiintolevylle myöhempää käyttöä varten
- Tallennetun luettelon lukeminen takaisin, näyttäminen sovelluksessa ja muokkaaminen
- Luettelon tallentaminen IEC-toimintolohkon ja ohjelman muodossa, joka sisältää kaikki Modbus-tiedonsiirron toteuttamiseen tarvittavat ohjelmaoperaattorit ja sisäkkäiset toiminnot.
- IEC-funktiolohko ja ohjelma virheettömiksi, yksinkertaisiksi ja selkeiksi projektin toteuttamista varten.

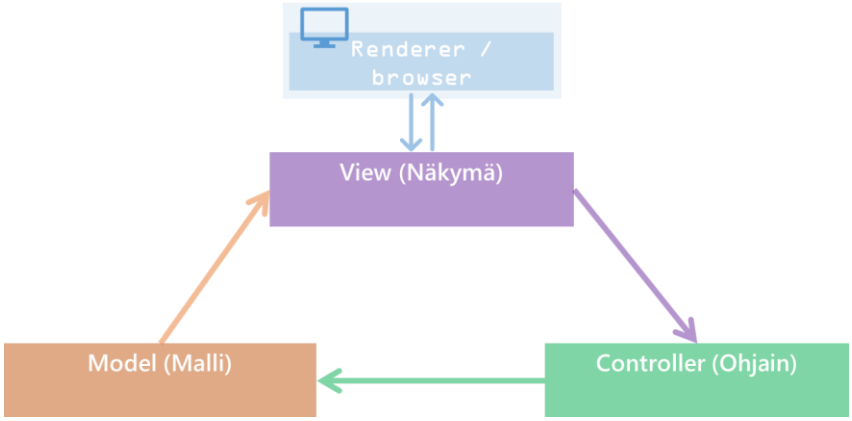
## 7.4. Ohjelmointimalli

Tässä projektissa me valitsimme MVC ohjelmointimalli.

MVC (Model - View - Controller) on ohjelmointimalli tai kehys, jota käytetään kaupallisissa ja verkkosovelluksissa. Sen päätarkoitus on erottaa liiketoimintalogiikka (sovelluslogiikka) ja näkymälogiikka (käyttöliittymä) toisistaan. Tämä erottelu mahdollistaa koodausprosessin yksinkertaistamisen ja erilaisten näkymäelementtien ja tyylien soveltamisen vaikuttamatta sovelluksen muihin osiin. (Oracle 2011)

- Malli: edustaa sovelluksen liiketoimintakerrosta. Ne ovat objekteja ja/tai luokkia, jotka kuljettavat tietoja ja voivat myös sisältää logiikan, jolla ohjainta päivitetään, jos tiedot muuttuvat.
- Näkymä: Se edustaa sovelluksen esityskerrosta. Sitä käytetään mallin sisältämien tietojen visualisointiin.
- Ohjain: Se vaikuttaa sekä malliin että näkymään. Sitä käytetään ohjaamaan sovelluksen kulkua eli tiedon kulkua malliobjektin kautta ja päivittämään näkymää, kun tietoja muutetaan.

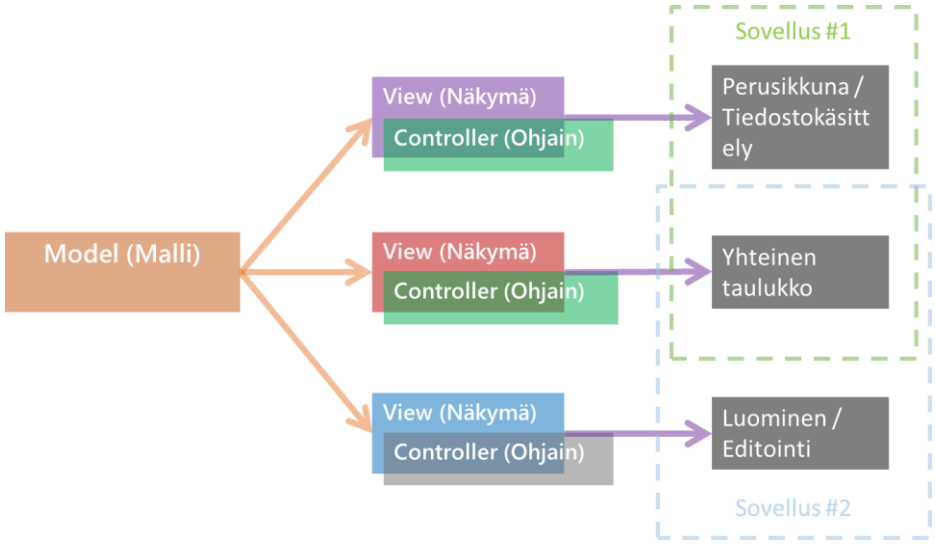
MVC-ohjelmointimalli erottaa datan, ohjauksen ja näkymän vastuut periaatteessa erillisiin objekteihin. Näiden objektien on toimittava yhdessä, jotta ne voivat käsitellä käyttäjän syötteitä, tallentaa tietoja ja visualisoida ne näytöllä.




Kuva 16. MVC-ohjelmointimallin perusrakenne

- Skaalautuvuusominaisuus, joka auttaa sovellusta kasvamaan.
- Komponentteja on helppo ylläpitää, koska riippuvuuksia on vähemmän.
- Mallia voidaan käyttää uudelleen useissa näkymissä, mikä mahdollistaa koodin uudelleenkäytettävyyden.
- Kehittäjät voivat työskennellä kolmen kerroksen (malli, näkymä ja ohjain) kanssa samanaikaisesti.
- MVC tekee sovelluksesta helpommin ymmärrettävän.
- MVC:n avulla kutakin kerrosta ylläpidetään erikseen, joten kehittäjien ei tarvitse käsitellä massiivista koodia.

- Sovellusta on helpompi laajentaa ja testata.
- MVC helpottaa koodin uudelleenkäyttöä useissa ikkunoissa tai sovelluksissa ilman modulaaristen elementtien muuttamista.




Kuva 17. MVC-ohjelmointimallin osien uudelleenkäyttö

MVC-ohjelmointimallin mukaan on tarpeen erottaa sovelluslogiikasta vastaavat Ja-vaFX luokat käyttöliittymän vuorovaikutuksesta vastaavista luokista. Sovelluslogiikkaan kuuluvat seuraavat osat:

- Käyttäjän syöttämien tietojen kerääminen ja "rekisterin konfigurointiobjektien" luominen.
- Syöttötietojen yhdenmukaisuuden tarkistaminen
- "Rekisterit"-tietokokonaisuuden tuottaminen "Rekisterin konfigurointi"-kokoelmasta.
- Tietokokonaisuuden tallentaminen XML-muodossa tulevaa käyttöä varten.
- Tiedostojen tallentaminen IEC-muodossa toimintolohkona ja kutsuohjelmana.

# 8. IEC ohjelmointikielen ohjelma

IEC 61131-3 -ohjelmointikielistandardia käytetään laajalti eri valmistajien PLC-ohjaimissa (Programmable Logic Controller) ympäri maailmaa. Kansainvälisen sähkötekni-sen komission (IEC) ansiosta on syntynyt viisi standardikieltä sekä prosessi- että erillisohjainten ohjelmointia varten:

- Ladder Diagram (LD)
- Toimintolohkokaavio (FBD)
- sekventiaalinen toimintokaavio (SFC)
- käskyluettelo (IL)
- Strukturoitu teksti (ST)

(Thayer, 2009)

Strukturoitu teksti (ST) muistuttaa läheisesti korkean tason tietokoneohjelmointikieltä, kuten Pascalia ja C:tä. IEC61131-3 ST-kieli on yleistynyt eniten, sillä sitä käytetään lukuisissa automaatiohankkeissa, joissa on monimutkaista logiikkaa.

## 8.1. Structured Text (ST) ominaisuudet

IEC-kielistä ST-kieli vastaa ehkä parhaiten PLC-ohjelmoinnin kasvavaa monimutkaisuutta. Toimintolohkot, laskutoimitukset ja data-analyysi voidaan toteuttaa paljon helpommin kuin Ladder- tai IL-kielellä. Päätöksentekosilmukat ja osoittimet (muuttujat, joita käytetään epäsuoraan osoitteistokseen) mahdollistavat kompaktimman ohjelmatoteutuksen kuin Ladder-kielellä. Tämä helpottaa monimutkaisen ohjelman jäsentämistä.

ST-kieli on pääohjelmointikieli kaikissa Fidelix FX-sarjan PLC:ssä ja sitä käytetään jokaisessa BMS-projektissa. Projekti-insinöörin tehtävänä on luoda selkeä ja virheetön ST-koodi, joka toimii PLC:ssä projektin toimintojakson SoO (Sequence of Operations) mukaisesti. Yksi eniten käytetyistä ST-standardiohjelmalohkoista on kolmannen osapuolen Modbus-laitteen integrointi. ST-koodilohko, jossa on tunnetut toiminnalliset osat, on välttämätön Modbus-laitteen kanssa kommunikoimiseksi.

Jokainen ST-ohjelmakoodi (toiminto, toimintolohko, ohjelma) on muodostettava seuraavien perussääntöjen mukaisesti:

1. Oikea alkutekstilohko, jossa on alustaminen, jossa luetellaan kaikkien tyyppien ja käyttötarkoitusten kaikki muuttujat, muiden funktiolohkojen nimen.

```
PROGRAM EXAMPLE_PRG
VAR
iResult   :INT;
fbMB      : EXAMPLE_FB;

END_VAR
```

2. Pääkoodiosa, jossa on:
    a. muuttujat ja arvot laskentaoperaattoreineen
    b. loogiset operaattorit, ehdolliset operaattorit, sykliset ja aikaoperaattorit.
    c. funktioiden ja funktiolohkojen kutsut oikeine parametreineen.
    d. ohjainkohtaiset operaattorit.
    e. Ohjelmakoodin on oltava yhteensopiva ST-kielen syntaksin kanssa, ja sen on läpäistävä ohjelmointiympäristösovelluksen (Infoteam OpenPCS) tarkastus.
    f. Ohjelman parametrien sarja, kuten PLC-tyyppi, ohjelman sykli, symbolikoodisivu jne. Parametrit ovat ohjainkohtaisia ja niitä tarvitaan kääntäjälle.
    g.

```
fbMB(
iModuleAddress        := 1,
iPort := 6,
coil_11 := '',        (*** COIL VALUE ***)
discrete_22 := '',    (*** DISCRETE VALUE ***)
holding_333 := '',    (*** HOLDING VALUE ***)
inputreg_444 := '',   (*** INPUT VALUE ***)
id_ModuleAlarm := '');
```

3. Oikea loppuosa ja lopputeksti.

```
END_PROGRAM
```

## 8.2. Rajapinnan rakenne

Rajapinnan vuorovaikutus sisältää seuraavat elementit:

- Pääikkuna, jossa on perustoiminnot, kuten sovelluksen sulkeminen, tiedostotoiminnot, informaatioikkuna.
- Käyttöliittymä (alaikkuna) uusille tai olemassa oleville "rekisterikonfiguraatioiden" käyttäjätiedoille.
- Käyttöliittymä (pääikkunan osa) tallennettujen tietokokonaisuuksien ja aktiivisten "rekisterikonfiguraatioiden" luettelointia varten.

# 9. Toteuttamisen vaihe

Kehitys tehdään e(fx)clipse SDK 4.6.0:lla, jossa on integroitu tuki graafisen käyttöliittymän kehitystyökalulle SceneBuilder:lle. Graafinen käyttöliittymä luodaan ja editoidaan Gluon SceneBuilder 8.5.0 sovelluksessa.

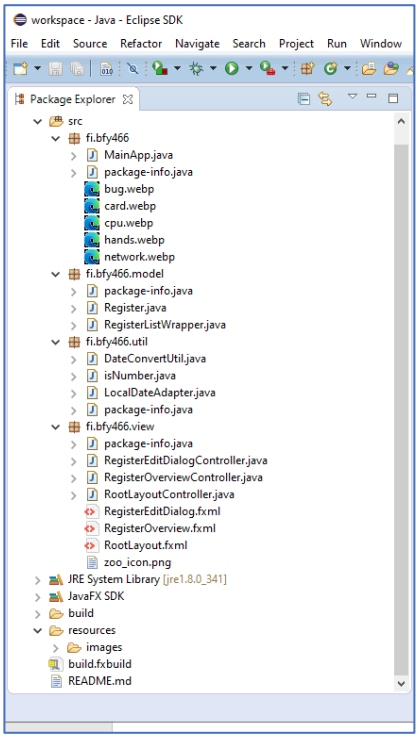
## 9.1. Java paketit ja luokat

Projekti sisältää 3 pääpakettia ja 1 apupaketin:
**fi.bfy466** sisältää perusluokan (*MainApp.java*), jossa tiedostojen ja JavaFX-rajapinnan vuorovaikutus tapahtuu.
MainApp.java-luokassa toteutetaan seuraavat toiminnot:
- Tiedostojen luominen, avaaminen ja tallentaminen. Windowsin tiedostojärjestelmän vuorovaikutus.
- JavaFX-toiminnon (Scene) luominen, esittäminen ja sulkeminen. JavaFX-kirjaston geneerisen ra-rajapinnan vuorovaikutus.
- Objektien ja niiden luetteloiden käsittely, tekstitiedostojen käsittely.

**fi.bfy466.model** pakkauksessa on luokkia, jossa ”register” objekti ja niiden kokoelmat on toteutettu. Luokassa on toteutettu vuorovaikutukset käyttöliittymän ja käyttöjärjestelmän
**fi.bfy466.model** luokat sisältävät objektien perusluomisen, ominaisuudet ja listat, joita niistä muokataan. "Register" on ydinolio, joka muodostaa "Register List", kokoelman, jota käytetään sovelluksen ulostulossa (IEC 61131-3 kutsuohjelma + funktionblokki).

**fi.bfy466.util** pakkauksessa on luokkia, jotka ovat aputoimintoja.

**fi.bfy466.view**. On luokkia, joissa graafiset ja käyttöliittymät on toteutettu. Nämä luokat ovat vuorovaikutuksessa JavaFX-kirjaston elementtien kanssa.




Kuva 18. Eclipse projektin rakenne.

## 9.2. Sovelluksen logiikka

MVC-ohjelmointimallin mukaan on tarpeen erottaa sovelluslogiikasta vastaavat JavaFX-luokat käyttöliittymän vuorovaikutuksesta vastaavista luokista. Sovelluslogiikka sisältää seuraavat osat:

1. Käyttäjän syöttämien tietojen kerääminen ja rekisterikonfiguraatio "register configuration" -objektien muodostaminen.
2. Syötettyjen tietojen yhdenmukaisuuden tarkistaminen
3. "Rekisterit"-tietokokonaisuuden muodostaminen "Rekisterin kokoonpanon" kokoelmasta.
4. Tietokokonaisuuden tallentaminen XML-muodossa tulevaa käyttöä varten.
5. Tietokokonaisuuden tallentaminen IEC-muodossa toimintolohkona ja kutsuvana ohjelmana.

Rajapinnan vuorovaikutus sisältää seuraavat osat:

1. Pääikkuna, jossa on perustoiminnot, kuten sovelluksen sulkeminen, tiedostotoiminnot, informaatioikkuna.

2. Käyttöliittymä (alaikkuna) uusia tai olemassa olevien "rekisterikokoonpanon" käyttäjätietoja varten.
3. Käyttöliittymä (pääikkunan osa) tallennettujen tietojen ja aktiivisten "rekisterikonfiguraatioiden" luettelointia varten.

MVC-mallissa erotetaan periaatteessa tietojen, ohjauksen ja visualisoinnin vastuut erillisiin objekteihin. Näiden objektien tulisi toimia yhdessä käyttäjän syötteen käsittelemiseksi, tietojen tallentamiseksi ja niiden visualisoimiseksi näytölle.

## 9.3. Sovelluksen osiin jakaminen

Jotta voimme luoda erilliset sovellusosat näkymälle ja logiikalle, meidän on luotava erilliset luokat objekteille ja niiden tietojenkäsittelylle sekä erilliset luokat käyttäjältä kerättyjen objektitietojen näkymärajapinnalle ja väliaikaiseen muistiin tallennettujen objektitietojen näyttämiselle. Näiden luokkien hallitsemiseksi yhdessä meidän on luotava "Controller"-luokka(t), jolla on yhteydet sovelluksen objektien ominaisuuksien ja JavaFX:n graafisten objektien ja rajapintojen välillä.

"Controller" on tulevassa kehityksessä "välikappale" sovelluksen osan, joka saattaa olla muuttumaton eri ympäristöissä, ja eri graafisten käyttöliittymien välillä eri renderöijissä ja selaimissa. Tällaisen järjestelmän käyttö antaa kehittäjälle joustavuutta kehysten ja graafisten kirjastojen valinnassa. Tällaisen "modulaarisen" projektirakenteen avulla koko projekti voitaisiin siirtää toiseen kehykseen helpommin suhteellisen pienillä koodimuutoksilla.

Tässä työssä kehitämme pienen ja hyödyllisen sovelluksen, jonka graafinen käyttöliittymä perustuu JavaFX-kirjastoihin.

### 9.3.1. (Model part) Malliosa

Sovelluksessa käytettävä pääobjekti on yksi Modbus-rekisteri, jolla on useita parametreja ja ominaisuuksia. "Register"-objektin parametrien määrä on rajoitettu ja niillä on omat arvoalueensa. Näin varmistetaan mahdollisuus tarkistaa käyttäjältä saadut oikeat arvot ja asettaa joitakin parametreja oletusarvoiksi syöttöprosessin helpottamiseksi. Erityyppisiä "Register"-objekteja käsitellään eri tavoin, ja ne käyttävät eri elementtejä IEC-

toimintolohkossa ja ohjelmassa. Tämä objekti on osa tietokokonaisuutta (kokoelmaa), joka sisältää yhden tai useamman erityyppisen "rekisteri"-objektin. Tätä varten luodaan "Register.java"-luokka, jossa on "Modbus register" -objektin parametrit, sen ominaisuudet ja standardimenetelmät.

Useiden "rekisteri"-objektien täydellisen tietokokonaisuuden määrittelemiseksi ja tallentamiseksi on luotava kokoelma "rekisteri"-objekteja ja määriteltävä luettelon vakiomenetelmät tietojen syöttöä, käsittelyä ja tulostusta varten. Tätä luetteloa käytetään sen näyttämiseen käyttöliittymässä ja tallentamiseen XML-muodossa myöhempää käyttöä varten. Samaa objektiluettelo-luokkaa käytetään luotaessa tuloksena IEC-ohjelmaa, jossa on kaikki "rekisteri"-objektit samassa IEC-funktiolohkossa ja kutsuohjelmassa.

"RegisterListWrapper.java"-luokka sisältää JavaFX-merkinnät ja vakiomenetelmät, joilla palautetaan luettelo nykyisistä määritellyistä objekteista. Luokkaa käytetään objektiluettelon hakemiseen näyttöä varten ja sen muuntamiseen XML-tiedostoon ja tuloksena olevaan IEC-ohjelmaan.

### 9.3.2. (View part) näkymän osa

JavaFX-grafiikkakirjasto käyttää erityistä merkintäformaattia FXML, jonka rakenne muistuttaa XML-tiedostoja ja jossa on "vanhempi ja lapsi" -arkkitehtuuri. FXML-merkintätunnisteet liittyvät JavaFX-kirjaston elementteihin ja niiden ominaisuuksiin. Elementtityypit tuodaan FXML-tiedostoon, jotta voidaan viitata vastaaviin ominaisuuksiin. Tällainen standardointi helpottaa yhteensopivuutta JavaFX-kirjastojen muiden versioiden kanssa. Graafisten elementtien vakioidut ominaisuudet mahdollistavat myös erilaisten tyylien soveltamisen lähes kaikkiin käyttöliittymän elementteihin.

FXML-tiedostossa käyttöliittymäikkunat ja -elementit määritellään puurakenteisen merkintäjärjestelmän avulla, joka sisältää elementin tyypin, sijainnin, tekstin, muuttujat ja vuorovaikutusominaisuudet. Itse asiassa on mahdollista luoda käyttöliittymäikkunoita tai -lomakkeita ilman graafista editoria, koska kaikki määritykset voidaan tehdä FXML-tiedostossa. Tehokas ja nopea prototyyppien luominen ja mahdollisuus muuttaa samankaltaisia ominaisuuksia koko tiedostossa lisäävät JavaFX-pohjaisten käyttöliittymien arvoa.

Merkintätunnisteet sisältävät sijainti- ja asettelutunnisteet, ohjainluokan ja muuttujien ominaisuudet, teksti-, tyyli- ja käyttäjäohjausominaisuudet. Jokainen ominaisuus kuuluu vastaavaan JavaFX-elementtiin.

JavaFX-käyttöliittymien luomiseen voidaan käyttää erityistä GUI-kehitysympäristöä. Gluon SccneBuilder GUI-suunnitteluohjelmisto vakiokirjastoineen on yleisimmin käytetty JavaFX-käyttöliittymien suunnitteluun. Tapauksessamme käytämme Eclipse JavaFX SDK:n integroitua SceneBuilder versiota 8.5.0. Nopea vaihtaminen FXML-tiedostojen ja niiden graafisen tulkinnan välillä helpottaa koko kehitysprosessia.

Jokainen interaktiivinen ikkuna tai lomake JavaFX:ssa linkitetään vastaavaan Java-näkymäluokkaan (Controller) ja sen alias FXML-objektiin tässä luokassa. Kaikki Java-luokan FXML-olion ominaisuudet yhdistetään näin vastaavan graafisen elementin ominaisuuksiin. Tällainen kiinteä yhteys varmistaa, että Java-tietojen arvojen näyttämisessä ei tapahdu virheitä objekteista. Sama päinvastoin: käyttäjän JavaFX-lomakkeella syöttämästä tiedosta tulee Java-luokan objektin ominaisuus. Kuten edellä todettiin, välittäjänä toimivan "Controller"-luokan käyttö "Model"- ja "View"-luokkien välillä takaa joustavan kehityksen, kun projektissa käytetään erilaisia ja monimutkaisia tietoja.

MVC-ohjelmointimallin lisäetuna näemme, että sovelluslogiikka voidaan tehdä erillään graafisista objekteista ja niiden ominaisuuksista. "Malli"-luokat, mukaan lukien MainApp.java-luokka, voivat toimia alkeisobjektien ja kokoelmien kanssa koskematta niiden visualisointiosaan. Jos sovelluslogiikkaluokkiin tai graafiseen suunnitteluun tehdään massiivisia muutoksia, ne vaikuttavat vain osaan luokista, mikä lisää joustavuutta ja helpottaa refaktoroinnin kaltaisia prosesseja.

### 9.3.3. (Auxiliary part) Apuosa

Projektissamme käytämme joskus maan (ja käyttöjärjestelmän) määrittämiä muuttujia, kuten päivämäärämuotoa ja aikaleimoja. Näitä tietomuotoja on helpompi käsitellä erityisten apuluokkien avulla. Esimerkiksi DateConvertUtil.java-luokka voi jäsentää

merkkijonomuuttujan paikallisesti määriteltyyn päivämäärämuotoon ja palauttaa vastaavan objektin. Tätä objektia voidaan käyttää XML-tiedostossa tai muussa luokassa.

Projektissa on useita kenttiä, joihin syötetään käyttöliittymän tietoja. Näiden syötteiden testaamiseksi ja koodin yksinkertaistamiseksi saatetaan joutua luopumaan monimutkaisista säännöllisistä lausekkeista, kun syötetietoja tarkistetaan kirjoitusvirheiden ja johdonmukaisuuden varalta. Luokat, joissa on tällaisia toimintoja, kuten isNumber.java, voisivat olla hyödyllisiä projektimme "näkymä"-osassa. Näin käyttöliittymään syötetyt tiedot toimitetaan testattavaan sovelluslogiikkaan.

### 9.3.4. (Application logic) Sovelluslogiikka

MainApp.java luokka sisältää suurimman osan projektin sovelluslogiikasta ja tärkeimmät syöttö- ja tulostoiminnot. Siinä on toteutettu tiedosto-operaatiot, XML:n ja tuloksena syntyvän IEC-ohjelman sisällön luominen, kommunikointi näkymäohjainluokkien kanssa ja joitakin testausominaisuuksia. MainApp.java luo tärkeimmän JavaFX GUI -ikkunan (primaryStage) ja kommunikoi sitten vastaavien (view controller) näkymäohjainluokkien kanssa luoden ali-ikkunoita ja lomakkeita tietoja varten. Tietojen hallinta siirretään edelleen vastaaville näkymänohjainluokille.

MainApp.java kommunikoi käyttöjärjestelmän resurssien kanssa luodakseen, tallentaakseen ja avatakseen tiedostoja, sekä XML- että IEC-ohjelmia. Tämä tapahtuu järjestelmän määrittelemien sulautettujen Java-funktioiden avulla.

MainApp.java luokan olennainen tehtävä on luoda IEC-ohjelman tekstit syötettyjen "rekisterikokoelman" tietojen mukaan. Vastaavat metodit, joihin GUI-objektin ("Create and Save IEC as...") kutsuu, hakevat nykyisen objektiluettelon, luovat tyhjän tiedoston ja kirjoittavat tekstirivejä kunkin luettelon jäsenen ominaisuuksien mukaisesti. Tuloksena syntyneet tiedostot (IEC-ohjelma + toimintolohko) tallennetaan kiintolevylle järjestelmätiedoston valintaikkunoiden avulla.

## 9.4. (User interface) Käyttöliittymä

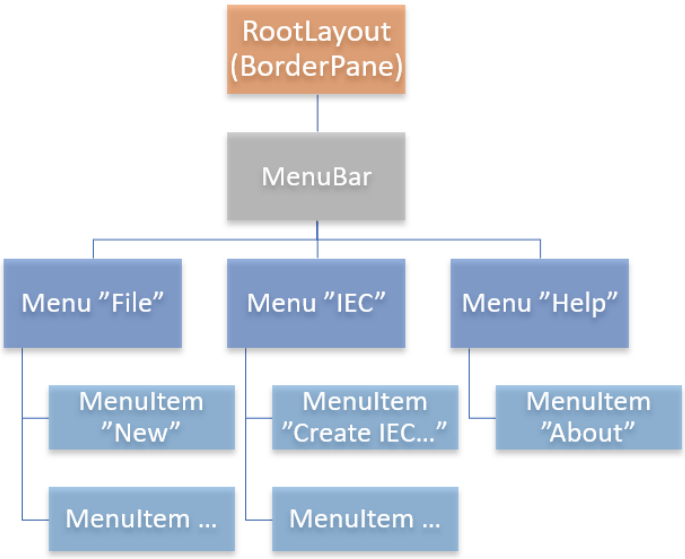
Graafinen käyttöliittymä (Stage) koostuu kolmesta pääosasta: Pääikkuna, sen sisältö ja erillinen ikkuna rekisterien lisäämistä ja muokkaamista varten. Stage on pääkontti, joka on yleensä ikkuna, jossa on kehys ja tyypilliset minimointi-, maksimointi- ja sulkupainikkeet. Stage:n sisälle lisätään Scene, joka voidaan tietysti vaihtaa toiseen Scene:en. Scene:n sisälle lisätään varsinaiset JavaFX-solmut, kuten AnchorPane, TextBox jne.

### 9.4.1. ”RootLayout” ikkunan rakenne

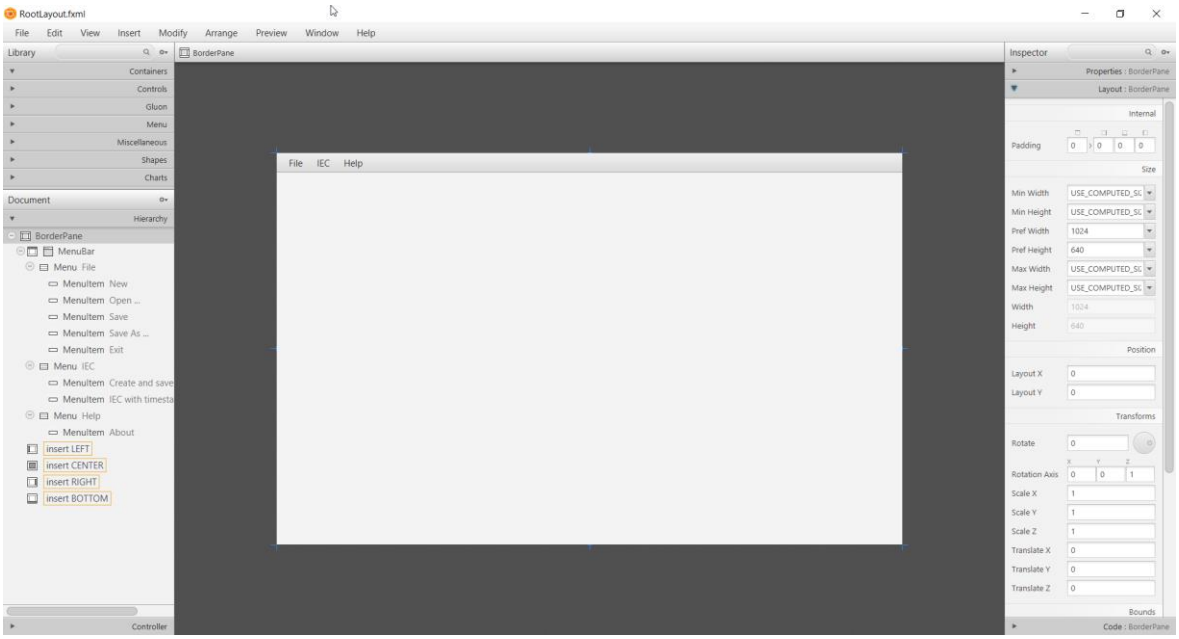
Juurisolmu (RootLayout) sisältää kaikki sovelluksen ja tiedostojen hallinnan peruskomennot, tavalliset ikkunapainikkeet ja otsikon. GUI:n (GUI, Graphical user interface) sisäinen rakenne on kuvassa 19.

Yksittäisiä JavaFX:n kohtausgrafiikassa olevia kohteita kutsutaan solmuiksi (node). Kukin solmu luokitellaan joko haarasolmuksi (eli sillä voi olla lapsia) tai lehtisolmuksi (eli sillä ei voi olla lapsia). Puun ensimmäistä solmua kutsutaan aina juurisolmuksi, eikä sillä ole koskaan vanhempaa.

Juurisolmu on MenuBar-haarasolmun vanhempi, ja MenuBar-solmu on kolmen haarasolmun "File", "IEC", Help" vanhempi, jotka ovat vastaavien valikkokohteita sisältävien lehtisolmujen vanhempia ("New", "Open" ... "Create IEC...", "About" jne.).



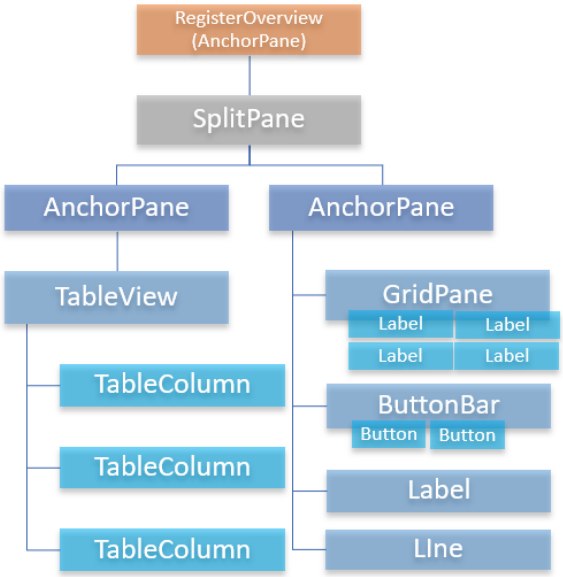

Kuva 19. RootLayout Scene:n solmustruktuuri



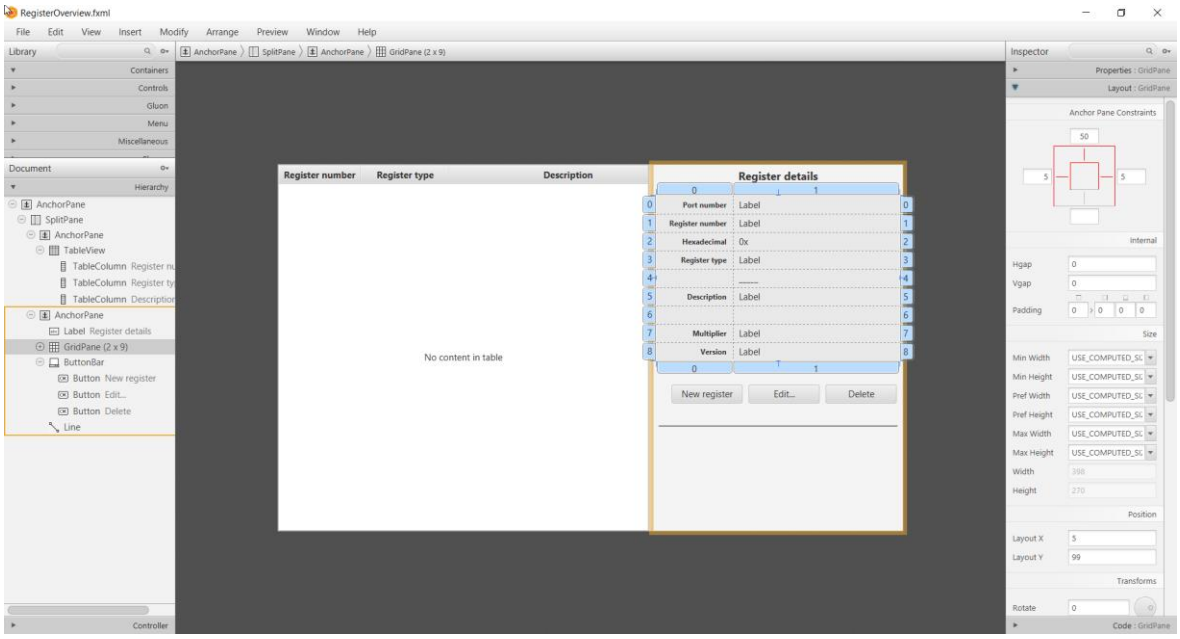
Kuva 20. RootLayout graafisen struktuurin editointi SceneBuilder sovelluksessa.

### 9.4.2. ”RegisterOverview” paneelin rakenne

Pääikkunaan sijoitetaan kaksi SplitPane-solmun alla olevaa ikkunaruutua: vasen AnchorPane, jossa on TableView, joka näyttää nykyiseen joukkoon (registerList) sisältyvät "Register"-tiedot, ja oikea AnchorPane, jossa näkyvät valitut "Register"-tiedot ja Add/edit-valintaikkunan pääohjauspainikkeet. Apuelementit, kuten tekstilaput ja viivat, sijoitetaan oikeaan paneeliin.




Kuva 21. RegisterOverview Scene:n solmu struktuuri




Kuva 22. RegisterOverview Scene:n graafisen struktuurin editointi SceneBuilder sovelluksessa.

RegisterOverview-kohtaan on sijoitettu taulukkonäkymä, jossa näkyvät nykyisen tietokokonaisuuden "Rekisteriluettelo" sisältämät "rekisteri"-objektin tiedot. Tämä tietokokonaisuus näytetään missä tahansa seuraavista tilanteista:

- On luotu onnistuneesti vähintään yksi "rekisteri"-objekti.
- On olemassa onnistuneesti avattu XML-tiedosto, joka sisältää kaikki "rekisteriluettelon" tiedot.

Molemmissa tapauksissa näytetään perustiedot "rekistereistä".

Kun jokin taulukon olemassa olevista riveistä on valittu (napsauttamalla tai siirtämällä fokus), vastaavat tiedot "rekisteristä" näkyvät kokonaisuudessaan oikean ruudun "GridPane"-elementissä. Tämä mahdollistaa toimintojen suorittamisen tällä hetkellä valitulla "rekisteri"-objektilla painikkeilla.

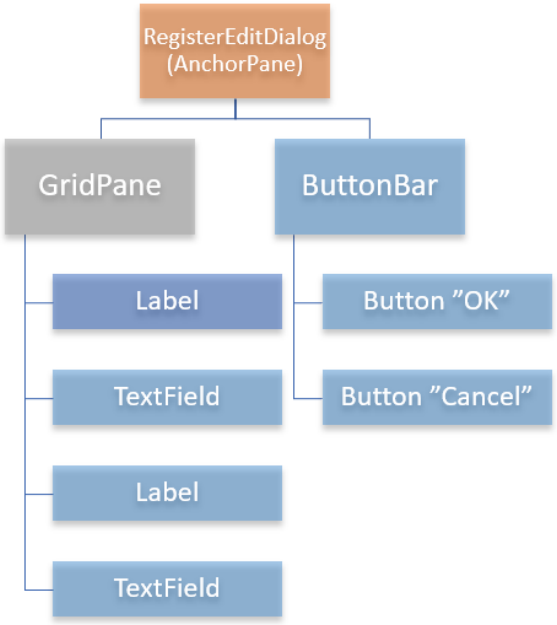
Oikeassa SplitPane-elementissä on painikepalkki, jossa on tärkeimmät "rekisteri"-objektin toiminnot, kuten uuden luominen, muokkaaminen ja poistaminen. "Uusi" ja "Muokkaa" -painikkeet kutsuvat esiin erillisen ikkunan, jossa on tietojen syöttökentät. "Poista"-painike poistaa vasemmassa paneelissa valitun ja oikeassa paneelissa näkyvän "rekisteri"-objektin. Jos rekisteriä ei ole valittu (tai taulukossa ei ole enää rekistereitä), painike aktivoi ponnahdusikkunan. Sama hälytysviesti aktivoituu, kun käyttäjä yrittää painaa "Muokkaa", kun rekisteriobjektia ei ole valittu.

Jotta käyttäjä näkee Modbus-standardin mukaisen rekisterinumeron (osoitteen), on olemassa Label-objekti, jossa on laskettu syöttämänsä aloitusrekisterin osoitteen heksadesimaaliarvo. Tätä aloitusrekisterin muotoa käytetään usein eri valmistajien Modbus-laitteiden dokumentaatiossa.

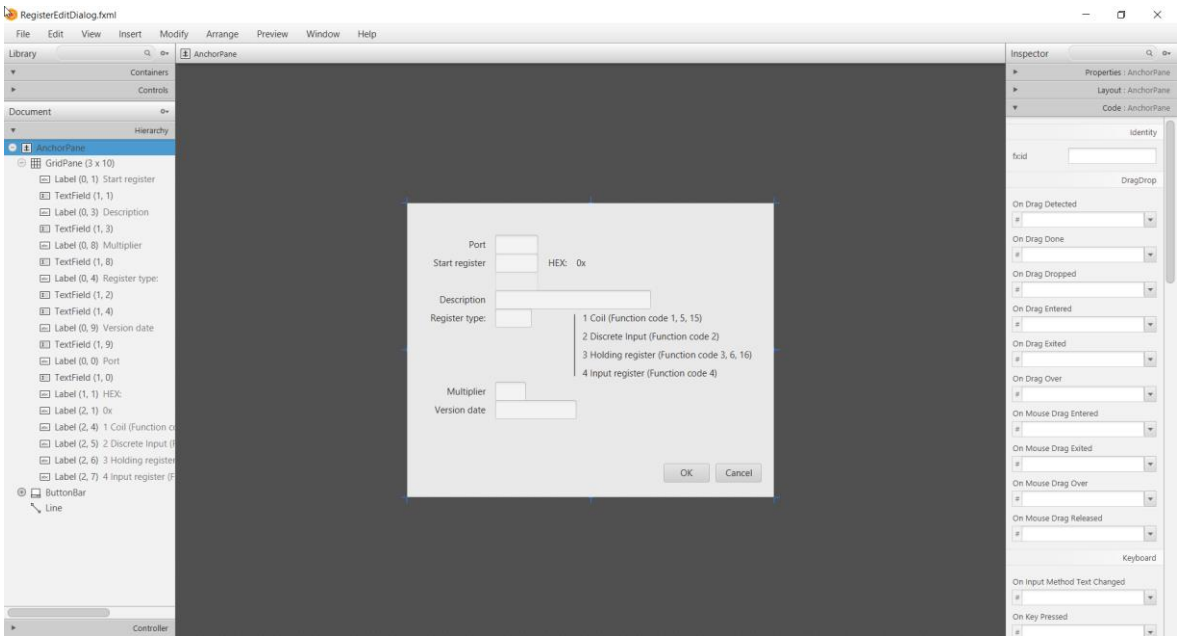
### 9.4.3. ”EditRegisterDialog” ikkunan rakenne

"Uusi rekisteri"-painike avaa erillisen valintaikkunan "EditRegisterDialog", jossa käyttäjä voi syöttää kaikki tarvittavat "rekisteri"-objektin parametrit. Tämä ikkuna luodaan esiasetetuilla oletusparametreilla jokaiselle objektin ominaisuudelle. Tekstikenttäsyöttöjä käytetään käyttäjän syöttämiä tietoja varten. Jokaiselle TextField-kentälle on vastaava arvo, joka asetetaan nykyisen "register"-objektin ominaisuuteen.

Klikkaamalla "Ok", kaikki "register"-objektin ominaisuudet, mukaan lukien oletusarvo (jos käyttäjä ei ole muuttanut arvoa), tallennetaan juuri luotuun "register"-ominaisuuteen. Tämän jälkeen "EdirRegisterDialog"-ikkuna sulkeutuu. Tämän jälkeen "register"-objekti näkyy RegisterOvervew-ikkunan vasemmanpuoleisessa ruudussa nykyisten "rekisterien" luettelossa.




Kuva 23. RegisterEditDialog Scene:n solmu struktuuri




Kuva 24. RegisterEditDialog ikkunan graafisen struktuurin editointi SceneBuilder sovelluksessa.

# 10.Pohdinta

Tässä opinnäytetyössä on tarkasteltu sekä teollisuuden että rakennusten automaatiojärjestelmien perusperiaatteita. Näitä automaatiojärjestelmiä on tutkittu sekä arkkitehtuurin että teknisen komponentin näkökulmasta. On määritelty automaatiojärjestelmän tiedon muuntamisen eri tasot, siihen liittyvien laitteiden tyypit ja tarkoitukset; käyttöominaisuudet ja esimerkkejä sovelluksista.

Erityistä huomiota kiinnitettiin menetelmiin ja tekniikoihin, joilla tietoja siirretään automaatiojärjestelmän eri osien ja tasojen välillä tehokasta hallintaa varten. Rakennusautomaatiojärjestelmä valittiin, koska sen rakenne ja toiminta ovat kirjoittajalle tuttuja hänen työstään ja koska rakennusautomaatio koostuu monista heterogeenisistä elementeistä ja sisältää erilaisia protokollia, portteja ja tiedonsiirtoasetuksia laitteiden välillä. Tällaista monimuotoisuutta ei yleensä löydy teollisuusautomaatiojärjestelmistä.

Eri protokollia ja tiedonsiirtomenetelmiä tarkasteltaessa on kiinnitetty erityistä huomiota niiden vakioarkkitehtuuriin, teknisiin mahdollisuuksiin ja rajoituksiin. Soveltamisaloja eri ympäristöissä ja jommankumman protokollan valinnan toteutettavuutta on arvioitu. Ainoastaan yleisimmät standardiprotokollat, joista on helppo löytää tietoa kirjallisuudesta ja Internetistä, otettiin huomioon. Näiden protokollien käyttöä tarkasteltiin yhteensopivuuden kannalta rakennusautomaatiojärjestelmän erilaisien kerrosten kanssa. Päätelmänä todettiin, että Modbus olisi käytettävä rakennusautomaatiojärjestelmien nykyisiin tehtäviin, mutta korostettiin, että on ilmeistä, että uudempi protokolla, kuten BACnet, syrjäyttää ja korvaa aikanaan vanhemmat Modbus-, KNX- ja LonTalk-protokollat.

Tämän opinnäytetyön päätavoitteena oli tutkia mahdollisuutta kehittää tietokonesovellusta, jolla voitaisiin luoda rakenteinen rekisteriluettelo ja ohjelmateksti IEC61131.3-kielellä kaksisuuntaista tiedonsiirtoa varten ohjelmoitavan ohjaimen ja Modbus-laitteen välillä. Tuloksena syntyvää rekisteriluetteloa ja ohjelmatekstiä voivat käyttää projektisuunnittelijat toteuttaessaan rakennusautomaatioprojekteja Fidelix Oy:n laitteilla. Tällainen ominaisuus helpottaisi huomattavasti insinöörin työtä ja säästäisi työaikaa ja kustannuksia.

Ohjelmoinnin suunnitteluun valittiin Haaga-Helian kursseilla aiemmin opiskellut ohjelmointimenetelmät, -tekniikat ja -tekniikat, kuten ALM-kehityssyklimalli, MVC-sovellusrakentamismalli, Java-ohjelmointikieli, JavaFX graafinen rajapintakirjasto ja Eclipse kehitysympäristö. Kaikkia näitä elementtejä on aiemmin tutkittu erikseen, ja tässä asiakirjassa kirjoittaja on pyrkinyt yhdistämään niiden hyödylliset puolet halutun tuloksen saavuttamiseksi.

Johtopäätökset tehtiin ALM-mallin hyödyllisyydestä jatkuvana prosessina, jossa luodaan, testataan ja parannetaan mitä tahansa ohjelmistotuotetta. Valittu malli on tehokas tässä pienessä hankkeessa, ja sitä käytetään todennäköisesti ohjelmistotuotteen parantamiseen ja monimutkaistamiseen lisäämällä ja testaamalla jatkuvasti uusia ominaisuuksia ja ohjelmistomoduuleja.
Tutkittiin MVC-mallin yleisiä periaatteita ja etuja, sillä sen avulla voidaan luoda pieni joustava sovellus lyhyessä ajassa. Periaate, jonka mukaan ohjelmistotuotteen eri osien toiminnot erotetaan toisistaan, mahdollistaa ohjelmistotuotteen vaiheittaisen rakentamisen.

Java-kielen ja JavaFX-kirjaston valinta perustui aiempaan tutustumiseen ja opiskeluun Haaga-Helian kurssilla sekä siihen, että Java-kielellä kirjoitettua ohjelmaa voidaan käyttää vähäisin muutoksin muissa käyttöjärjestelmissä ja eri arkkitehtuurin omaavissa tietokoneissa. Koska ohjelmaa saatetaan tulevaisuudessa joutua käyttämään Linux-käyttöjärjestelmässä, on perusteltua valita yleiskieli.

Kirjoittaja oli myös aiemmin opiskellut Eclipse-ohjelmointiympäristöä kursseilla ja omatoimisesti, joten hänen valintansa oli enemmän tai vähemmän ennalta määrätty. Pienen sovelluksen kehittämisen aikana se osoittautui varsin joustavaksi ja tehokkaaksi. Sisäänrakennettu tuki JavaFX-kirjastojen kanssa työskentelylle lisää Eclipse ohjelmointiympäristöön etuja. Työn aikana kirjoittaja totesi, että tämä ympäristö soveltuu pieniin hankkeisiin, joissa on vain vähän vaatimuksia yhteistoiminnalliselle kehitykselle muiden ohjelmoijien kanssa, koska tietojen jakamiseen tarkoitetut välineet ovat melko vaikeita ottaa käyttöön.

Yhteenvetona opinnäytetyön tuloksista voidaan todeta, että opinnäytetyön tavoitteet on saavutettu onnistuneesti, ja tuloksia (pöytätietokoneen sovellus) voidaan hyödyntää todellisissa rakennusautomaatioprojekteissa. Ohjelmalla luodut rekisteriluettelot ja ohjelmatekstit ovat varsin hyödyllisiä kommunikoitaessa eri Modbus-laitteiden kanssa. Tämän opinnäytetyön jatkokehitys voisi olla vastaavien toimintojen siirtäminen "asiakaspalvelin"-arkkitehtuuriin perustuvaan Internet-sovellukseen sekä kenttälaitteiden yhteyden suorituskyvyn testaaminen lisäämällä Modbus-porttitoiminto USB-sarjaportin muuntimen kautta. Lisäominaisuutena olisi luultavasti lisätä tuki samanlaisille strukturoiduille rekisteriluetteloille muille viestintäprotokollille, kuten BACnet:lle, KNX:lle ja muille. Tulevaisuudessa tätä sovellusta voidaan parantaa ja laajentaa käyttämällä ALM-menetelmiä ja MVC-arkkitehtuurin ominaisuuksia.

# 11. Lähteet


Chamorro-Atalaya, O. 2020, Lighting control network based on KNX protocol, for the reduction of energy consumption, Indonesian Journal of Electrical Engineering and Computer Science. Saatavilla: https://www.researchgate.net/publication/344027725_Lighting_control_network_based_on_KNX_protocol_for_the_reduction_of_energy_consumption (Haettu 26.10.2022)

Dutertre B. 2003, Formal modeling and analysis of the Modbus protocol. Saatavilla: http://www.csl.sri.com/~bruno/publis/formal_modbus.pdf (Haettu 10.5.2020)

Echelon. 1994, Saatavilla: http://stitcs.com/en/LonWorks/Lontalk%20Protocol%20Spec.pdf (Haettu 27.10.2022)

Eden-Rump E. 2021, How to apply MVC in JavaFX. Saatavilla: https://edencoding.com/mvc-in-javafx (Haettu 16.10.2021)

Evans, B., Flanagan D. 2015, Java in a Nutshell, 6. p., O'Reilly Media, Sebastopol, CA

Gupta, A.K.. Arora, S.K. Westcott, J.R. 2013, Industrial automation and robotics. 1 p. Dulles. Mercury Learning and Information 2017. 577 s. ISBN: 978-1-938549-30-4

Härkönen, P. Mikkola, J. Piikkilä, V. Sahala, A. Sahlstén, T. Sandström, B. Sirviö, A. Spangar, T. Sulku, J. 2012. Rakennusautomaatiojärjestelmät. 3 p. Espoo. Sähkötieto ry. 2012. 286 s. ISBN: 978-952-231-071-2

Javapoint 2020, MVC Architecture in Java. Saatavilla: https://www.javatpoint.com/mvc-architecture-in-java (Haettu 10.5.2020)

Keinänen, T. Kärkkäinen, P. Lähetkangas, M. Sumujärvi, M. 2007. Automaatio-järjestelmien logiikat ja ohjaustekniikat. 1 p. Porvoo. WSOY Oppimateriaalit Oy. 2007. 306 s. ISBN 978-951-0-32423-3

KNX Association. 2013, System-architecture Saatavilla: https://my.knx.org/en/shop/knx-specifications?product_type=knx-specifications (Haettu 27.10.2022)

Modbus Organization. 2012, Modbus Application Protocol V1.1b3, Saatavilla: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf (Haettu 12.10.2022)

Niazi, M.A., 2020, What is the BACnet Protocol and How is it Used in Building Automation Systems to Control Data Exchange? Saatavilla: https://control.com/technical-articles/what-is-the-bacnet-protocol/ (Haettu 27.10.2022)

Oracle 2014, JavaFX: Working with the JavaFX Scene Graph. Saatavilla: https://docs.oracle.com/javase/8/javafx/scene-graph-tutorial/index.html (Haettu 10.5.2020)

PractiTest 2020, Application Lifecycle Management (ALM), Saatavilla: https://www.practitest.com/application-lifecycle-management/ (Haettu 9.5.2020)

Produal 2022, HDH - Room CO2 transmitter/controller. Saatavilla: https://www.produal.com/sku-1135100.html (Haettu: 10.5.2020)

Surina, E.A., 2007, Вестник МГСУ Том 2 Выпуск 3, 2007 Актуальные вопросы и оценка эффективности инвестиционных проектов автоматизации зданий (Rakennusautomaation investointihankkeiden ajankohtaiset kysymykset ja arviointi). Saatavilla: http://vestnikmgsu.ru/ru/component/sjarchive (Haettu 16.10.2021)

Thayer T. 2009, Speaking in Tongues: Understanding the IEC 61131-3 Programming Languages. Saatavilla: https://www.controleng.com/articles/speaking-in-tongues-understanding-the-iec-61131-3-programming-languages (Haettu 16.10.2021)

Vats R. 2020, Java MVC Project [Step-By-Step Process Explained]. Saatavilla: https://www.upgrad.com/blog/java-mvc-project (Haettu: 20.5.2021)