HAMK
Hämeen ammatti-
korkeakoulu

Datan kerääminen, tallennus ja visualisointi valmistavassa teollisuudessa

Ammattikorkeakoulututkinnon opinnäytetyö
Sähkö- ja automaatiotekniikka, insinööri (AMK)
Kevät 2025
Antti Harjajärvi

HAMK
Hämeen ammatti-
korkeakoulu

Sähkö- ja automaatiotekniikka, insinööri (AMK) Tiivistelmä
Tekijä Antti Harjajärvi Vuosi 2025
Työn nimi Datan kerääminen, tallennus ja visualisointi valmistavassa teollisuudessa
Ohjaaja Juha Sarkula

Opinnäytetyössä tutkittiin teollisuuden datankeruun, tallennuksen ja visualisoinnin mahdollisuuksia tuotantoprosessien tehostamiseksi, sekä kehitettiin ja toteutettiin pilottiratkaisu valitulle tuotantokoneelle. Työn toimeksiantajana toimi Pihla Group Oy, joka näkee digitalisaation tärkeänä osana tuotannon kehittämistä. Pihla Groupilla oli aikaisemmin käytössä eri toimittajien kehittämiä konseptitason ratkaisuja, mutta nämä eivät vastanneet yrityksen laajempia tarpeita.

Opinnäytetyön ensimmäisessä vaiheessa suoritettiin kartoitus, jossa selvitettiin tehtaan koneilta kerättävän datan tarpeet ja sen käyttökohteet tuotannon johtamisessa. Tämän pohjalta kehitettiin pilottitoteutus datankeruu-, tallennus- ja visualisointiratkaisulle, joka käytti MQTT-protokollaa tiedonsiirrossa, InfluxDB-tietokantaa tallennuksessa ja Grafana-ohjelmistoa visualisoinnissa. Toteutus keskittyi yhteen tuotannon koneeseen.

Tuloksena Pihla Group Oy sai käyttöönsä perustiedot ja pilottiratkaisun, jotka tukevat yritystä laajemman tiedonkeruujärjestelmän toteutuksessa. Kartoituksessa tunnistetut tarpeet ja käyttökohteet auttavat selkeyttämään tavoitteet ja odotukset myös tuleville toteuttajille. Työssä kehitettyä järjestelmää on myös mahdollista laajentaa tai käyttää muissa pilottihankkeissa.

Johtopäätöksissä korostui tarve jatkaa datankeruun laajentamista muihin tuotantolaitteisiin ja lisätä yhteistyötä eri toimittajien kanssa. Käytön laajentamisessa tunnistettiin mahdollisuuksia muun muassa tekoälyn ja konenäön hyödyntämiseen laadunvalvonnassa ja kunnossapidossa.

Avainsanat Automaatio, digitalisaatio, tiedonkeruu, IoT, visualisointi

Sivut 38 sivua ja liitteitä 5 sivua

HAMK
Hämeen ammatti-
korkeakoulu

DP	Electrical and Automation Engineering	Abstract
Author	Antti Harjajärvi	Year 2025
Subject	Data Collection, Storage and Visualization in Manufacturing Industry	
Supervisor	Juha Sarkula	

This thesis examines the possibilities of data collection, storage, and visualization in the manufacturing industry to enhance production processes. Additionally, a pilot solution was developed and implemented for a selected production machine.

Thesis was commissioned by Pihla Group Oy, who view digitalization as important element in the development of production. Pihla Group had previously used concept-level solutions developed by various suppliers, but these did not meet the broader needs of the company.

In the first phase of thesis, a survey was conducted to determine the data needs from factory machines and its applications in production management. Based on this, a pilot implementation was developed for a data collection, storage, and visualization solution, which used the MQTT protocol for data transmission, the InfluxDB database for storage, and the Grafana software for visualization. The implementation focused on one production machine.

As a result, Pihla Group Oy obtained basic information and a pilot solution that support the company in implementing a more extensive data collection system. The needs and applications identified in the survey help clarify goals and expectations for future implementers as well. The system developed in this work can also be expanded or used in other pilot projects.

The conclusion emphasizes the need to continue expanding data collection to other production equipment and to increase cooperation with various suppliers. In the expansion of use, opportunities were identified, such as utilizing artificial intelligence and machine vision for quality control and maintenance.

Keywords Automation, digitalization, data collection, IoT, visualization

Pages 38 pages and appendices 5 pages

Sisällys

Kuvat, taulukot ja kaavat

Liitteet

1 Johdanto

Digitalisaatio ja automaatio ohjaavat teollisuusympäristöjä kehittämään yhä tehokkaampia tuotantoprosesseja. Tämän myötä yritykset etsivät jatkuvasti uusia ratkaisuja datan keräämiseen, analysointiin ja hyödyntämiseen, jotta ne voisivat parantaa tuotannon tehokkuutta ja reagointikykyä muutoksiin. Tässä opinnäytetyössä perehdytään teollisuuden datankeruun, tallennuksen ja visualisoinnin mahdollisuuksiin, sekä kehitetään ja toteutetaan pilottiratkaisu valitulle tuotantokoneelle.

Työn tilaajana toimivassa Pihla Group Oy:ssä digitalisaatio ja automaatio nähdään tärkeässä roolissa tulevaisuudessa, minkä takia yritys haluaa panostaa näihin. Työssä käsiteltävät datankeruun ratkaisut eivät ole uusia yrityksessä, mutta aikaisemmin tehdyt ratkaisut rajoittuivat konseptitason ratkaisuihin, joita ei olla pystytty hyödyntämään halutussa laajuudessa. Tässä työssä pyritään tarjoamaan kattavampi ja yhtenäisempi lähestymistapa näiden ratkaisujen kehittämiseksi.

Työn ensimmäinen vaihe on suorittaa kattava kartoitus, jossa selvitetään tehtaan koneilta kerättävän datan tarpeita ja niiden käyttökohteita datalla johtamiseen. Kartoitus aloitetaan perehtymällä yrityksen tarpeisiin ja tutustumalla aiemmista datankeruujärjestelmistä saatuihin kokemuksiin.

Tästä lähtökohdasta työssä suunnitellaan ja toteutetaan datankeruu-, tallennus- ja visualisointiratkaisu yhdelle tuotantokoneelle. Tämä sisältää muun muassa viestirakenteiden ja järjestelmäarkkitehtuurin suunnittelun sekä niiden integroinnin osaksi olemassa olevaa tuotantoympäristöä. Toteutuksen valinta kohdistuu tuotantokoneeseen, jonka toimintaperiaate on mahdollisimman yleisesti käytetty Pihla Group Oy:ssä. Tällä varmistetaan, että kehitetty ratkaisu olisi helposti sovellettavissa mahdollisimman laajasti myös muihin koneisiin. Tämän toteutuksen avulla pyritään osoittamaan käytännössä, miten data-analytiikan parantaminen voi tehostaa tuotantoprosesseja. Työssä käytettävät työkalut, sovellukset ja järjestelmäarkkitehtuuri valikoitui aikaisemman kokemuksen ja soveltuvuuden takia.

Työn avulla Pihla Group Oy voi saada paremmat lähtötiedot laajemman datankeruuratkaisun suunnittelua ja toteutusta varten, joko sisäisten tai ulkoisten toimittajien avulla. Työ myös tarjoaa lukijalle yleiskuvan datankeruun, tallennuksen ja visualisoinnin toteutuksesta ja

mahdollisuuksista toimialasta riippumatta, havainnollistaen pilottitoteutuksella kuinka vastaava toteutus olisi mahdollista toteuttaa.

Pihla Group Oy:n verkkosivuilla yrityksen perustiedot esitetään seuraavasti.

> Pihla Group Oy on vuonna 2004 perustettu yritys, joka on osa Inwido AB:ta. Yritys valmistaa ikkunoita ja ovia Pihla-, PihlaPRO, Tiivi-, Profin-, Klas1, Sydänpuu, Metallityö Välimäki, Puuseppien -tuotemerkeillä, sekä ilmanvaihtoratkaisuja Finluft-tuotemerkillä ja auringonsuojaratkaisuja Artic-tuotemerkillä. Pihla Group Oy:lla on kymmenen tuotantolaitosta, jotka sijaitsevat Ruovedellä, Kannuksessa, Haapajärvellä, Pudasjärvellä, Kuusamossa, Joutsassa, Nokialla, Hyvinkäällä, Ylöjärvellä ja Joensuussa. Näistä suurin sijaitsee Ruovedellä, joka työllistää noin 200 työntekijää. (Pihla Group Oy, n.d.)

Tuotanto työskentelee pääasiassa arkisin kahdessa tai kolmessa vuorossa riippuen tuotannon kuormituksesta ja tuotantolinjasta.

Kuvassa 1 näkyvään Ruoveden tehdasalueeseen kuuluu avautuvan- ja kiinteiden ikkunoiden tuotantotilojen lisäksi ovitehdas sekä useita varastohalleja.

Kuva 1. Ilmakuva Pihla Group Oy:n Ruoveden tehdasalueesta (Solarigo Systems Oy, henkilökohtainen tiedonanto, 2023).

2 Taustatiedot ja teknologiat

Teollisuuden tiedonkeruujärjestelmät keräävät tietoa antureista, tuotantolinjoilta, valvontajärjestelmistä ja muista automaatiolaitteista. Tämä tieto voi sisältää useita eri muuttujia, kuten lämpötilaa, painetta, virtausnopeutta ja värähtelyä, jotka kuvaavat tuotantoprosessin tilaa ja koneiden suorituskykyä. Tiedonkeruujärjestelmiä hyödynnetään erityisesti prosessien optimointiin, ennakoivaan kunnossapitoon ja laadunvalvontaan.

Tyypillisesti tiedonkeruu teollisuuslaitoksissa perustuu teollisuusjärjestelmien valvonnassa käytettäviin SCADA-järjestelmiin (Supervisory Control and Data Acquisition) tai niiden ohjauksessa käytettäviin PLC-laitteisiin (Programmable Logic Controller). Näiden järjestelmien avulla kerätty tieto voi tukea päätöksentekoa ja optimointia, joilla voidaan tehostaa tuotantoa, parantaa energiatehokkuutta ja kehittää koko toimintaketjua. Tiedon analysointi onkin kriittistä yrityksen kilpailukyvyn säilyttämiseksi ja parantamiseksi. (Muelaner, 2021)

2.1 OEE Mittarointi

OEE (Overall Equipment Effectiveness) eli suomeksi KNL (käytettävyys, nopeus, laatu) on teollisuudessa laajasti käytetty, kolmesta osa-alueesta koostuva mittari, joka auttaa arvioimaan tuotantoprosessin tehokkuutta. Täydellisessä tilanteessa koneet käyvät häiriöttä, niiden tuotantonopeus on maksimissaan ja tuotannon laatuvirheet on saatu poistettua. Tällöin OEE-arvo olisi 100 %, kuitenkin käytännössä tämän saavuttaminen haastavaa. Pinjan mukaan valmistavassa teollisuudessa keskimääräinen OEE-luku on noin 60 %. OEE lasketaan kertomalla käytettävyyden, nopeuden ja laadun prosenttiosuudet keskenään. (Pinja, 2022, ss. 5–6)

OEE-mittarissa on tärkeää ymmärtää, mitä kunkin osa-alueen taustalla tapahtuu. Pelkän OEE-arvon seuraaminen ei riitä, vaan huomio tulee kiinnittää käytettävyyden, nopeuden ja laadun yksittäisiin tekijöihin. Näin saadaan esimerkiksi selville missä kohtaa tuotantoprosessia esiintyy eniten hukkaa ja mihin kehittämistoimenpiteet kannattaa kohdentaa. Lisäksi on varottava OEE-mittauksen tyypillisiä sudenkuoppia, kuten esimerkiksi vaihtoaikojen poisjättämistä, virheellisen nopeusarvon käyttöä ja liiallista datan keruuta, joka voi kuormittaa operaattoreita ilman todellista lisähyötyä. (Pinja, 2022, ss. 8–11)

OEE ei ole välttämättä paras tai ainoa tunnusluku jokaiseen tuotantoympäristöön. Sen soveltuvuus tulee arvioida yhdessä muiden mittareiden kanssa. Lisäksi on tärkeää varmistaa, että tuotantolaitoksen perustiedot kuten ymmärrys laitteiden nopeuksista ja vallitsevista käyttöolosuhteista on kunnossa. Vasta tämän jälkeen OEE-mittaus palvelee tarkoitustaan ja toimii tehokkaana työkaluna tuotannon tehostamiseksi.

2.1.1 Tuotannon optimointi datankeruulla

Kerätyn tiedon avulla voidaan optimoida tuotantoprosesseja siten, että resurssien käyttö tehostuu ja tuotannon laatu paranee. Tuotannon optimointi hyödyntää tietoa monilta eri osa-alueilta, kuten koneiden käyttöasteesta, raaka-aineiden kulutuksesta, energiatehokkuudesta, asetteenvaihdoista ja tuotantolinjan pullonkauloista. Tähän tarkoitukseen voidaan käyttää analytiikkatyökaluja, kuten koneoppimista ja algoritmeja, joiden avulla voidaan optimoida tuotantoparametreja tai tuotantojärjestystä. Esimerkiksi energian ja raaka-aineiden kulutuksen minimointi saavutetaan analysoimalla prosessien tehokkuutta ja tunnistamalla mahdollisuuksia resurssien käytön parantamiseen. (Alisa, 2023)

2.1.2 Ennakoiva kunnossapito

Ennakoiva kunnossapito on myös yksi keskeisimmistä alueista, joissa kerättyä dataa hyödynnetään. Perinteisesti kunnossapito on perustunut joko vikatilanteisiin tai kiinteisiin huoltoaikatauluihin, mutta älykkäässä ennakoivassa kunnossapidossa analysoidaan reaaliaikaista anturi- ja käyttödataa, kuten lämpötilaa, tärinää tai painetta. Tämän tiedon analysointi mahdollistaa huoltotarpeen ennustamisen ennen vikaantumista, mikä vähentää odottamattomia tuotantokatkoksia ja parantaa tuotantoprosessien luotettavuutta. (VTT, n.d.)

2.1.3 Laadunvalvonta

Reaaliaikainen tiedonkeruu tuotantolinjoilta mahdollistaa laadun poikkeamien havaitsemisen varhaisessa vaiheessa ja mahdollistaa nopean reagoinnin ongelmatilanteisiin. Historiallisen datan analysointi auttaa tunnistamaan toistuvat ongelmat ja mahdollistaa muutokset laatuongelmien vähentämiseksi. Laadunvalvonnassa hyödynnetään yhä useammin kehittyneitä analytiikkatyökaluja ja tekoälyä, jotka voivat ennustaa laatupoikkeamia jo ennen niiden syntymistä. Tämä tapahtuu vertaamalla antureilta tulevaa tietoa mallinnettuihin "normaaliolosuhteisiin", mikä mahdollistaa mahdollisten poikkeamien ennakoinnin. (Pinja, 2023)

2.2 Sarjanumerot

Yksilöivät sarjanumerot ovat olennainen osa valmistavan teollisuuden tietokeruuta, prosessien hallintaa ja jäljitettävyyttä. Jokaiselle valmistetulle tuotteelle, osalle tai komponentille voidaan antaa yksilöivä sarjanumero, joka helpottaa sen seurantaa koko elinkaaren ajan. Joillakin teollisuuden aloilla jäljitettävyys voi olla lakisääteistä, kuten esimerkiksi elintarviketeollisuudessa (Simola, 2019, s. 6).

Sarjanumerot mahdollistavat tuotteiden ja komponenttien jäljitettävyyden. Esimerkiksi jos jokin osa osoittautuu vialliseksi, sarjanumeron avulla voidaan nopeasti selvittää, missä ja milloin osa on valmistettu, sekä ketkä ovat toimittaneet siihen liittyvät raaka-aineet tai komponentit. Näin laatuongelmien lähde pystytään paikantamaan tarkasti ja korjaavat toimenpiteet voidaan kohdentaa juuri oikeille tuotantolinjoille tai alihankkijoille. (Simola, 2019, s. 14)

Sarjanumeroiden käytön merkitys kasvaa, kun valmistetaan yksilöityjä mittatilaustuotteita kuten ikkunoita ja ovia, koska erilaisia tuotevariaatioita voi olla tuhansia ja yksittäisen komponentin puuttuminen voi estää loppukokoonpanon. Kun jokaiselle valmistettavalle tuotteen osalle on annettu sarjanumero, voidaan seurata jokaisen osan etenemistä tuotantolinjalla. Tuotannon eri vaiheissa voidaan tallentaa tietoja, kuten koneiden suorituskyvystä, lämpötiloista ja aseteajoista, jotka yhdistetään osan sarjanumeroon. Jos valmiissa tuotteessa myöhemmin ilmenee laatuongelma, nämä tiedot auttavat selvittämään, missä ja milloin tuote on valmistettu sekä mitä prosessiparametreja on ollut käytössä. Näin virheiden juurisyyt löytyvät tehokkaammin ja niitä voidaan jatkossa ennaltaehkäistä. (Simola, 2019, ss. 8, 36, 54)

Sarjanumeroita voidaan hyödyntää automaatiossa tarjoamalla tuotantokoneille reaaliaikaista tietoa kunkin osan työvaiheista. Esimerkiksi ikkunoiden valmistuksessa käytettävä työstökone voisi automaattisesti hakea tietokannasta karmiin tarvittavien työstöjen paikkatiedot luetun sarjanumeron perusteella. Tämä mahdollistaisi joustavamman tuotantojärjestyksen nykyisen ennalta määritetyn valmistusjärjestyksen sijaan ja vähentäisi operaattorivirheiden riskiä.

2.2.1 Merkintätavat

Sarjanumeron merkitsemisessä tuotteeseen voidaan käyttää useita eri tekniikoita ja menetelmiä, jotka valitaan tuotteen vaatimuksien ja käyttöympäristön perusteella.

Sarjanumeron merkintätavan valinta riippuu useista tekijöistä, kuten tuotteen materiaalista, elinkaaresta ja vaaditusta merkinnän kestävyydestä.

Alla viisi yleistä tapaa merkitä sarjanumero:

1. Lasermerkintä: Tehtävä merkintä on pysyvä tapa sarjanumeron merkitsemiseen. Oikein valitulla laserin tyypillä merkintä voidaan tehdä lähes mihin tahansa kiinteään materiaaliin.
2. Tarrat ja etiketit: Sarjanumero voidaan tulostaa tarraan tai etikettiin, joka kiinnitetään tuotteeseen. Tarrat ovat yleisesti käytetty menetelmä tuotteissa, joissa pysyvä merkintä ei ole välttämätön.
3. Iskumerkintä: Mekaaninen merkintätapa, jossa teräskärkeä käytetään pysyvän merkinnän tekemiseen metallin tai muun kovan materiaalin pintaan.
4. Mustesuihkutulostus: Nopea tapa sarjanumeroiden merkitsemiseen eri materiaaleihin, kuten metalleihin, muoveihin, pahviin tai lasiin. Se soveltuu erityisesti sarjatuotantoon, jossa tarvitaan ei-pysyviä tai pysyviä merkintöjä vähäisellä kulumisen kestävyydellä.
5. Kemiallinen merkintä: Perustuu sähkökemialliseen prosessiin, jossa elektrolyyttiä ja virtalähdettä käyttäen saadaan aikaan pysyvä merkintä metallipintaan. Tämä merkintätapa sopii vain metallituotteiden merkintään. (Simola, 2019, ss. 23–31)

Pihla Groupilla näistä merkintätavoista on yleisimmässä käytössä tarrat, lasermerkintä ja mustesuihkutulostus. Aikaisemmin on käytetty pääasiassa mustesuihkumerkintää yksinkertaisiin numeromerkintöihin, mutta uudemmissa merkintätarpeissa on siirrytty käyttämään lasermerkintää sen joustavuuden ja vähäisten käyttökustannuksien takia. Lasermerkinnän etuina on myös parempi kulutuksen kestävyys ja mahdollisuus saada merkintä näkymään maalatun pinnan alta.

2.2.2 Datamatriisi

ECC200-matriisikoodi, joka tunnetaan myös nimellä datamatriisi, on kaksiulotteinen koodausmenetelmä, mikä mahdollistaa suuren tietomäärän tallentamisen pieneen tilaan (Deuil ym., 2007, s. 27). Tämä koodi voi sisältää esimerkiksi sarjanumeron, valmistusajankohdan sekä muita tuotetietoja.

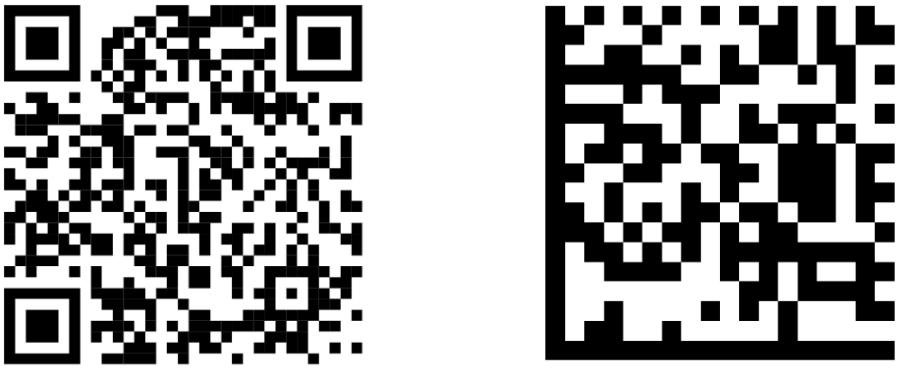
Matriisikoodit ja yleisemmin tunnetut QR-koodit ovat molemmat kaksiulotteisia tunnistusmerkintöjä, mutta niiden rakenteessa ja käyttötarkoituksissa on eroja. Datamatriisikoodi on eriväristä soluista koostuva kaksiulotteinen koodi, mikä on tavallisesti

neliömäinen, vaikka myös suorakulmion muotoisia versioita on olemassa. Sen solujen määrä lisääntyy tallennetun tiedon määrän kasvaessa, ja sen tallennuskapasiteetti on enintään 2 335 aakkosnumeerista merkkiä. Koodin reunan L-muotoinen rakenne toimii tunnistuselementtinä skannereille, jotka lukevat koodin. (Tremblay, 2019)

QR-koodi on myös erivärisistä soluista koostuva kaksiulotteinen koodi. Se tallentaa enimmillään 4 296 aakkosnumeerista merkkiä. QR-koodin tunnistuselementteihin kuuluvat neliörakenteet, jotka helpottavat koodin tunnistamista. (Tremblay, 2019)

Vaikka datamatriisi- ja QR-koodit saattavat vaikuttaa samankaltaisilta, niissä on kuitenkin eroja, esimerkiksi datamatriisikoodit ovat kompaktimpia ja soveltuvat paremmin pienten esineiden, kuten elektronisten komponenttien merkintään. Datamatriisikoodit voivat sisältää suuremman määrän merkkejä pienemmässä tilassa verrattuna QR-koodeihin. QR-koodit sen sijaan sopivat hyvin tapauksiin, joissa tarvitaan suurempaa merkkimäärää. Molemmat koodit ovat skaalattavia, mutta datamatriisikoodien pienemmän koon ansiosta ne soveltuvat paremmin pieniin merkintäpaikkoihin. (Tremblay, 2019). Kuvassa 2 on esitetty samalla sisällöllä oleva QR-koodi ja ECC200-matriisikoodi ja siitä voidaan havaita, että QR-koodin pistekoko on huomattavasti pienempi.

Kuva 2. QR- ja datamatriisikoodi.



QR-koodi Datamatriisi

Kuvassa 3 on esimerkki Pihla Group Oy:n Ruoveden tehtaalla käytössä olevista ECC200-matriisikoodimerkinnöistä, joita käytetään ikkunan karmien yksilöimiseen. Lasermerkintälaitteella tuotetut merkinnät sisältävät matriisikoodin, joka mahdollistaa kappaleiden identifioinnin. Jokaisessa karmissa on kaksi identtistä merkintää, jotta tiedot säilyvät luettavina myös siinä tapauksessa, että toinen merkintä vaurioituu tai sijoittuu oksakohdan päälle.

Kuva 3. Matriisikoodilla merkittyjä karmeja.

2.3 Teollisuuden tiedonsiirtoprotokollat

Teollisuuden digitalisaatio ja IoT-ratkaisut perustuvat suurelta osin siihen, että erilaiset laitteet ja järjestelmät voivat jakaa dataa keskenään luotettavasti ja mahdollisimman reaaliaikaisesti. Yleisesti käytettyjä protokollia ovat esimerkiksi MQTT, OPC UA, Modbus TCP ja Profinet, jotka kaikki tarjoavat omia etujaan ja soveltuvat erilaisiin käyttötarkoituksiin.

OPC (Open Platform Communications) on laajasti käytetty teollisuudessa erityisesti OPC UA standardin muodossa. Sen arkkitehtuuri koostuu tyypillisesti yhdestä tai useammasta palvelimesta (OPC server) ja niitä hyödyntävistä asiakkaista (OPC client). Standardin kehityksessä on panostettu erityisesti tietoturvaan. Lisäksi se on alustariippumaton, mikä helpottaa eri laitealustojen ja valmistajien järjestelmien yhteensopivuutta. (Tapia ym., 2023, ss. 5–6)

Modbus on puolestaan pitkään käytössä ollut teollisuuden kommunikaatioprotokolla, joka myös perustuu client/server malliin. Modbus TCP hyödyntää TCP/IP yhteyksiä, ja asiakkaat lukevat tai kirjoittavat dataa palvelimena toimivalta laitteelta. Modbusin vahvuuksiin kuuluvat laaja laitetuki, luotettavuus ja avoin lähdekoodi. Tietoturvan näkökulmasta Modbus TCP:ssä ei ole sisäänrakennettuna salaus- tai todennusmekanismeja, joten näiden käyttö vaatii erillisiä ratkaisuja kuten VPN tai TLS/SSL käyttöä. (Tapia ym., 2023, ss. 3–5)

Profinet on moderni ethernet-pohjainen kenttäväylätekniikka, mikä on suunniteltu teollisuuden reaaliaikaiseen tiedonsiirtoon. Se pohjautuu avoimeen ja standardoituun arkkitehtuuriin, joka tarjoaa korkean siirtonopeuden, joustavuutta sekä vikasietoisuutta. Profinet on laajalti käytössä etenkin ohjain- ja kenttälaitteiden välisessä kommunikaatiossa. (Tapia ym., 2023, ss. 4–5)

MQTT (Message Queuing Telemetry Transport) on avoin tiedonsiirtoprotokolla, mikä kehitettiin vuonna 1999 alun perin resurssirajoitteisia laitteita ja pienikaistaisia yhteyksiä varten. MQTT:n suosio on kasvanut erityisesti teollisuuden IoT-ratkaisuissa sen kevyen rakenteen ja tehokkuuden ansiosta. Protokolla hyödyntää julkaisija-tilaaja-mallia (publish-subscribe), jossa viestit välitetään asiakkaiden (client) ja palvelimen (broker) välillä ilman suoraa yhteyttä viestien lähettäjän ja vastaanottajan välillä. (MQTT.org, n.d.)

Julkaisija-tilaaja-mallissa laitteet toimivat joko julkaisijoina tai tilaajina. Julkaisijat lähettävät viestejä tiettyihin aiheisiin (topics), ja tilaajat tilaavat kyseisiä aiheita saadakseen niihin liittyvät viestit. Tällä tavoin palvelin eli broker, välittää viestit automaattisesti oikeille vastaanottajille.

Teollisuusympäristöissä MQTT:n hyödyt korostuvat erityisesti suurissa IoT-verkoissa, joissa voi olla tuhansia laitteita, jotka lähettävät dataa jatkuvasti. Nämä anturit voivat tuottaa pieniä määriä tietoa, mutta hyvin tiheästi, esimerkiksi reaaliaikaisessa koneiden valvonnassa. (King Ho, 2021)

2.4 Node-RED ohjelmointiympäristö

Node-RED on avoimen lähdekoodin ohjelmointialusta, joka on erityisesti suunniteltu käyttämään ns. low code -lähestymistapaa tapahtumalähtöisten järjestelmien toteutuksessa (OpenJS Foundation & Contributors, n.d.-b). Se soveltuu erityisen hyvin tilanteisiin, joissa halutaan nopeasti rakentaa ja testata prototyyppejä tai kytkeä eri järjestelmien tiedonsiirtoputkia toisiinsa ilman raskasta ohjelmointiprosessia.

Node-REDin suurin etu verrattuna esimerkiksi perinteiseen Python-ohjelmointiin tai räätälöityihin C/C++ sovelluksiin on sen graafinen käyttöliittymä. Node-RED toimii verkkoselaimessa, mikä tekee siitä helposti käytettävän ja monipuolisen työkalun eri alustoilla (OpenJS Foundation & Contributors, n.d.-b).

Node-RED-ohjelmointi perustuu visuaaliseen tapaan, jossa käytetään erilaisia moduuleita, joita kutsutaan nodeiksi. Node on yksittäinen moduuli, joka suorittaa tietyn tehtävän. Node voi olla esimerkiksi HTTP-pyyntö, tietojen muokkaus, anturitiedon lukeminen tai sähköpostin lähettäminen (OpenJS Foundation & Contributors, n.d.-d). Käyttäjä rakentaa toiminnallisuuksia raahaamalla nodeja käyttöliittymän vasemmasta reunasta työskentelyalueelle. Tämän jälkeen nodet yhdistetään toisiinsa halutulla tavalla, muodostaen ketjuja, joita kutsutaan tietovirtauksiksi (flows) (OpenJS Foundation & Contributors, n.d.-c).

Flow on useiden nodejen muodostama tietovirta, jossa yksittäiset nodet suorittavat toimintansa ketjutettuna. Flow:n tarkoitus on kuljettaa ja muokata tietoja nodelta toiselle, kunnes saavutetaan haluttu lopputulos. Jokainen flow alkaa yleensä tiedon lähteestä, esimerkiksi anturista, palvelimelta tai muusta tiedon tuottajasta. Se päättyy määriteltyyn kohteeseen, kuten tietokantaan, konsoliin tai verkkopalvelun rajapintaan. (OpenJS Foundation & Contributors, n.d.-c)

Node-RED perustuu Node.js-ympäristöön, mikä tekee JavaScriptistä sen keskeisen ohjelmointikielen (OpenJS Foundation & Contributors, n.d.-b). JavaScript on ohjelmointikieli, jota käytetään erityisesti verkkosovellusten kehittämisessä, mutta sen käyttö on laajentunut palvelinpuolelle, esimerkiksi Node.js-ympäristön kautta. Node.js on JavaScript-pohjainen ajonaikainen alusta, jonka avulla voidaan suorittaa JavaScript-koodia palvelimella (OpenJS Foundation & Contributors, n.d.-a). Vaikka Node-RED on suunniteltu visuaalista ohjelmointia varten, käyttäjät voivat tarvittaessa käyttää JavaScript-koodia esimerkiksi funktio-nodessa.

2.5 Tietokannat

IoT-järjestelmissä kerätään suuria määriä anturi-, laite- ja prosessidataa, joka on tyypillisesti aikasidonnaista. Tällöin nousee esiin kysymys, millaiseen tietokantaan data kannattaa tallentaa. Yleisiä vaihtoehtoja ovat perinteiset relaatiotietokannat (esimerkiksi MySQL, SQL Server, PostgreSQL), NoSQL tietokannat (kuten MongoDB, Cassandra) ja erityisesti aikasarjoihin suunnitellut tietokannat (kuten InfluxDB, TimescaleDB ja Prometheus). (Mahler, 2022)

Perinteisten relaatiotietokantojen etuna on niiden laaja tunnettavuus ja SQL kielen hyödyntäminen. Ne eivät kuitenkaan ole aina optimaalisia suurilla datamäärillä. Aikasarjoihin perustuvat tietokannat on suunniteltu niin, että data tallennetaan mahdollisimman tehokkaasti. Niissä korostuu suurien tietomäärien nopea kirjoitus- ja lukunopeus, suorituskyvyn säilyttäminen kasvavilla datamäärillä sekä helppokäyttöiset analysointityökalut. Tästä syystä aikasarjatietokanta on monissa IoT-tapauksissa paras valinta. Aikasarjatietokantojen heikkoutena on, että niihin tallennetun datan muokkaaminen tai poistaminen ei yleensä ole suositeltavaa tai edes mahdollista. Tämä johtuu aikasarjatietokannoissa käytetyistä indeksointi- ja tallennustavoista. (Mahler, 2022)

InfluxDB on InfluxData-yhtiön kehittämä aikasarjatietokanta, mikä valikoitui työssä käytettäväksi tietokannaksi aikaisempien käyttökokemuksien perusteella. InfluxDB ydinominaisuus on sen kyky tallentaa ja käsitellä suuria määriä aikasidonnaista dataa, kuten IoT-laitteiden anturidataa tai palvelinmittauksia. (Thamatam, 2022)

InfluxDB tietokanta sisältää kaksi pääkomponenttia: tag keys ja field keys. Tag keys -arvoja käytetään tunnisteina indeksoinnissa ja kyselyiden suorittamisessa ja niiden tulisi olla mahdollisimman pysyviä. Field keys puolestaan tallennetaan usein muuttava data, kuten mittausarvot. (InfluxData Inc., n.d.-a).

InfluxDB V2 käyttää kyselykielenä Fluxia, joka on erityisesti kehitetty aikasarjojen ja sensoridatan analysointiin. Flux eroaa perinteisistä kyselykielistä siinä, että se muistuttaa enemmän ohjelmointikieliä, koska kyselyt suoritetaan ylhäältä alaspäin. Fluxin avulla käyttäjät voivat kirjoittaa joustavia ja monimutkaisia analyysejä aikasarjatiedoista, mikä tekee siitä tehokkaan työkalun tietyissä käyttötapauksissa. Kuitenkin Flux-kielen jatkokehitys on lopetettu ja InfluxDB V3 tukee jatkossa SQL-kieltä kyselyiden tekemiseen. SQL kieli on huomattavasti tunnetumpi maailmalla ja yleinen tapa käsitellä tietokantoja. SQL käyttö täten mahdollistaa myös laajempien kehitysyhteisöjen hyödyntämisen ja helpottaa InfluxDB:n integroimista muihin tietokantajärjestelmiin ja työkaluihin. (InfluxData Inc., n.d.-c)

Työn toteuttamisen aikana V3 ei ollut vielä saatavilla Azure pilvipalvelussa olevissa InfluxDB ympäristöissä, jonka takia työn tietokantakyselyt toteutettiin Flux-kielellä.

2.6 Datan visualisointi

Datankeruusta kertyy usein suuri määrä dataa, jonka ymmärrettävä esittäminen edellyttää erilaisten visualisointialustojen käyttöä. Moni teollisuuden toimija turvautuu liiketoimintatiedon

hallintaan kehitettyihin BI-ohjelmistoihin tai suosivat valmiita pilvipohjaisia palveluita, joissa analytiikan ja raportoinnin työkalut on nivottu yhteen tallennus- ja tiedonkeruualustojen kanssa. Maksulliset BI-järjestelmät, kuten Microsoftin Power-BI keskittyvät visuaalisesti näyttäviin koontinäyttöihin ja liiketoimintaprosessien tarkasteluun. Alustan valinnassa vaikuttavat käyttötarkoitus, datavirtojen tyyppi, tietoturvavaatimukset sekä laajentumismahdollisuudet. (MetricFire, 2023)

Työssä käytettävä Grafana on avoimen lähdekoodin tietojen visualisointi- ja seurantaohjelmisto, jota käytetään mittaristojen, hälytysten ja raporttien luomiseen. Se mahdollistaa eri lähteistä kerättyjen tietojen esittämisen visuaalisesti selkeässä ja helposti ymmärrettävässä muodossa. Grafanan käyttö on ollut yleistä erityisesti IT-sektorilla, mutta sen käyttö on lisääntynyt myös IoT ratkaisuissa sen joustavuuden ja monipuolisten ominaisuuksien vuoksi (Gontcharov & Theocharis, 2024).

Grafana kehitettiin alun perin vuonna 2014 Torkel Ödegaardin toimesta, ja se on kasvanut yhdeksi suosituimmista tietojen visualisointi- ja seuranta-alustoista maailmassa. Grafana tukee useita eri tietolähteitä, kuten InfluxDB, Prometheus, Graphite, SQL ja monia muita. Grafana on avoimen lähdekoodin työkalu, joten se on maksuton. Grafana Labs tarjoaa myös avoimen lähdekoodin projektin päälle rakennettuja yrityksille suunnattuja ominaisuuksia lisämaksusta. (Gram, n.d.)

Yksi Grafanan keskeisiä ominaisuuksia on mittaristot (Dashboard), jotka mahdollistavat monipuolisten näkymien luomisen, joihin voi yhdistää eri lähteistä saatavia tietoja. Lisäksi on mahdollista määrittää automaattisia hälytyksiä, joiden avulla käyttäjät voivat saada ilmoituksia, kun määritetyt raja-arvot täyttyvät.

Grafana on erinomainen työkalu moniin seurantatarpeisiin, mutta vaativampiin sovelluksiin voi olla tarpeen käyttää räätälöityjä ohjelmistoratkaisuja. Grafanan rajoitukset tulevat esille erityisesti silloin, kun on tarve tietyille ominaisuuksille tai analytiikalle, joka ei ole suoraan saatavissa Grafanassa. Räätälöidyt sovellukset voivat tarjota useita etuja, kuten tietyn teollisuudenalan eritysvaatimukset, laitteistolle optimoidut ominaisuudet tai integraation yrityksen omiin järjestelmiin.

3 Suunnitelma

Suunnitelman tavoitteena on kartoittaa yrityksen tarpeet datankeruulle ja yhtenäistää aiemmat hajanaiset ratkaisut. Samalla pyritään huomioimaan tulevaisuuden tarpeet, kuten

ennakoiva kunnossapito ja tekoälyn käyttömahdollisuudet, sekä sarjanumeroseuranta, joka on olennainen mittatilaustuotteiden valmistuksessa.

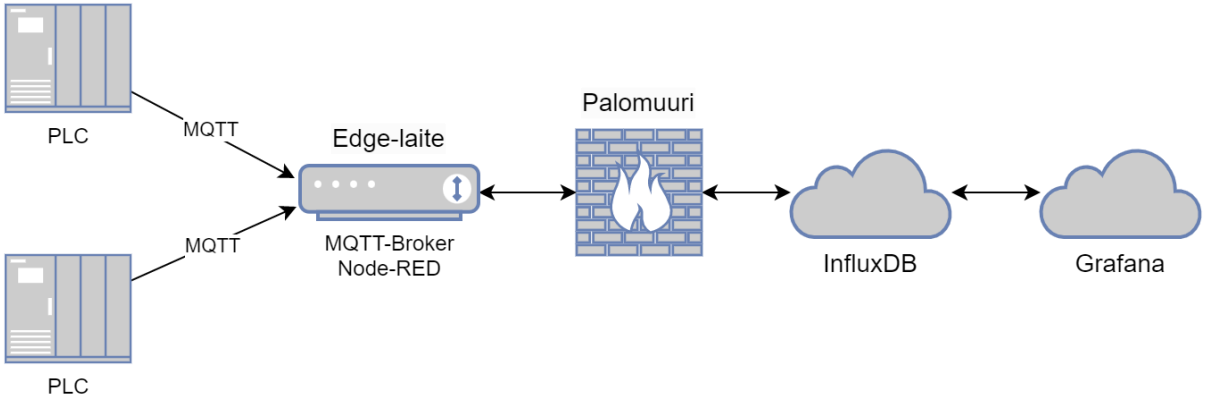
Suunnitteluvaiheessa ja järjestelmän toteutuksessa hyödynnetään mahdollisimman laajasti aikaisempia kartoituksia sekä aiemmista tiedonkeruun toteutuksista saatuja kokemuksia. Aikaisemmissa toteutuksissa InfluxDB-tietokanta, Grafana ja Node-RED on todettu toimiviksi ja luotettaviksi ratkaisuiksi, ja siksi ne valittiin myös tämän järjestelmän perustaksi.

3.1 Järjestelmäarkkitehtuuri

Pihla Group Oy:n Ruoveden tehtaalla on ollut testikäytössä datankeruujärjestelmä, johon on ollut yhdistettynä muutama tuotannon kone. Järjestelmästä saadun aikaisemman kokemuksen myötä, tätä järjestelmää päätettiin jatkokehittää tässä työssä. Olemassa olevan tiedonkeruujärjestelmän arkkitehtuuri perustui valmistajakohtaisien tiedonsiirtoprotokollien, kuten Siemens S7 ja Omron FINS käyttöön. Näiden lisäksi käytettiin avointa Modbus TCP tiedonsiirtoprotokollaa. Näissä toteutuksissa Node-RED luki suoraan koneen PLC:n muistista tarvittavat tiedot määräajoin, yleensä minuutin välein. Tässä ratkaisussa oli kuitenkin useampia ongelmia. Ensinnäkin tiedonsiirto perustui siihen, että palvelin joutui jatkuvasti kyselemään logiikalta tietoja, vaikka niiden arvo ei olisi muuttunut ja tämä jatkuva kysely tallensi turhaa tietoa tietokantaan. Kuitenkin suurimpana puutteena oli, ettei kyseinen tiedonsiirtomenetelmä mahdollistanut aikaleimoihin perustuvan tiedon keräämistä, kuten koneen tilatietoja tai kappaleiden työvaiheaikaleimoja Lisäksi useampaa tiedonsiirtoprotokollaa käytettäessä järjestelmän ylläpito ja laajentaminen on haastavaa. Näiden puutteiden vuoksi vanhan arkkitehtuurin tiedonsiirtoprotokolla päätettiin korvata uudella.

Uudeksi tiedonsiirtoprotokollaksi valikoitui MQTT sen laajan tuen ja joustavuuden vuoksi. MQTT tarjoaa mahdollisuuden tiedonsiirtoon, joka perustuu tapahtumalähtöiseen tiedonvälitykseen. Näin tiedot lähetetään vain silloin, kun ne muuttuvat, mikä vähentää ylimääräisen tiedon siirtoa ja tallentamista. Lisäksi MQTT mahdollistaa tiedon hyödyntämisen tai keräämisen useisiin eri järjestelmiin, mikä parantaa järjestelmän integrointimahdollisuuksia. Kuvassa 4 havainnollistettu uusi järjestelmäarkkitehtuuri yksinkertaistettuna, jossa kaksi tuotantolaitetta on yhdistettynä datankeruuseen. Edge-laite on asennettu omaan erotettuun VLANiin eli virtuaaliseen verkkosegmenttiin, josta on rajoitettu yhteys tuotantoverkkoon. Tämä varmistaa, että tuotantolaitteet eivät ole suoraan yhteydessä internetiin, mikä parantaa tietoturvaa ja vähentää kyberuhkien riskiä.

Kuva 4. Uusi järjestelmäarkkitehtuuri.



Jos järjestelmää halutaan laajentaa useammalla toimipisteelle, on mahdollista käyttää keskitettyä pilvipohjaista MQTT-Brokeria. Tällöin tehdaskohtainen MQTT-Broker määritetään toimimaan siltatilassa (Bridge), joka välittää kaikki viestit pilvessä sijaitsevaan MQTT-Brokeriin.

3.2 Koneen valinta

Työn alkuperäisessä suunnitelmassa valittiin kone, josta ei vielä keretty tietoa. Myöhemmässä suunnitteluvaiheessa kuitenkin selvisi, ettei alun perin suunnitellun laitteen NJ301-1200 logiikka tukenut Omronin tarjoamaa MQTT-kirjastoa. Omronin yhteyshenkilöltä saimme tiedon, että viralliset ohjelmakirjastot sisälsivät tuen TLS-salaukselle, mikä rajaa laitetuen tiettyihin logiikkamalleihin. Saimme kuitenkin epävirallisen version ilman TLS tukea laajemmalla laitetuella. Tämä sisältää tuen NJ301-1200 logiikalle versiosta 1.14 alkaen, mutta tämä ei valitettavasti auttanut, sillä työn kohteena olevassa laitteessa oli versiolla 1.10 oleva logiikka. Laajempi kartoitus osoitti, että suurin osa tuotannon koneiden Omron-logiikoista oli tätä vanhempaa firmware versiota.

Tästä syystä valittu kone jouduttiin vaihtamaan toiseen koneeseen, joka käyttää uudemman sukupolven NX1P2-logiikkaa. Tämä kone oli myös osa olemassa olevaa tiedonkeruujärjestelmää. Jos tiedonkeruuta halutaan myöhemmin laajentaa NJ301-logiikoille, on kuitenkin mahdollista toteuttaa kaikkien kyseisten logiikoiden päivitykset yhdellä kertaa Omronin teknikon toimesta paikan päällä.

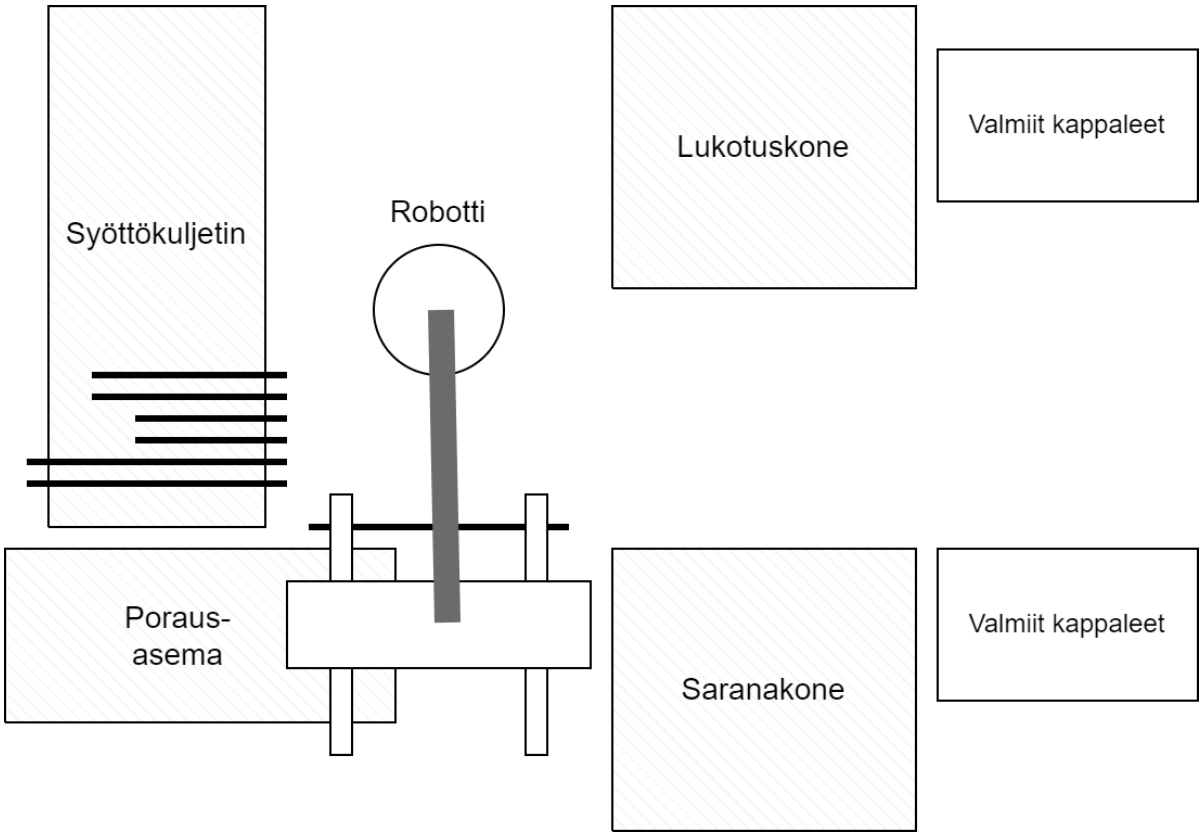
Valintaan vaikutti myös saatavilla olevat tiedot ja tiedon laajuus. Valmistettavien tuotteiden tiedot siirretään useimmiten tuotannon koneille yrityksen resurssinhallintajärjestelmän eli

ERP-järjestelmän (Enterprise Resource Planning) tuottamien CSV-tiedostojen avulla. CSV-tiedosto on yksinkertainen tiedostomuoto, joka sisältää taulukkomuotoista dataa pilkulla tai muulla erotinmerkillä eroteltuna. Nämä CSV-tiedostot ovat jokaiselle tuotannon koneelle räätälöityjä ja sisältävät vain kyseisen koneen tarvitsemat tiedot. Valitun koneen tiedosto sisälsi valmiina kaikki halutut tiedot sarjanumeroa lukuun ottamatta.

Sarjanumero ei ollut kuitenkaan kriittinen työn tässä vaiheessa, sillä logiikka pystyy generoimaan yksilöivän koodin jokaiselle osalle. Tämä on riittävä, kunnes useampien koneiden seurantaa ryhdytään kehittämään. Tälle hetkellä yksilöivä koodi tarjoaa tarpeeksi tarkan seurannan työn tavoitteiden saavuttamiseksi. Sarjanumero on mahdollista lisätä tulevaisuudessa CSV-tiedostoon ERP järjestelmään tehtävällä muutoksella.

Valittu kone on osa tuotantosolua, jossa teollisuusrobotti on keskeisessä roolissa. Robotti siirtää ikkunoiden ulkopuitteiden alumiiniosia eri työstökoneiden välillä. Prosessi alkaa, kun kappale asetetaan syöttökuljettimelle, josta se siirretään porausasemaan. Porausasemalla kappaleeseen tehdään kulma- ja ruuvinreiät, jotka ovat tarpeen myöhempää kasausta ja mahdollisia aukipitolaitteita varten. Työstetty kappale siirtyy tämän jälkeen joko saranointikoneeseen tai lukotuskoneeseen riippuen kappaleeseen tehtävistä työstöistä. Kuvassa 5 on esitetty hahmotelma solusta, jossa robotti ja muut laitteet toimivat, mustat viivat esittävät työstettäviä ulkopuitteita.

Kuva 5. Konesolun yksinkertaistettu layout.



3.3 Kartoitus kerättävästä datasta

Opinnäytetyön tiedonkeruun tarpeiden kartoitus aloitettiin tutkimalla aikaisemmista tiedonkeruujärjestelmistä saatuja kokemuksia. Lisäksi hyödynnettiin Mantec Oy:n tekemän päivittäisjohtamiseen ja tuotannon tehostamiseen liittyvän kartoituksen tuloksia ja kokemuksia (Mantec Oy, henkilökohtainen tiedonanto, 2022). Näistä kaikista nousi esiin erityisesti tarve kerätä tarkempaa, osakohtaisesti yksilöityä tietoa valmistettavista tuotteista.

Pihlan valmistamat ikkunat ja ovet ovat kaikki mittatilaustuotteita, joten kunkin tuotteen osan yksilöivän tiedon kerääminen on olennaista. Tärkeä osa tätä tiedonkeruuta on osien yksilöivän koodin, eli sarjanumeron tallentaminen. Tämä mahdollistaisi esimerkiksi tuotetyyppien vaikutusten seurannan tuotannon tehokkuuteen sekä reaaliaikaisen komponenttien valmistumisen seurannan, mikä tukisi tuotannon hallintaa ja helpottaisi ongelmien varhaista tunnistamista. Tuotekohtainen tieto auttaisi myös paikantamaan tuotantolaitteiden ongelmakohtia, kuten raaka-ainehukan seurantaa työvaihe- ja tuotetyypeittäin.

Huomiota on kiinnitetty siihen, ettei siirrettäviin tietoihin sisälly asiakastietoja, jotta yksityisyydensuoja säilyy.

Tiedonkeruuta varten koostettiin lista tiedoista, jotka tulisi kerätä automaattisesti tuotantoprosessin yhteydessä. Nämä tiedot sisältävät:

1. Koneen tilatieto (esimerkiksi seis, pysäytetty, huolto tai tuotanto)
2. Koneen häiriön tai pysäytyksen syyt
3. Valmistuneiden, epäonnistuneiden ja uudelleentehtyjen kappaleiden yksilöivät tiedot kuten sarjanumero, tyyppi, pituus ja kappalemäärä
4. Koneen mittaukset (esimerkiksi lämpötila, virta, paine, tärinä, energiankulutus, käyttöaika ja käyttömäärä).

Kaikkea tietoa ei kuitenkaan ole mahdollista kerätä automaattisesti tai automaattinen keruu saattaa osoittautua liian monimutkaiseksi. Näissä tapauksissa tiedonkeruu vaatii käsin tehtävää syöttöä operaattorin tai muun työntekijän toimesta. Käsin kerättävää tietoa ovat:

1. Koneen pysäytyksen tai vian tarkemmat syyt
2. Vuorotiedot (milloin koneen kuuluisi käydä)
3. Operaattoritiedot (nimimerkki, määrä)
4. Laatutarkastukset ja mittaukset.

4 Toteutus

Toteutuksen alussa varmistettiin kaikkien tarvittavien ohjelmistojen saatavuus ja asennettiin puuttuvat ohjelmistot. Järjestelmässä käytettävät ohjelmistoista InfluxDB ja Grafana oli jo valmiiksi hankittu Microsoftin Azure-pilvipalvelun kautta ennen varsinaisen työn aloittamista. Lisäksi Node-RED oli asennettuna paikalliselle palvelimelle.

Eclipse-Mosquitto MQTT Broker asennettiin samalle paikalliselle palvelimelle, jolla Node-RED jo sijaitsi. Asennus tehtiin käyttämällä Docker-kontti asennusta eli ohjelmistot toimivat omilla virtuaalialustoilla. Docker mahdollistaa ohjelmistojen nopean uudelleenasennuksen eri laitteille ja helpottaa ohjelmistojen päivittämistä, koska asennukseen tarvitaan vain määritystiedosto.

4.1 MQTT

MQTT-protokollan valintaan vaikutti erityisesti sen tarjoama push-malli, jossa laitteet lähettävät tietoa palvelimelle tarpeen mukaan, toisin kuin perinteisessä pull-mallissa, jossa palvelin hakee tiedot logiikasta ennalta määrätyin väliajoin. Push-malli on tehokkaampi, erityisesti silloin, kun tarvitaan epäsäännöllisten tapahtumien, kuten kappaleen työstön aloitus- ja lopetusaikaleimojen, tallennusta.

Lisäksi MQTT mahdollistaa kerätyn tiedon laajan käytön, sillä sama tieto on luettavissa useista eri paikoista samanaikaisesti. Tämä tekee siitä erinomaisen valinnan, kun tietoa tarvitaan eri järjestelmien käyttöön samanaikaisesti.

Koneiden ja kappaleiden tilan seuranta perustuu siihen, että viesti lähetetään aina, kun tilassa tapahtuu muutoksia. Esimerkiksi koneen tilan muutokset (käynnissä, seis, huolto) tai kappaleen tilan muutokset (otettu työstöön, valmistunut, epäonnistui).

MQTT-topicien rakenteessa noudatettiin alla listattuja HiveMQ:n blogissa mainittuja parhaita käytäntöjä, jotta rakenne pysyisi käytettävänä ja selkeänä (HiveMQ Team, 2024).

1. Ei välilyöntejä
2. Pidä rakenne lyhyenä ja selkeänä
3. Käytä hierarkkista rakennetta, jossa osiot erotetaan kauttaviivalla (/)
4. Vältä erikoismerkkejä.

Viestirakenteen suunnittelu osoittautui monimutkaiseksi, koska se vaikuttaa suoraan tietokantaan ja siihen, kuinka tietokantakyselyitä voidaan myöhemmin suorittaa. Viestirakenteen suunnittelun tässä vaiheessa oli jo huomioitava, millaisia tietoja tarvitaan tietokantakyselyiden helpottamiseksi tulevaisuudessa. Tämä edellytti ennakointia, jossa huomioitiin mahdollisimman kattavasti tulevat tarpeet ja käyttötapaukset.

Suunnittelun pohjalta valmistui kolme erilaista viestirakennetta, jotka kattavat mahdollisimman monta erilaista käyttötarvetta. Kaikkiin viesteihin on suositeltavaa lisätä laitteen kellonaika, vaikka tämä ei olisi tarkka, koska tämä voi auttaa ongelmatilanteiden selvittämisessä. Kaikki viestit käyttävät JSON-formaattia. JSON (JavaScript Object Notation) on kevyt ja ihmisen luettavissa oleva tiedonvaihtoformaatti, joka mahdollistaa helpon rakenteellisen tiedon jäsentämisen ja käsittelyn eri sovelluksissa.

4.1.1 Kappaleen tilatieto

Kuvassa 6 on esimerkki ensimmäisestä viestirakenteesta, joka kuvaa yksittäisen kappaleen etenemistä valmistusprosessissa. Sen avulla tiedetään, milloin kappale on otettu työn alle, mihin sarjaan se kuuluu ja millä tekonumerolla ja sarjanumerolla se on merkitty. Viestissä ilmenee, minkä tyyppisestä osasta on kyse, kuinka monta kappaletta kone käsittelee samanaikaisesti ja mikä on kappaleiden pituus. Lisäksi siitä selviää kappaleen senhetkinen tila, esimerkiksi onko se juuri aloitettu vai onko työvaihe saatu valmiiksi. Viesti antaa myös tiedon siitä, kuinka kauan kappale on viipynyt edellisessä tilassaan, minkä avulla voidaan arvioida esimerkiksi työvaiheiden kestoa ja tuotannon sujuvuutta.

Viestin topic muodostuu seuraavasti: {factory}/{location}/{line}/{machine}/part_status.

Kuva 6. Kappaleen tilatietoviestin esimerkki.

```
{
  "timestamp": "2024-05-31T12:00:00Z", // Aikaleima RFC3339 tai ISO 8601 muodossa
  "batch": "R2423001", // Sarja (string)
  "build_number": "13", // Tekonumero (string)
  "part_type": "ULKOPUITE 4 MM", //Osan tyyppi (string)
  "part_count": 1, // Kappalemäärä eli monta kappaletta kone käsittelee kerralla (INT)
  "part_length": 1320, // Pituus (INT)
  "part_serial": "311331100592-1-26-3-U-4-1", // Kappaleen sarjanumero (string)
  "status": "started", // Tilatieto (string) started/done/redone/failed
  "duration": 25 // Aika sekunteina kauan kappale oli aikaisemmassa tilassa, aloitusviessä (started) arvo on 0 (INT)
}
```

4.1.2 Koneen tilatieto

Toista kuvassa 7 esitettyä viestirakennetta käytetään koneen toimintatilan seuraamiseen. Se ilmoittaa, milloin koneen tila muuttuu, mikä on uusi tila ja onko kyse tilan alkamisesta (incoming) vai päättymisestä (outgoing). Alkamistietoa käytetään reaaliaikaisessa aikajanassa ja päättymistietoa KNL laskennassa. Viestissä ilmoitetaan myös syy koneen tilan muutokselle. Esimerkiksi huolto tai asetusten muutos, sekä kuinka kauan kone on ollut kyseisessä tilassa ennen tämän päättymistä. Lisäksi viesti voi sisältää kappalesarjanumeron, jotta voidaan selvittää, onko tietyllä osalla tai sarjalla yhteys koneen häiriötiloihin. Viesti lähetetään aina koneen tilan muuttuessa.

Viestin topic muodostuu seuraavasti: {factory}/{location}/{line}/{machine}/status.

Kuva 7. Koneen tilatietoviestin esimerkki.

```
{
  "timestamp": "2024-05-31T12:34:56Z", // Aikaleima RFC3339 tai ISO 8601
muodossa
  "machine_state": "maintenance", // Koneen tila (string)
shutdown/stopped/waiting/setup/maintenance/production
  "event_status" : "incoming", // Tilan status incoming = alkanut tai
outgoing = loppunut (string)
  "duration": 25, // Aika sekunteina kauan laite oli kyseisissä tilassa,
lisätään vain outgoing viestiin, muuten 0 (INT)
  "part_serial": "311331100592-1-26-3-U-4-1", // Sarjanumero tai koneen
generoima yksilöivä koodi jokaiselle kappaleelle (string)
  "comment": "anturin S13 vaihto" // Vapaamuotoinen kommentti
operaattorilta. Sisällytetään vain jos käytössä
}
```

Rakenne voi sisältää myös operaattorin laatiman vapaamuotoisen kommentin ja kuittauksen, joka voi tarkentaa tapahtuman taustoja.

4.1.3 Mittaus

Kolmas kuvassa 8 esitetty viesti välittää mittaustiedon jostakin seurattavasta suureesta, kuten lämpötilasta tai painetiedosta. Se kertoo, milloin mittaus on tehty, mikä tarkentava nimi anturilla on ja missä mittayksikössä arvo on ilmoitettu. Tämän lisäksi viestissä voidaan tarvittaessa ilmoittaa, mihin kappaleeseen mittaus liittyy, jolloin mittaustieto voidaan yhdistää suoraan kyseisen kappaleen työstöön tai laadunvalvontaan.

Viestin topic muodostuu seuraavasti:
{factory}/{location}/{line}/{machine}/measurements/{measurement}.

Kuva 8. Mittaustiedon viestirakenteen esimerkki.

```
{
  "timestamp": "2024-05-31T12:01:00Z", // Aikaleima RFC3339 tai ISO 8601
muodossa
  "value": 15.2, // Mittausarvo (real)
  "name": "S124", // Anturin tai mittauksen tarkentava nimi (string)
  "measurement_unit": "C", // Mittayksikkö (string)
  "part_serial": "311331100592-1-26-3-U-4-1", // Edellisen tehdyn tai
teossa olevan kappaleen sarjanumero. Jätetään pois jos ei tiedossa.
}
```

4.2 PLC ohjelmointi

Ohjelmamuutosten valmistelu aloitettiin varmistamalla, että Omron NJ/NX sarjan logiikoiden ohjelmointiin käytettävä Sysmac Studio projekti oli ajan tasalla yhdistämällä logiikkaan ja suorittamalla ohjelman synkronointi. Tämän jälkeen logiikan muistiin tallennettujen muuttujien

arvoista otettiin varmuuskopio. MQTT-kommunikoinnin mahdollistava kirjaston asennus tehtiin Omronin W625-ohjeen mukaisesti, minkä jälkeen Omronin malliprojektista kopioitiin perustoiminnot MQTT-tiedonsiirtoa varten (OMRON Corporation, 2021). Muutokset ladattiin koneen logiikkaan ja ohjelman toimivuus vahvistettiin toteamalla viestien saapuminen MQTT-palvelimelle MQTT Explorer -ohjelmalla.

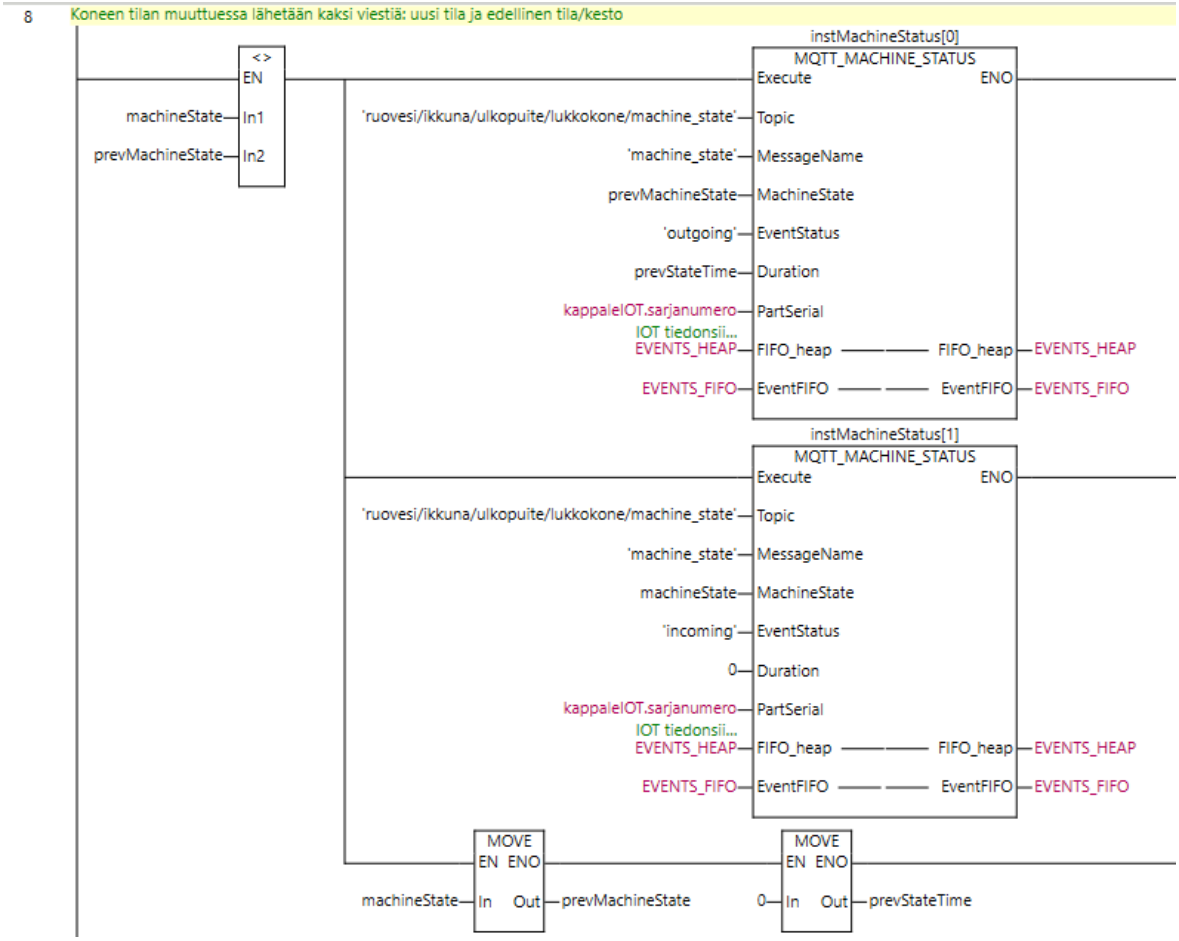
Ohjelmakirjaston toiminnan varmistamisen jälkeen MQTT-viestin muodostusta varten tehtiin funktiot, jotka muodostavat lähetettävän viestin JSON-muotoon. Kuvassa 9 on esitettynä MQTT_MACHINE_STATUS funktio, joka kokoaa koneen tilatiedot JSON-muotoiseen viestiin tämän lähetystä varten. Tämä funktio on toteutettu Structured Text (ST) -ohjelmointikielellä ja sen suunnittelussa on hyödynnetty Omronin malliprojektista löytynyttä esimerkkitoteutusta (OMRON Corporation, 2021).

Kuva 9. Koneen tilatietoviestin muodostaminen.

```
Clear(jsBuffer);

JSONObj(Object:=MessageName, Buffer:=jsBuffer);
  JSON_DATE(VariableName:='timestamp', VariableValue:=GetTime(),
Buffer:=jsBuffer);
  JSON_STRING(VariableName:='machine_state', VariableValue:=MachineState,
Buffer:=jsBuffer);
  JSON_STRING(VariableName:='event_status', VariableValue:=EventStatus,
Buffer:=jsBuffer);
  JSON_DINT(VariableName:='duration', VariableValue:=Duration,
Buffer:=jsBuffer);
  JSON_STRING(VariableName:='part_serial', VariableValue:=PartSerial,
Buffer:=jsBuffer);
JSONCloseObj(jsBuffer);

//Build message
stackEvent.Payload := JSONBuild(jsBuffer);

//Add topic
stackEvent.Topic := Topic;

//Push event to FIFO
StackPush(stackEvent,EventFIFO[0],StackSize,FIFO_heap);
```

Tämän jälkeen toteutettiin ohjelma, joka määrittelee koneen käyntitilan, häiriötilan ja työstössä olevan kappaleen tilan. Koneen tilatieto perustuu edellisten minuuttien tuotantomäärään. Kuvassa 10 on esitetty tikapuukaavio (LAD) tehty ohjelma, joka tallentaa koneen tilan muuttuessa JSON-viestin lähetyspuskuriin käyttämällä edellisessä kuvassa 9 esitettyä funktiota. Lähetyspuskurin ansiosta logiikan ja MQTT-palvelimen väliset lyhyet yhteyskatkokset eivät aiheuta lähetettävien tietojen menettämistä.

Kuva 10. Koneen tilatiedon lähetys.



Logiikan tarvitsemat tiedot kappaleiden työstöistä ja niiden paikoituksista saadaan solun lukkokoneelta EtherNet/IP-protokollalla ja vastaavasti lukkokone saa tiedot solussa olevalta PC:ltä, johon on asennettu Omronin kehittämä CX Supervisor SCADA-ohjelmisto. Operaattori lataa uutta sarjaa aloitettaessa verkkoasemalla olevan CSV-tiedoston, joka sisältää kaikki koneiden tarvitsemat tiedot kappaleiden valmistamiseen. Luetut tiedot tallennetaan CX Supervisorin sisäiseen muistiin, josta lukkokone kyselee niitä yksittäiselle kappaleelle kerrallaan ja kun kone kuittaa kappaleen valmistuneeksi, kuittaantuu tämä operaattorinäkymässä valmiiksi.

Jotta kaikki tiedonkeruussa tarvittavat tiedot olisivat saatavilla, molempien logiikoiden tiedonsiirtoon määriteltiin uudet tiedot. Muutos oli helppo toteuttaa, koska tuotetiedot oli määritetty typeProduct-nimiseen Struct-tietotyyppiin. Tämä tietotyyppi päivitettiin molempiin logiikoihin. Struct-tietotyypissä useammat muuttujat ryhmitellään yhdeksi kokonaisuudeksi, uudelleenkäytön helpottamiseksi. Kuvassa 11 on esitetty kyseinen tietotyyppi.

Kuva 11. Tietotyyppi kappaletiedoille.

typeProduct	STRUCT	NJ			
tyyppi	INT				1=Normi, 2=Termo, 3=Supertermo
pituus	INT				Puitteen pituus
kulmareikä	INT				0=Ei reikiä 1=Kulmareiät
aukipito	INT				0=Ei reikiä 1=2 reikää, 2=4 reikää, 3=Puitteen keskituki
aukipitoKat	INT				Aukipitoreiät: 1=Alkuun, 2=Loppuun
aukipitoSiirto	INT				Aukipidon siirto: 1=Normaali, 2=Alasaranoitu(reikien 26mm siirto keskemmälle puitetta)
vesireikä	INT				1=Vesireikä, termoon molempiin päihin
poistopaikka	INT				1 = Sarana, 2 = Lukko, 3 = Pöytä
sarja	STRING[20]				
tekonumero	STRING[10]				
osanTyyppi	STRING[20]				
sarjanumero	STRING[60]				Matriisikoodi

Tietotyypin päivittämisen jälkeen uudet tiedot lisättiin CX Supervisor projektista löytyvään skriptiin, joka suoritetaan aina, kun lukkokone pyytää uusia tietoja. Projekti tallennettiin runtime tiedostoon, joka suoritetaan valvomo PC:lle asennetulla CX Supervisor ohjelmistolla.

Logiikkaohjelman muutokset valmisteltiin niin, että niiden vaikutus koneen toimintaan olisi mahdollisimman vähäinen. Tämän tarkoituksena oli varmistaa, ettei virheet mittauksissa tai MQTT-tiedonsiirrossa voisi häiritä koneen muuta toimintaa.

Ohjelmamuutosten lataus logiikoihin toteutettiin työntekijöiden kahvitauon aikana, sillä Omron logiikat vaativat ohjelman suorittamisen pysäyttämisen isompien ohjelmamuutoksien latauksessa. Muutosten latauksen jälkeen havaittiin, ettei logiikoiden välinen tiedonsiirto toiminut toiseen suuntaan. Vika selvisi kuitenkin nopeasti: Sysmac Studion "EtherNet/IP Connection Settings" asetuksissa "Size (Byte)" arvo ei ollut päivittynyt automaattisesti uusien siirrettävien tietojen lisäyksen jälkeen. Tämän arvon korjaamisen jälkeen tiedonsiirto alkoi toimia ja solu saatiin takaisin käyttökuntoon ennen tauon päättymistä. MQTT-tiedonsiirron toiminta varmistettiin MQTT Explorer -ohjelman avulla.

Lopulliset pienemmät korjaukset ja hienosäädöt oli mahdollista tehdä logiikkaohjelmaan käyttämällä ”Online Edit” toimintoa, joka sallii pienemmät ohjelmamuutokset tuotannon aikana ilman ohjelman suorittamisen pysäyttämistä.

4.3 InfluxDB tietokanta

InfluxDB valikoitui opinnäytetyön tietokantaratkaisuksi aikaisempien positiivisten kokemusten perusteella ja koska se on suunniteltu erityisesti IoT-datan keräämiseen. Sen etuja

verrattuna perinteisiin SQL-tietokantoihin ovat yksinkertaisempi tietomallin suunnittelu sekä parempi suorituskyky suurten ja nopeasti muuttuvien tietomäärien käsittelyssä.

Tietokantarakenne pohjautuu MQTT-viestirakenteeseen, joka vähentää tarvetta esikäsitellä dataa ennen sen tallennusta tietokantaan.

Vaikka tietokannan suunnittelussa pyrittiin välttämään ylimääräisen tiedon tallentamista, haluttiin samalla varmistaa, että tietokannasta voidaan tehdä monipuolisia kyselyitä. Esimerkiksi analyysit tietyn tuotetyypin vaikutuksesta virheiden määrään tai koneen lämpötilaan tulee olla mahdollisia.

InfluxDB-tietokannan suorituskyvyn optimoinnissa keskeinen huomio on kardinaliteetin hallinta. Kardinaliteetti viittaa eri tunnisteavainten (tag keys) uniikkien arvojen määrään. Sijoittamalla muuttuvat tiedot, kuten sarjanumerot field key -kenttiin ja harvemmin muuttuvat tiedot tag key -kenttiin, kardinaliteetti on pienempi. Korkea kardinaliteetti voi heikentää tietokannan suorituskykyä, sillä se lisää muistinkulutusta ja tallennustilan tarvetta. InfluxDB V3-versiossa tämä ongelma on huomattavasti pienempi. (InfluxData Inc., n.d.-b)

Alustavissa laskelmissa kardinaliteetti saattoi nousta kymmeniin miljooniin, mikä ylittää pilvipalvelussa sallitun maksimimäärän, joka on miljoona. Tarkemmassa analyysissä kuitenkin ilmeni, että todellinen kardinaliteetti on huomattavasti pienempi. Esimerkiksi mittausten määräksi saatiin 159, ja huomioimalla kaikki eri tunnisteiden mahdollisuudet kardinaliteetti nousi vain noin 257 664:ään. Tämän lisäksi tagien yhdistelmät eivät vaikuta merkittävästi kardinaliteettiin, mikäli esimerkiksi tietty anturi toimii vain tietyssä tuotantolinjassa.

4.3.1 Tietokantarakenne

Tallennettaville tiedoille muodostui kaksi erillistä tietokantarakennetta, joista ensimmäistä käytettiin kappaleiden seurantaan eli siihen tallennettiin kappaleen tilatiedot. Liitteessä 4 on esimerkki tästä tietokantarakenteesta. Tämä tietokanta sisältää eränumeron (batch), tekonumeron (build_number), kappaleen tyypin (part_type), pituuden (part_length), sarjanumeron (part_serial) ja valmistusvaiheen keston (duration). Lisäksi lisätunnisteet (tag) kuten vuoro (shift), tehtaan sijainti (factory), valmistuslinja (line) ja laite (machine).

Toisen liitteen 5 esimerkin mukaista tietokantaa käytettiin koneen käynninseurantaan ja anturidatan keräykseen. Tietokanta sisältää mitattavan arvon nimen (field), mittaukseen

liittyvä lisätieto tai anturin nimi (name), mittayksikön (measurement_unit), sarjanumeron (part_serial) ja näiden lisäksi samat lisätunnisteet kuin ensimmäisessä tietokannassa.

Jatkokehityksessä rakenteen hyödyntäminen olisi myös mahdollista SQL-pohjaisessa tietokannassa. Erilliseen konerekisteriin voitaisiin siirtää tiedot, kuten factory, location, line ja machine, jotka tallennettaisiin omaan dimensiotauluunsa. Nämä tiedot korvattaisiin viittaamalla niitä machine_id-tunnuksella, joka toimisi kyseisen dimensiotaulun ensisijaisena avaimena (primary key).

Samanlainen ratkaisu olisi sovellettavissa myös kappaleseurannan tietojen käsittelyssä. Batch, build_number, part_type, part_count ja part_serial -tiedot voitaisiin siirtää erilliseen dimensiotauluun. Tässä tapauksessa part_serial toimisi dimensiotaulun ensisijaisena avaimena.

Dimensiotaulun voi ajatella olevan tietokantataulu, joka esimerkiksi tässä tapauksessa voi sisältää tietoja kuten koneen sijainnin, linjan ja tehtaan, jossa kone toimii. Tämä taulu ei itsessään sisällä tapahtumatietoja, mutta siihen viitataan viiteavainten (foreign keys) avulla tapahtumatiedot sisältävästä päätietotaulusta. Näin voidaan vähentää saman tiedon toistamista tietokannassa.

Ensisijainen avain (primary key) taas on sarake tai sarakkeiden yhdistelmä, mikä yksilöi jokaisen rivin tietokantataulussa. Tämä varmistaa, että jokainen rivi on uniikki. Dimensiotaulussa esimerkiksi machine_id tai part_serial toimii tällaisena ensisijaisena avaimena. Tämä auttaa pitämään tietokannan tiedot eheänä ja nopeuttaa tietokantahakuja.

4.3.2 Käyttöönotto

InfluxDB tietokannan hallinta tapahtui selainpohjaisesti kirjautumalla pilvipalveluun. Tiedonkeruun valmistelun aikana luotiin kaksi uutta bucketia eli taulua nimiltään “ruo_part_tracking” ja “ruo_machine_monitoring”. Luontivaiheessa asetettiin, ettei vanhoja tietoja poisteta koskaan, jotta kaikki historiallinen tieto olisi käytettävissä. Myöhemmin kun tiedonkeruu laajenee, automaattinen poisto voi tulla tarpeelliseksi säilyttämisestä aiheutuvien kulujen vähentämiseksi. Lisäksi Node-RED- ja Grafana-palveluille luotiin omat API-avaimet, jotka antavat tarvittavat käyttöoikeudet tietokantaan. Grafanalle annettiin vain lukuoikeudet, jotta varmistetaan ettei tietokantaan voi tehdä muutoksia virheellisillä kyselyillä. Nämä avaimet syötettiin Grafana ja Node-RED palveluihin.

Kun tiedonkeruu tietokantaan aloitettiin, Flux-kyselyiden testaaminen oli helpointa suorittaa “Data Explorer” -käyttöliittymässä. Tämä tarjoaa automaattisen syntaksintarkistuksen sekä ohjeet käytettävissä oleviin funktioihin sivuvalikosta. Kuvassa 12 on esimerkki Flux-kyselystä, joka hakee koneen käyntitilahistorian halutulta aikaväliltä.

Kuva 12. InfluxDB Flux-kysely koneen tilahistoriaa varten.

```
from(bucket: "ruo_machine_monitoring")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "machine_status")
  |> filter(fn: (r) => r["location"] == "ikkuna")
  |> filter(fn: (r) => r["line"] == "ulkopuite")
  |> filter(fn: (r) => r["machine"] == "lukkokone")
  |> filter(fn: (r) => r["event_status"] == "incoming")
  |> filter(fn: (r) => r["_field"] == "duration")
  |> filter(fn: (r) => exists r["machine_state"])
  |> keep(columns: ["_time", "machine_state"])
  |> group()
  |> sort(columns: ["_time"], desc: false)
```

Lisäksi yritettiin toteuttaa uudelleentehtyjen kappaleiden laskentaa suoraan Flux-kyselyssä. Kyselyssä laskettiin sarjanumerot, jotka esiintyivät useampaan kertaan. Tämä kysely osoittautui kuitenkin liian hitaaksi ja monimutkaiseksi, joten tämä päätettiin toteuttaa Node-RED ohjelmassa.

Ensimmäisten InfluxDB-tietokantakyselyiden testauksen jälkeen havaittiin tarve lisätä kappaleiden työstöaika lähetettävään MQTT viestiin, jotta sitä ei tarvitsisi määritellä aikaleimojen perusteella. Aikaleimojen perusteella tehtävä laskenta tietokantakyselyissä todettiin hitaaksi ja tarpeettoman monimutkaiseksi.

4.4 Node-RED ohjelmointi

Node-RED tarjoaa monia tapoja ohjelmoida, esimerkiksi no-code-lähestymistavan, jossa ohjelmointi tapahtuu kokonaan valmiilla nodeilla. Tässä työssä päädyttiin kuitenkin käyttämään enemmän JavaScript-koodiin perustuvia nodeja ohjelman yksinkertaistamiseksi ja muokattavuuden parantamiseksi. Node-RED Javascript pohjaisten funktioiden ohjelmakoodin toteuttamisessa käytettiin apuna ChatGPT 4o tekoälytyökalua.

InfluxDB tietokanta määritettiin asentamalla ensin Node-RED Palette asetusvalikosta ”node-red-contrib-influxdb” kirjasto, tämän jälkeen ”influxdb batch” node raahattiin työskentelytilaan. Tämän noden asetuksiin määritettiin InfluxDB tietokannan osoite, aikaisemmin määritetty token, organisaatio ja bucket eli tietokanta, johon kirjoitetaan. MQTT

määrittäminen onnistui käyttämällä Node-RED esiasennettua ”mqtt in” nodea, johon määritettiin MQTT-broker IP osoite ja luettava topic.

Koska uudelleentehtyjen kappaleiden tunnistamista ei ollut järkevää toteuttaa suoraan logiikassa tai tietokantakyselyissä niin tämä takia uudelleentehdyn kappaleen tunnistus toteutettiin Node-REDin avulla seuraavalla tavalla: Kun MQTT-solmu vastaanottaa viestin, muodostetaan viestistä saatujen tietojen perusteella InfluxDB-tietokantakysely. Kyselyllä haetaan kyseisellä sarjanumerolla aikaisemmin samasta koneesta tallennettuja tietoja. Jos tietoja ei löydy, voidaan olettaa kappaleen olevan ensimmäistä kertaa valmistuksessa. Jos tietoja taas löytyy, kyseessä on uudelleentehty kappale. InfluxDB-pilvipalvelua käytettäessä kyselyiden määrän lisääntyminen tulee nostamaan kuukausimaksua noin viidellä eurolla per laite, jossa kyselyitä tehdään tällä tavalla. Tästä syystä laitekannan kasvaessa voi olla tarpeen käyttää esimerkiksi rinnakkaista paikallista InfluxDB-asennusta kustannusten pienentämiseksi.

Liitteessä 1 on esitetty lopullinen Node-RED-ohjelmavirtaus eli flow, joka siirtää MQTT:stä luetut tiedot InfluxDB-tietokantaan. Tämä virtaus koostuu kolmesta pääkomponentista: kappaleen tilan seurannasta, koneen tilan seurannasta ja vikojen seurannasta. Ensimmäinen rivi kuvaa "part_status" solmua, joka vastaanottaa osan tilan. Tämän jälkeen muodostaa tietokantakysely liitteen 2 "Create InfluxDB query" funktion avulla, jolla tarkistetaan, onko kappale uudelleentehty. Lopulta liitteen 3 "Create part payload" funktio käsittelee datan oikeaan muotoon ennen tallennusta. Tiedot tallennetaan tietokannan tauluun nimeltä "ruo_part_tracking".

Liitteen 1 ohjelmavirtauksen keskimmäisessä rivissä solmu vastaa koneen tilan monitoroinnista. Vastaanotettu tieto käsitellään "Create machine status payload" funktiolla, joka on yksinkertaistettu versio liitteen 3 funktiosta. Tiedot tallennetaan tietokannan "ruo_machine_monitoring" tauluun. Alimmassa rivissä on solmu, joka käsittelee koneen vikatilat "Create fault payload" funktiolla ja tallentaa ne samaan tietokantaan.

Liitteen 3 "Create part payload" funktiossa on myös määritetty tietokantaan lisättävä tieto, mikä kolmesta eri vuorosta on kyseessä helpottamaan tietokantakyselyiden tekemistä. Tehtailla vuorot ovat vakioutu seuraavasti: aamu 06–14, ilta 14–22 ja yö 22–06.

Lisäksi sarjojen kokonaiskappalemäärien hakemiseen hyödynnettiin ERP-järjestelmän rajapintaa, joka mahdollistaa tietojen hakemisen ERP-järjestelmän SQL-tietokannasta ilman suoraan yhteyttä SQL-tietokantaan. Rajapintakysely tehdään saadun ”batch”-tiedon

perusteella, vastauksena saadaan sarjan kokonaiskappalemäärät. Kyselyn jälkeen sarjan nimi tallennetaan Node-REDin välimuistiin, jolloin kysely voidaan ohittaa, jos tieto on jo tallennettuna. Tämä estää tarpeettomat toistuvat kyselyt samalle sarjalle.

Node-RED ohjelmaan tehtyjen muutoksien testaus onnistui kopioimalla MQTT viestin sisältö inject noden ”msg.payload” kenttään JSON muodossa ja asettamalla ”msg.topic” kenttään haluttu MQTT topic. Virtauksen loppuun asetettu debug node tulostaa viestit debug viestikenttään tarkistusta varten. Kun ohjelman toimivuus todettiin halutunlaiseksi, aloitettiin tallennus tietokantaan.

4.5 Visualisointi Grafanalla

Grafana näkymien suunnitteluun ei laadittu tarkempaa toteutussuunnitelmaa. Sen sijaan päätettiin hyödyntää aiemmin toteutettuja näkymiä, joita jatkokehitettiin kartoituksessa esiin tulleiden tarpeiden perusteella. Näkymien suunnittelu ja toteuttaminen oli sujuvampaa, kun tarvittava tieto oli saatavilla tietokannasta ja se voitiin visualisoida konkreettisesti, jolloin lopputulos oli heti nähtävillä.

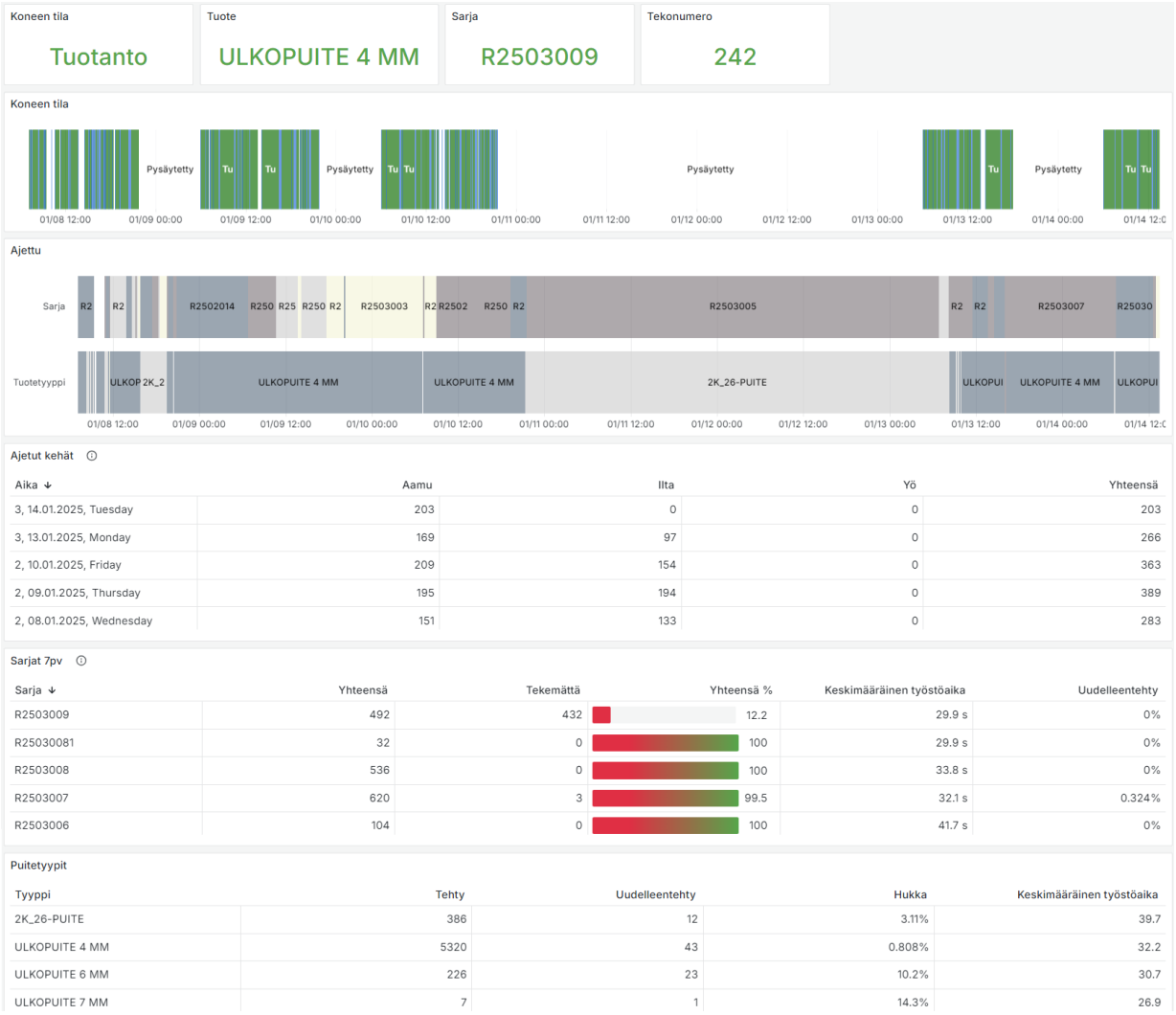
Vaikka tämän työn painopiste oli yksittäisen koneen visualisoinnissa, varmistettiin, että Grafanassa olisi kaikki tarvittavat ominaisuudet kattamaan myös muita koneita ja toimipisteitä, kuten käyttöoikeuksien hallinta. Azure Grafanan osalta käyttäjän käyttöoikeudet lisätään Azuren hallinnan kautta, jonka jälkeen käyttäjä pääsee kirjautumaan Active Directory eli AD-kirjautumisella, mikä on Microsoftin käyttäjien ja resurssien hallintaratkaisu. AD-kirjautumisen ansiosta erillisiä käyttäjätunnuksia ei tarvita. Ensimmäisen kirjautumisen jälkeen käyttäjälle voidaan lisätä Grafanan hallintapaneelista tarvittavat ryhmät ja käyttöoikeudet. Käyttöoikeuksien hallinta mahdollistaa katselu- ja muokkausoikeuksien kohdistamisen joustavasti, esimerkiksi tehdas-, linja- tai konekohtaisesti. Tämä on tärkeää, kun tarkoituksena on laajentaa järjestelmää useampiin toimipisteisiin tai erilaisten käyttäjäryhmien lisääntyessä.

Grafanaan toteutettiin kolme näkymää, jotka perustuivat kartoituksessa havaittuihin tarpeisiin ja aiempiin toteutuksiin. Näkymien tavoitteena oli tarjota selkeä ja helposti ymmärrettävä kokonaisuus.

Kuvassa 13 on esitettynä ensimmäinen näkymä, jossa on nähtävillä koneen reaaliaikaiset tilatiedot, johon sisältyivät ajossa olevan tuotteen tyyppi, sarja ja tekonumero. Seuraavaksi näkymässä on kolme aikajanaa, jotka kuvaavat koneen tilaa, ajettuja sarjoja ja tuotetyyppejä.

Näiden jälkeen on nähtävillä valmistettujen kappaleiden määrät päiväkohtaisesti ja vuorotason tarkkuudella. Lisäksi sarjakohtaisesti on nähtävillä sarjan puitteiden kokonaismäärä, valmistuneiden kappaleiden määrä, sarjan keskimääräinen työstöaika ja uudelleen tehtyjen kappaleiden määrä. Lopuksi näkymässä on vielä tuotetyyppikohtaiset kappalemäärät, näiden hukka ja keskimääräinen työstöaika.

Kuva 13. Koneen päänäkymä Grafanassa.



Toinen kuvan 14 näkymä keskittyy yleisimpiin KPI-mittareihin viikkotasolla. Siinä on nähtävillä käytettävyyttä, häiriöaikaa, seisonta-aikaa, ajotehoa ja tuottavuutta kuvaavat mittarit. Lisäksi häiriöihin liittyen on ajalliset, että määrälliset näkymät, joilla on mahdollista havaita seurattujen häiriöiden määrä ja niiden ajallinen kesto. Tästä on mahdollista havaita koneen toiminnassa pidemmällä aikavälillä tapahtuvia muutoksia tehokkuudessa tai häiriöissä.

Kuva 14. Koneen KPI näkymä Grafanassa.

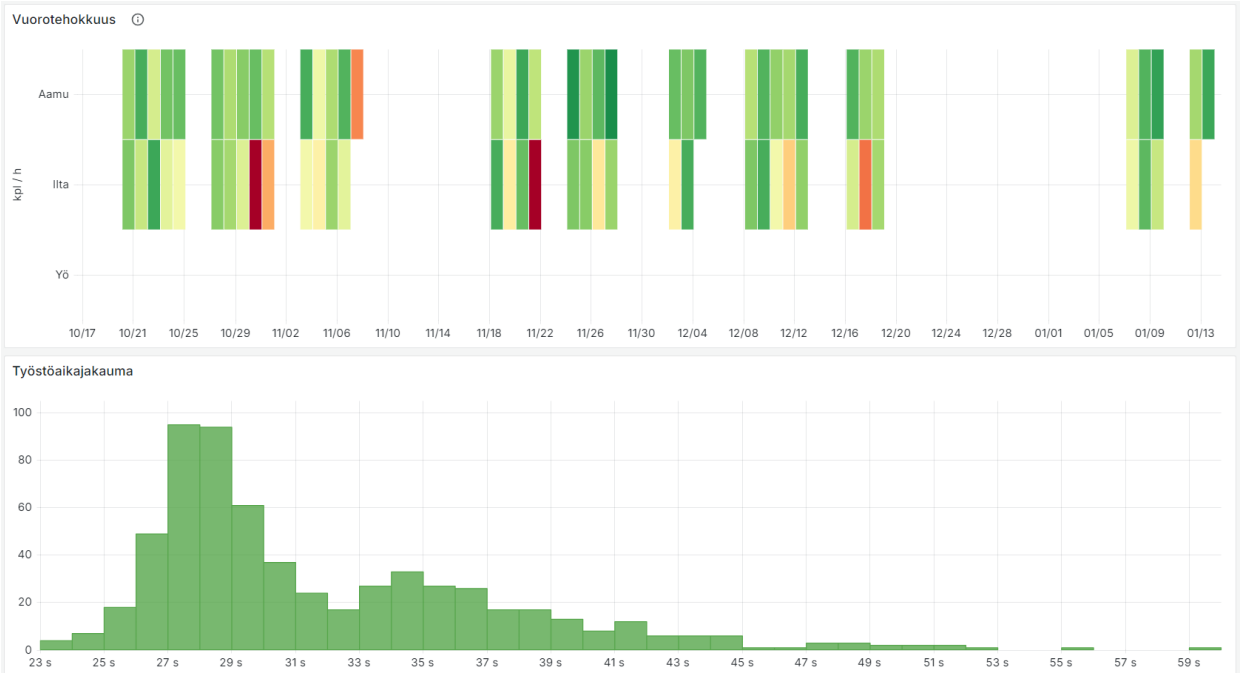
KPI näkymässä on käytetty seuraavia Mantec raportista sovellettuja laskukaavoja. (Mantec Oy, henkilökohtainen tiedonanto, 2022)

1. Käytettävyys = ajoaika / (ajoaika + häiriöaika).
2. Häiriö % = häiriöaika / (ajoaika + häiriöaika).
3. Ajoteho kpl/h = (ajetut + uudelleenajetut) / (ajoaika + häiriöaika).
4. Linjan tuottavuus kpl/h = ajetut / (ajoaika + häiriöaika).
5. Puitehukka (pituus) % = uudelleenajetut pituus / (ajetut pituus + uudelleenajetut pituus).
6. Aikahukka (kpl) % = uudelleenajetut kpl / (ajetut kpl + uudelleenajetut kpl).

Joidenkin KPI-mittareiden tueksi tarvittaisiin tieto koneen suunnitellusta käyntiajasta, koska kaikkia koneita ei käytetä säännöllisesti yhtä suurta määrä tunteja vuorokaudessa. Tämän tiedon tallentaminen edellyttäisi joko työnjohdolta käsin syöttöä tai erillistä järjestelmää, johon tiedot voitaisiin tallentaa.

Kolmannen näkymän tarkoituksena oli esitellä mahdollisimman monipuolisesti erilaisia ominaisuuksia, joita järjestelmä tarjosi. Näkymässä on esimerkiksi lämpökartta vuorojen tehokkuuksista, jossa tehokkaimmat vuorot korostuvat vihreän sävyillä ja heikommat punaisilla. Lisäksi mukana on histogrammit, jotka kuvaavat työstöaikojen ja kappaleiden pituuksien jakautumista. XY-kaavion avulla esitettiin kappaleen pituuden vaikutusta työstöaikaan. Kuvassa 15 esillä näistä vuorotehokkuuden lämpökartta ja työstöaika histogrammi.

Kuva 15. Osa lisätietonäkymästä Grafanassa.



Näkymien toteutuksessa otettiin huomioon tietokantakyselyiden määrä ja niiden nopeus. Tietokantakyselyiden määrän hallinta on tärkeää erityisesti pilvipalveluita käytettäessä, koska näiden hinnoittelu perustuu usein tehtyjen kyselyiden tai käytetyn datan määrään. Grafana tarjoaa mahdollisuuden käyttää toista näkymää tietolähteenä, jolloin samoja tietoja ei tarvitse hakea useaan kertaan eri näkymiä varten. (Grafana Labs, n.d.)

4.6 Testaus ja käyttöönotto

Käyttöönoton aikana havaittiin useita pieniä virheitä, jotka liittyivät pääasiassa PLC-ohjelmaan. Esimerkiksi työstöjen kestoajat kirjautuivat ajoittain väärille kappaleille, erityisesti tilanteissa, joissa kone käsitteli useampia kappaleita samanaikaisesti. Tämä johtui siitä, ettei käsittelyaika aina kulkenut oikean kappaleen mukana. Ongelman ratkaisemiseksi

työstöaikojen hallinta liitettiin koneen sisäiseen FIFO-muistiin, joka käsittelee ensimmäisenä saapuneet tiedot ensin.

Lisäksi MQTT-tiedonsiirtorakenteeseen tehtiin muutoksia selkeyden parantamiseksi. Esimerkiksi ”status”-kenttä nimettiin uudelleen ”machine_state”-kentäksi, jotta nimi kuvaisi tarkemmin mitä tilaa tällä tarkoitetaan.

Korjausten ja muutosten valmistuttua testikäytössä ollut InfluxDB-tietokanta tyhjennettiin, minkä jälkeen tiedonkeruu aloitettiin pysyvään tietokantaan. Tällä varmistettiin, ettei tietokantaan jäänyt virheellisiä tietoja.

Kerättyjen tietojen paikkansapitävyyden varmistamiseksi hyödynnettiin aikaisemmasta tiedonkeruutoteutuksesta kerättyä dataa. Tässä toteutuksessa tiedot oli jo varmistettu vertaamalla tuloksia operaattoreiden ja työnjohdon tekemiin käsikirjauksiin. Osaa uusista tiedoista verrattiin koneelle syötettäviin CSV-tiedostoihin sekä seuraamalla koneen toimintaa. Kaikkia tietoja ei kuitenkaan ehditty varmistamaan työn aikataulun takia. Ennen laajempaa käyttöönottoa on tärkeää tehdä tarkempi validointi kerätylle datalle, tietokantakyselyille ja näkymille.

Käyttäjäpalautteen kerääminen oli tärkeä osa työtä. Saatu palaute oli positiivista, minkä vuoksi järjestelmän laajentamista toivottiin useampaan koneeseen. Lisäksi esitettiin toive, että myös operaattorit saisivat oman näkymän järjestelmään. Saatua palautetta hyödynnetään järjestelmän jatkokehityksessä.

5 Tulosten hyödyntäminen ja jatkokehitys

Työstä saadun konkreettisen hyödyn mittaaminen on haastavaa, koska potentiaaliset vaikutukset ovat laajat. Näihin kuuluvat muun muassa työajan säästö, parempi laatu, reklamaatioiden väheneminen sekä materiaalihukan minimointi. Vaikka näitä hyötyjä ei ole helppo mitata suoraan, niillä on selkeästi suuri merkitys tuotannon tehostamisessa ja kustannusten vähentämisessä.

Erityisesti kartoituksen yhteydessä tehty kooste kerättävistä tiedoista on osoittautunut erittäin arvokkaaksi. Sen avulla on jo voitu kohdentaa tiedonkeruun jatkokehitystä niihin asioihin, joilla on suurin merkitys. Lisäksi kooste helpottaa datankeruun tavoitteiden esittämistä ulkopuolisille toimijoille, kun voimme tarkemmin esittää mitä järjestelmältä odotetaan.

Työn yhteydessä toteutetun tiedonkeruun laajentamisesta ja jatkokehittämisestä on käyty keskusteluja useiden toimittajien kanssa. Suurimmaksi potentiaaliksi tunnistettiin kappaleiden seurannan laajentaminen, sillä sen avulla saavutettaisiin suurimmat hyödyt. Kappaleiden seuranta mahdollistaisi esimerkiksi tuotteiden tarkemman hinnoittelun, kun tiedon avulla voitaisiin paremmin arvioida eri tuotetyyppien vaikutuksia valmistusaikaan ja materiaalihukkaan. Lisäksi tuotannon valmistusjärjestyksen optimointi nähtiin tärkeänä osana tuotannon tehostamista.

Ohjelmistotoimittajat pitivät työn aikana toteutettua MQTT-pohjaista tiedonsiirtoa toimivana ratkaisuna, sillä sen avulla tiedon siirtäminen useampaan järjestelmään on yksinkertaista. Yksi toimittajista ryhtyi kartoittamaan edullista ja siirrettävää ratkaisua sarjanumeroiden lukemiseen tuotantolinjojen eri vaiheista sekä tämän tiedon välittämiseen MQTT-palvelimelle. Ennen järjestelmän laajentamista olisi kuitenkin suositeltavaa varmistaa sen skaalautuvuus suuremmalle laitemäärälle esimerkiksi simuloidussa testiympäristössä.

Datankeruujärjestelmän laajentamiseksi ja kehittämiseksi tunnistettiin useita mahdollisia jatkotoimenpiteitä:

1. Grafana näkymien, mittareiden ja analytiikan kehittäminen. Lisäksi Grafanan automaattisien sähköpostiin lähetettävien hälytyksien ja raporttien käyttöönotto.
2. PLC-kirjastojen, Node-RED-ohjelman ja Grafana näkymien kehittäminen niin, että uusien koneiden lisäykset onnistuisivat mahdollisimman yksinkertaisesti.
3. Konenäön hyödyntäminen tuotteen laadunvalvonnassa.
4. Koneiden tilatietojen ja häiriöiden tarkempi määrittely, kuten milloin kone on käynnissä tai pysähdytettynä. Tämä mahdollistaisi koneiden välisen vertailun.
5. Tekoälyn hyödyntäminen ennakoivassa kunnossapidossa kuten tekoälyn soveltaminen lämpötilamittausten analysoinnissa.
6. Tietokantaratkaisun jatkokehittäminen. Mahdollisesti jäsennetyn tai siivotun datan lähettäminen SQL- tai muuhun vastaavaan tietokantaan, joka tukisi laajemmin käytettyä Microsoft Power BI visualisointijärjestelmää. Power BI sopisi erityisesti pidemmältä aikaväliltä kerätyn datan analysointiin.

Työn loppuvaiheilla datankeruujärjestelmää päästiin jo laajentamaan isompaan linjakokonaisuuteen. Tämä linja on ollut Ruoveden tehtaalla pidempään pullonkaulana, joten datankeruusta saatavan hyödyn potentiaali on suurempi. Työstä saaduilla kokemuksilla ja määrityksillä pystyttiin tähän linjaan toteuttamaan vastaava datankeruu vähäisellä työmäärällä. Linjaston datankeruussa voitiin käyttää samoja määrityksiä tiedonsiirrolle ja

tietokantarakenteelle, lisäksi Grafana näkymät oli mahdollista kopioida pienillä muutoksilla lisätylle linjastolle.

6 Yhteenveto ja johtopäätökset

Opinnäytetyön tavoitteena oli tutkia teollisuuden datankeruun, tallennuksen ja visualisoinnin mahdollisuuksia tuotantoprosessien tehostamiseksi, sekä kehittää ja toteuttaa pilottiratkaisu yksittäiselle tuotannon koneelle. Suunnitteluvaiheessa kartoitettiin yrityksen tarpeita sekä käyttökohteita kerättävälle datalle. Kartoituksen perusteella päädyttiin MQTT-pohjaiseen tiedonsiirtoon ja aikaisempien kokemuksien perusteella hyödynnettiin edelleen InfluxDB-aikasarjatietokantaa, Node-RED ohjelmointiympäristöä ja Grafana-visualisointia. Työn aikana huomattiin, että erityisesti yksilöityjen aikaleimojen ja sarjanumeroiden kerääminen tarjoaa selvästi enemmän hyötyjä kuin alun perin osattiin arvioida. Näiden avulla pystytään yhdistämään esimerkiksi valmistusajat, laatuongelmat ja koneiden tilamuutokset entistä paremmin, mikä luo pohjan tuotannon optimoinnille, ennakoivalle kunnossapidolle sekä materiaalihukan vähentämiselle.

Työn merkittävin saavutus oli pilottijärjestelmän toteuttaminen. Ratkaisu onnistuttiin toteuttamaan siten, että aikaisemmassa datankeruun toteutuksessa havaitut puutteet tiedonsiirrossa korjattiin. Ratkaisu on helposti laajennettavissa useammille koneille, mikä todennettiin työn loppuvaiheessa tehdyssä laajennuksessa.

Työ osoittautui laajaksi ja haastavaksi erityisesti siksi, että toteutus tehtiin ilman ulkopuolista apua. Suunnittelun haastavimmaksi osuudeksi osoittautui MQTT-viestien ja tietokantarakenteiden suunnittelu, koska nämä toimivat pohjana koko tiedonkeruulle. Rakenteiden suunnittelussa piti huomioida mahdolliset tulevaisuuden tarpeet, myöhemmin lisättävät koneet sekä tietokantakyselyiden muodostus. Suunnittelussa tehdyt päätökset voivat vaikuttaa huomattavasti tiedonsiirrossa ja raportoinnissa käytettävien ohjelmien monimutkaisuuteen. Aikaisempi kokemus tiedonkeruusta auttoi hahmottamaan, miten rakenne kannattaa toteuttaa, jotta tietokantarakenne ja kyselyt pysyisivät mahdollisimman yksinkertaisina.

Haasteita ja rajoituksia ilmeni erityisesti tiedonsiirron yhteensopivuudessa vanhempien PLC-laitteiden kanssa, sillä näiden firmware versiot eivät tarjonneet tarvittavia ominaisuuksia. Lisäksi Flux-kyselykieli InfluxDB:ssä rajoitti Power BI -raportoinnin kaltaisia ratkaisuja, mikä voi jatkossa edellyttää SQL-pohjaisen tietokannan hyödyntämistä, jos InfluxDB ei saa

kyseistä tukea tulevaisuudessa. Puutteelliseksi jäi myös perusteellinen vertailu erilaisten tietokantaratkaisujen ja järjestelmäarkkitehtuurien välillä, mikä voisi olla tarpeen ennen laajempaa käyttöönottoa. Näistä huolimatta ratkaisu muodostaa vankan perustan, jota voidaan laajentaa ja hyödyntää koneoppimisen, tekoälysovellusten ja entistä kattavamman raportoinnin kehittämisessä.

Järjestelmän laajentamiselle on useita vaihtoehtoja, joten kehitysvaihtoehtojen huolellinen kartoitus ja priorisointi ovat tärkeitä, jotta järjestelmä voidaan mukauttaa tuleviin tarpeisiin tehokkaasti ja kustannustehokkaasti. Pitkän aikavälin hyötyjen varmistamiseksi on tärkeää jatkaa yhteistyötä eri toimittajien kanssa, jotta kehitystyö pysyisi jatkuvana.

Työn aikana toteutettu tiedonkeruujärjestelmä ja siihen liittyvä kehitystyö tarjoavat merkittäviä mahdollisuuksia tuotantoprosessin tehostamiseen. Työn myötä saavutetut tulokset antavat yritykselle pohjan laajemman tiedonkeruujärjestelmän toteuttamiselle, jonka avulla voidaan parantaa tuottavuutta, tehostaa laadunhallintaa ja saavuttaa kustannussäästöjä.

Lähteet

Alisa, A.-H. (22.12.2023). *Simulointi- ja optimointiteknologiat prosessiteollisuudessa*. Julkaisut @SeAMK. https://lehti.seamk.fi/alykkaat-ja-energiatehokkaat-jarjestelmat/prosessiteollisuuden-tuotantoprosessien-simulointi-ja-optimointiteknologiat/

Deuil, D., Nübling, S., & Endres, F. (2007). *Competence Guide Direct Part Marking*. SICK AG. https://cdn.sick.com/media/docs/1/01/101/special_information_competence_guide_direct_part_marking_en_im0058101.pdf

Gontcharov, D., & Theocharis, J. (1.7.2024). *Industrial IoT visualization: Why United Manufacturing Hub chose Grafana to power its IIoT platform*. Grafana Labs. https://grafana.com/blog/2024/07/01/industrial-iot-visualization-why-united-manufacturing-hub-chose-grafana-to-power-its-iiot-platform/

Grafana Labs. (n.d.). *Share query results with another panel | Grafana documentation*. Grafana Labs. Haettu 29.11.2024 osoitteesta https://grafana.com/docs/grafana/latest/panels-visualizations/query-transform-data/share-query/

Gram, L. (4.9.2019). *How We Differentiate Grafana Enterprise from Open Source Grafana*. Grafana Labs. https://grafana.com/blog/2019/09/04/how-we-differentiate-grafana-enterprise-from-open-source-grafana/

HiveMQ Team. (20.2.2024). *MQTT Topics, Wildcards, & Best Practices – MQTT Essentials: Part 5*. https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

InfluxData Inc. (n.d.-a). *InfluxDB schema design | InfluxDB Cloud (TSM) Documentation*. Haettu 5.6.2024 osoitteesta https://docs.influxdata.com/influxdb/cloud/write-data/best-practices/schema-design/

InfluxData Inc. (n.d.-b). *Resolve high series cardinality | InfluxDB Cloud (TSM) Documentation*. Haettu 4.10.2024 osoitteesta https://docs.influxdata.com/influxdb/cloud/write-data/best-practices/resolve-high-cardinality/

InfluxData Inc. (n.d.-c). *The future of Flux | Flux Documentation*. Haettu 4.10.2024 osoitteesta https://docs.influxdata.com/flux/v0/future-of-flux/

King Ho, A. (30.6.2021). *How MQTT is Used for Industrial Automation—Technical Articles*. https://control.com/technical-articles/how-mqtt-is-used-for-industrial-automation/

Mahler, C. (20.9.2022). *Relational Databases vs Time Series Databases*. InfluxData. https://www.influxdata.com/blog/relational-databases-vs-time-series-databases/

MetricFire. (30.11.2023). *Grafana vs. Power BI*. https://www.metricfire.com/blog/grafana-vs-power-bi/

MQTT.org. (n.d.). *MQTT FAQ*. Haettu 23.11.2024 osoitteesta https://mqtt.org/faq/

Muelaner, J. (5.12.2021). *The Role of PLCs in Industrial Control and Test and Measurement*. DigiKey. https://www.digikey.com/en/articles/the-role-of-plcs-in-industrial-control-and-test-and-measurement

OMRON Corporation. (2021). *Sysmac Library for User's Manual MQTT Communications Library*. https://files.omron.eu/downloads/latest/manual/en/w625_sysmac_library_for_mqtt_communications_library_users_manual_en.pdf?v=1

OpenJS Foundation & Contributors. (n.d.-a). *About Node.js*. Haettu 15.12.2024 osoitteesta https://nodejs.org/en/about

OpenJS Foundation & Contributors. (n.d.-b). *About: Node-RED*. Node-RED. Haettu 23.11.2024 osoitteesta https://nodered.org/about/

OpenJS Foundation & Contributors. (n.d.-c). *Flows: Node-RED*. Node-RED. Haettu 23.11.2024 osoitteesta https://nodered.org/docs/user-guide/editor/workspace/flows

OpenJS Foundation & Contributors. (n.d.-d). *The Core Nodes: Node-RED*. Haettu 23.11.2024 osoitteesta https://nodered.org/docs/user-guide/nodes

Pihla Group Oy. (n.d.). *Meistä*. Pihlagroup. Haettu 28.12.2024 osoitteesta https://pihlagroup.fi/meista/

Pinja. (2022). *Opas—Mittaroi OEE-tunnuslukua oikein—Päivitetty 2022*. https://blog.pinja.com/hubfs/Pinja/Guides/Opas%20-%20Mittaroi%20OEE-tunnuslukua%20oikein%20-%20pa%CC%88ivitetty%202022.pdf

Pinja. (27.10.2023). *Älykkään tuotannon voima: Tekoälyn mahdollisuudet teollisessa ympäristössä*. https://blog.pinja.com/fi/alykkaan-tuotannon-voima-tekoalyn-mahdollisuudet-teollisessa-ymparistossa

Simola, N. (2019). *Omavalmisteosien jäljitettävyyden kehittäminen valmistavassa teollisuudessa*. https://urn.fi/URN:NBN:fi:tuni-201911055754

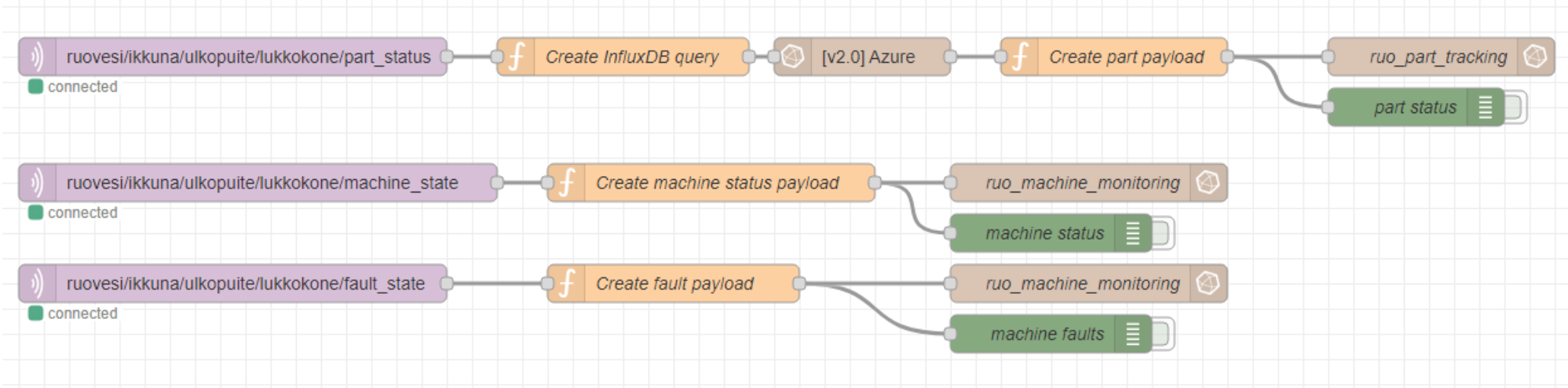
Tapia, E., Sastoque-Pinilla, L., Lopez-Novoa, U., Bediaga, I., & López de Lacalle, N. (2023). Assessing Industrial Communication Protocols to Bridge the Gap between Machine Tools and Software Monitoring. *Sensors (Basel, Switzerland)*, *23*(12). https://doi.org/10.3390/s23125694

Thamatam, V. K. (1.9.2022). *The What, Why, and How of Time Series Databases*. InfluxData. https://www.influxdata.com/blog/what-why-how-time-series-databases/

Tremblay, K. (27.6.2019). *Data Matrix Codes vs. QR Codes – What is the Difference?* Laserax. https://www.laserax.com/blog/data-matrix-vs-qr-codes

VTT. (n.d.). *Älykäs ennakoiva kunnossapito*. Haettu 27.12.2024 osoitteesta https://www.vttresearch.com/fi/palvelut/alykas-ennakoiva-kunnossapito

Liite 1. Lopullinen Node-RED virtaus



Liite 2. Node-RED funktio ”Create InfluxDB query”

```
// Save the original payload for use after the InfluxDB query
msg.originalPayload = RED.util.cloneMessage(msg.payload);

// Extract the components from msg.topic in the format
"factory/location/line/machine"
let topicParts = msg.topic.split("/");
let location = topicParts[1];
let line = topicParts[2];
let machine = topicParts[3];

let partSerial = msg.payload.part_status.part_serial;

let bucket = "ruo_part_tracking";

// Prepare the query for the InfluxDB node
msg.query = `
from(bucket: "${bucket}")
  |> range(start: -30d)
  |> filter(fn: (r) => r["_measurement"] == "part_status")
  |> filter(fn: (r) => r["_field"] == "part_serial")
  |> filter(fn: (r) => r["location"] == "${location}")
  |> filter(fn: (r) => r["line"] == "${line}")
  |> filter(fn: (r) => r["machine"] == "${machine}")
  |> filter(fn: (r) => r["_value"] == "${partSerial}")
  |> limit(n: 1)
`;

return msg;
```

Liite 3. Node-RED funktio ”Create part payload”

```
// Get original part_status from MQTT message
const partStatus = msg.originalPayload.part_status;

// Extract the components from msg.topic in the format
"factory/location/line/machine"
let topicParts = msg.topic.split("/");
let location = topicParts[1];
let line = topicParts[2];
let machine = topicParts[3];

// Function to determine shift based on timestamp
function getShift(timestamp) {
    const hours = timestamp.getHours();

    if (hours >= 6 && hours < 14) {
        return "morning";
    } else if (hours >= 14 && hours < 22) {
        return "evening";
    } else {
        return "night";
    }
}

// If part serial query returned data from database then part is redone
if (msg.payload && msg.payload.length > 0) {
    partStatus.status = "redone";
}

// Prepare the output payload
msg.payload = [
    {
        measurement: "part_status",
        fields: {
            batch: partStatus.batch,
            build_number: partStatus.build_number,
            part_serial: partStatus.part_serial,
            part_count: partStatus.part_count,
            part_length: partStatus.part_length,
            duration: partStatus.duration
        },
        tags: {
            location: location,
            line:  line,
            machine:  machine,
            part_type: partStatus.part_type,
            status: partStatus.status,
            shift: getShift(new Date())
        },
        timestamp: new Date()
    }
];

return msg;
```

Liite 4. Kappaleiden seurannan tietokantarakennemalli

time	batch (_field)	build_number (_field)	part_type (_field)	part_length (_field)	part_serial (_field)	part_count (_field)	duration (_field)	shift (tag)	status (tag)	factory (tag)	location (tag)	line (tag)	machine (tag)
2024-05-31T12:00:00Z	R2423001	13	ULKOPUITE 4 MM	1320	31133100592-1-26-3-U-13-1	1	0	morning	started	ruovesi	ikkuna	ulkopuite	lukkokone
2024-05-31T12:05:00Z	R2423001	13	ULKOPUITE 4 MM	1320	31133100592-1-26-3-U-13-1	1	25	morning	done	ruovesi	ikkuna	ulkopuite	lukkokone
2024-05-31T12:10:00Z	R2423001	14	2K_26-PUITE	560	31133100592-1-26-3-U-14-2	1	0	evening	started	ruovesi	ikkuna	ulkopuite	lukkokone
2024-05-31T14:15:00Z	R2423001	14	2K_26-PUITE	560	31133100592-1-26-3-U-14-2	1	35	evening	failed	ruovesi	ikkuna	ulkopuite	lukkokone
2024-05-31T23:15:00Z	R2423001	14	2K_26-PUITE	560	31133100592-1-26-3-U-14-2	1	36	night	rerun	ruovesi	ikkuna	ulkopuite	lukkokone

Liite 5. Koneen käynninseurannan ja mittauksien tietokantarakennemalli

time	_measurement	_field name	_field data	name (tag)	measurement_unit (tag)	part_serial (_field)	factory (tag)	location (tag)	line (tag)	machine (tag)
2024-05-31T12:00:00Z	machine_status	running	0	incoming	s	31133100592-1-26-3-K-13-1	ruovesi	ikkuna	karmilinja	tappikone
2024-05-31T13:00:00Z	machine_status	running	3600	outgoing	s	31133100592-1-26-3-K-13-1	ruovesi	ikkuna	karmilinja	tappikone
2024-05-31T13:00:00Z	machine_status	stopped	0	incoming	s	31133100592-1-26-3-K-13-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T13:05:00Z	machine_status	slow_production	0	incoming	s	31133100592-1-26-3-K-13-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T13:15:00Z	machine_status	running	0	incoming	s	31133100592-1-26-3-K-13-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T13:30:00Z	machine_status	running	1800	outgoing	s	31133100592-1-26-3-K-14-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T13:30:00Z	machine_faults	machine_state	syottohairio	outgoing		31133100592-1-26-3-K-14-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T13:30:00Z	machine_faults	duration	2500	outgoing	s	31133100592-1-26-3-K-14-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T12:00:00Z	machine_measurements	temperature	75,3	S123	C	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	venttiilikone
2024-05-31T12:00:00Z	machine_measurements	current	33,5	M5	A	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	tappikone
2024-05-31T12:00:00Z	machine_measurements	pressure	101,5	S125	Pa	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	tappikone
2024-05-31T12:00:00Z	machine_measurements	vibration	0,003	S126	mm/s	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	tappikone
2024-05-31T12:00:00Z	machine_measurements	energy_consumption	320	M5	W	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T12:00:00Z	machine_measurements	runtime	5344213	M5	s	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	jatkoskone
2024-05-31T12:00:00Z	machine_measurements	cycle_count	1209	M3	count	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	puitelinja	katkonta
2024-05-31T12:00:00Z	machine_measurements	alarm_count	325	feed	count	31133100592-1-26-3-U-13-1	ruovesi	ikkuna	karmilinja	tappikone
2024-05-31T12:00:00Z	machine_measurements	alarm_time	3897432	feed	s	31133100592-1-26-3-U-13-1	ruovesi	ovi	maalaus	ovirata