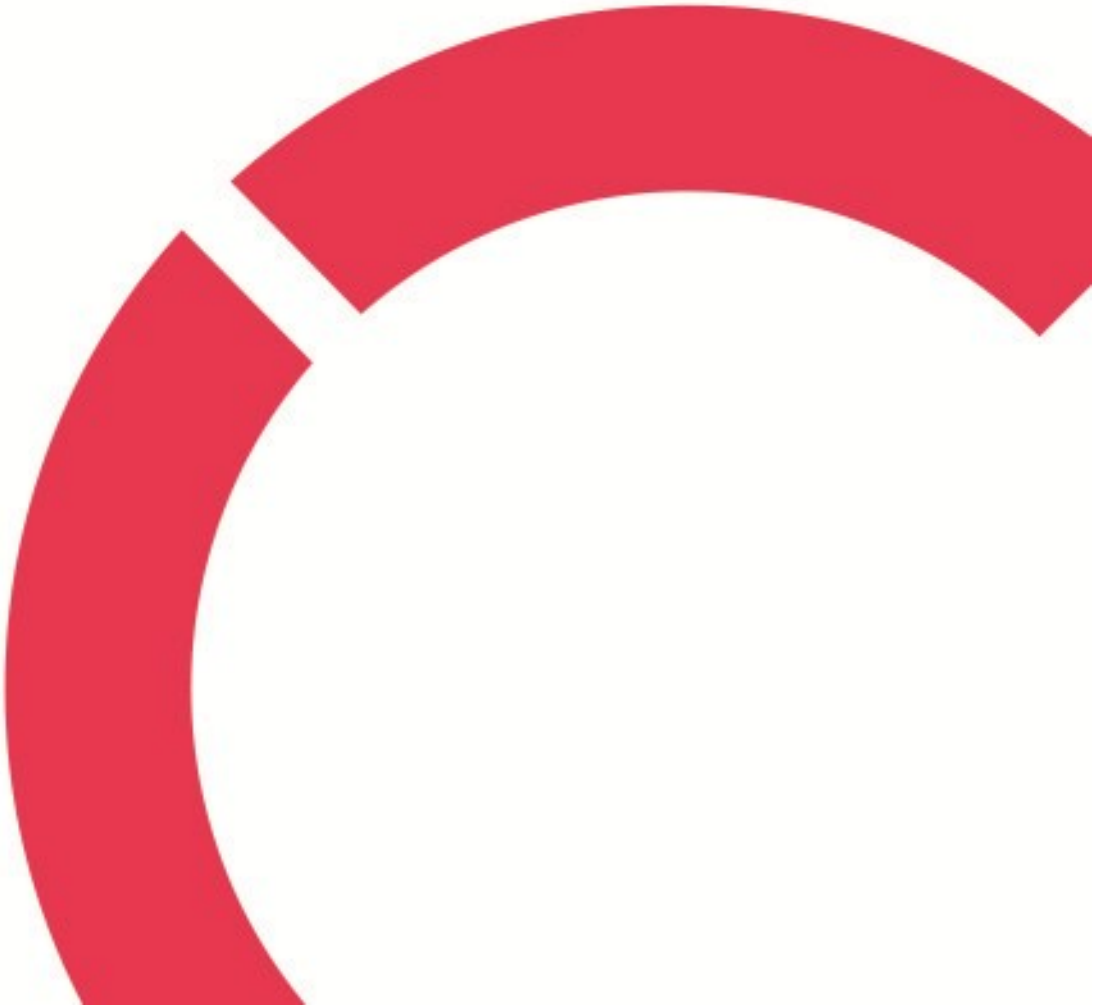
Joonas Eronen

TEOLLISEN IOT-JÄRJESTELMÄN KONSEPTI

Reuna–pilvi-integraatio AWS-pilvipalveluilla

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Insinööri (AMK), tieto- ja viestintätekniikka
Marraskuu 2025



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Marraskuu 2025	**Tekijä/tekijät** Joonas Eronen
Koulutus Insinööri (AMK), tieto- ja viestintätekniikka		☒ AMK ☐ YAMK
Työn nimi TEOLLISEN IOT-JÄRJESTELMÄN KONSEPTI. Reuna–pilvi-integraatio AWS-pilvipalveluilla		
Työn ohjaaja Ari Lamberg		**Sivumäärä** 81 + 11
Työelämäohjaaja Jussi Rautiainen		

Opinnäytetyön toimeksiantajana toimi MJV Automation Oy, teollisuusautomaation palveluja tarjoava yritys. Työn aiheena oli kehittää toimeksiantajalle konsepti kustannustehokkaasta teollisen internetin järjestelmästä, jonka avulla voidaan madaltaa kynnystä siirtyä hyödyntämään Teollisuus 4.0 -ratkaisuja ja niiden potentiaalia. Tavoitteena oli osoittaa, kuinka konseptitasoinen reuna–pilvi-integraatio voidaan toteuttaa ja miten sen avulla voidaan tukea teollisuusyritysten digitalisaatiota. Tietoperustan keskeisiä elementtejä olivat esineiden internet, teollinen internet, teollisuus 4.0, reunalaskenta ja pilvilaskenta sekä keskeiset standardit ja protokollat, kuten OPC Unified Architecture, Message Queuing Telemetry Transport ja Unified Namespace.

Konseptin reunaratkaisu toteutettiin Siemensin SIMATIC IOT2050 -yhdyskäytävälaitteella, johon asennettiin AWS IoT Greengrass -ympäristö ja kehitettiin Java-pohjainen reunasovellus. Pilviratkaisu toteutettiin hyödyntäen Amazon Web Services -pilvipalveluja. Yhdyskäytävälaitteessa AWS IoT Greengrass -ympäristössä suoritettava reunasovellus keräsi simuloitua prosessidataa OPC Unified Architecture -palvelimelta ja esikäsitteli sen. Esikäsittelyn jälkeen data siirrettiin pilveen AWS IoT Core -palveluun, josta se ohjattiin tallennettavaksi, analysoitavaksi ja visualisoitavaksi muiden Amazon Web Services -pilvipalveluiden avulla.

Työn tuloksena saatiin toimiva konsepti teollisen internetin järjestelmästä, jossa hyödynnettiin Teollisuus 4.0:aa tukevia periaatteita, muun muassa Unified Namespace -periaatteita. Toteutettu reunaratkaisu yhdessä käytettyjen pilvipalveluiden kanssa mahdollisti kustannustehokkaan ja skaalautuvan ratkaisun. Ratkaisu soveltuu erityisesti yrityksille ensimmäiseksi askeleeksi teollisen internetin hyödyntämiseen. Kehitetty ratkaisu toimii myös pohjana jatkokehitykselle, jossa voidaan hyödyntää edistyneempää data-analytiikkaa, tekoälyä ja älykkäitä päätöksentekomalleja.

Asiasanat
Amazon Web Services, AWS IoT Core, AWS IoT Greengrass, Esineiden internet, Message Queuing Telemetry Transport, OPC Unified Architecture, Reunalaskenta, Teollinen internet, Unified Namespace

ABSTRACT

Centria University of Applied Sciences	**Date** November 2025	**Author** Joonas Eronen
Degree programme Bachelor of Engineering, Information and Communications Technology		
Name of thesis CONCEPT OF AN INDUSTRIAL IOT SYSTEM. Edge-to-Cloud integration with Amazon Web Services		
Centria supervisor Ari Lamberg		**Pages** 81 + 11
Instructor representing commissioning institution or company Jussi Rautiainen		

The thesis was commissioned by MJV Automation Oy, a company providing industrial automation services. The aim of the thesis was to develop a proof of concept for a cost-effective industrial internet system for the commissioning company, to lower the barriers of adopting Industry 4.0 solutions and realizing their potential. The objective was to demonstrate how concept-level edge-to-cloud integration can be implemented and how it can be used to support the digitalization of industrial companies. Key elements of the theoretical framework were the Internet of Things, Industrial Internet, Industry 4.0, Edge and Cloud Computing, and key standards and protocols such as OPC Unified Architecture, Message Queuing Telemetry Transport and Unified Namespace.

The concept's edge solution was implemented on a Siemens SIMATIC IOT2050 gateway device, on which the AWS IoT Greengrass environment was installed, and a Java-based edge application was developed. The cloud solution was implemented using Amazon Web Services. The edge application, running in the AWS IoT Greengrass environment on the gateway device, collected simulated process data from the OPC Unified Architecture server and preprocessed it. After preprocessing, the data was transferred to the AWS IoT Core service, from where it was routed for storage, analysis, and visualization using other Amazon Web Services.

The result of the thesis was a working proof of concept for an industrial internet system that applied selected principles supporting Industry 4.0, including Unified Namespace principles. The implemented edge solution, together with the cloud services used, enabled a cost-effective and scalable solution. The solution is especially suitable for companies as a first step to adopting the industrial internet. The developed solution also provides a solid foundation for further development, where more advanced data analytics, artificial intelligence and intelligent decision-making models can be applied.

Key words
Amazon Web Services, AWS IoT Core, AWS IoT Greengrass, Edge computing, Industrial Internet of Things, Internet of Things, Message Queuing Telemetry Transport, OPC Unified Architecture, Unified Namespace

KÄSITTEIDEN MÄÄRITTELY

API

Application Programming Interface on sovellusohjelmointirajapinta, jonka avulla sovellukset tai palvelut voivat vaihtaa tietoa ja toiminnallisuuksia keskenään.

AWS

Amazon Web Services on pilvipalvelualusta, joka tarjoaa laajan valikoiman palveluita datan keruuseen, siirtoon, tallennukseen, analysointiin ja visualisointiin.

HTTP

Hypertext Transfer Protocol. Verkkoprotokolla, jota käytetään tiedonsiirtoon selaimen ja palvelimen välillä.

IoT

Internet of Things on verkko, jossa fyysiset esineet, kuten anturit, koneet ja laitteet, on liitetty internetiin mahdollistamaan automaattisen tiedonsiirron ja etäohjauksen.

IIoT

Industrial Internet of Things on teollinen sovellusmuoto IoT-teknologioista, jossa esimerkiksi koneet, anturit ja automaatiojärjestelmät yhdistetään verkkoon tuotannon seurantaa ja ohjausta varten.

MQTT

Message Queuing Telemetry Transport on kevyt ja tehokas viestinvälitysprotokolla, jota käytetään erityisesti IoT- ja IIoT-sovelluksissa datan siirtoon pienellä kaistanleveydellä ja vähäisellä virrankulutuksella.

OPC UA

Open Platform Communications Unified Architecture on teollisuudessa käytettävä viestintäprotokolla, joka mahdollistaa laitteiden ja järjestelmien välisen standardoidun, alustariippumattoman ja tietoturvallisen tiedonsiirron.

Pilvilaskenta (cloud computing)
Pilvilaskenta on laskentamalli, jossa laskentateho ja datan tallennus tapahtuvat keskitetysti pilvipalveluissa paikallisten laitteiden sijaan.

PLC
Programmable Logic Controller on teollisuudessa käytettävä ohjelmoitava logiikkaohjain, joka vastaanottaa signaaleja esimerkiksi antureilta tai laitteilta ja ohjaa toimilaitteita reaaliaikaisesti ohjelmoidun logiikan mukaisesti.

REST
Representational State Transfer. Arkkitehtuurityyli, joka määrittää, miten verkkopalvelut voivat kommunikoida HTTP-protokollaa hyödyntäen.

Reunalaite (edge device)
Kenttätason laite, kuten anturi, toimilaite tai älykamera, joka tuottaa dataa fyysisestä prosessista. Joissakin tapauksissa reunalaite voi suorittaa yksinkertaista paikallista käsittelyä, kuten datan suodatusta.

Reunalaskenta (edge computing)
Reunalaskenta on laskentamalli, jossa osa datan käsittelystä ja analysoinnista tapahtuu lähellä datan lähdettä, esimerkiksi reunasolmussa tai yhdyskäytävälaitteessa.

Reunasolmu (edge node)
Laskentayksikkö, joka kokoaa useiden reunalaitteiden tuottamaa dataa ja suorittaa sen datan pohjalta analytiikkaa tai päätöksentekoa. Reunasolmun avulla voidaan vähentää pilveen siirrettävän datan määrää ja mahdollistaa paikallinen optimointi esimerkiksi prosessien ohjauksessa.

TCP/IP
Transmission Control Protocol / Internet Protocol. Internetin perusprotokollapino, jota käytetään tiedonsiirrossa verkkojen välillä.

Teollisuus 4.0 (Industry 4.0)
Neljäs teollinen vallankumous, jossa yhdistyvät automaatio, reaaliaikainen data-analytiikka, IoT ja tekoäly.

UNS

Unified Namespace on arkkitehtuuri, jossa kaikki tuotanto- ja prosessidata yhdistetään yhteen keskitettyyn, tapahtumapohjaiseen nimialueeseen, josta eri järjestelmät voivat lukea tai julkaista dataa.

Yhdyskäytävälaite (edge gateway)

Reunasolmun erikoistapaus, jonka päätehtävä on toimia OT- ja IT-järjestelmien rajapinnassa ja muuntaa protokollia, esimerkiksi OPC UA:sta MQTT:ksi. Monet modernit yhdyskäytävälaitteet sisältävät lisäksi reunasolmun ominaisuuksia, kuten datan esikäsittelyä ja paikallista analytiikkaa.

1 JOHDANTO

Tämän opinnäytetyön tekohetkellä teollisuus on elänyt jo jonkin aikaa Teollisuus 4.0 -aikakautta eli neljättä teollista vallankumousta, jonka myötä esineiden internet (IoT), reaaliaikainen data-analytiikka ja tekoälyratkaisut ovat muuttaneet teollisuuden prosesseja esimerkiksi valmistavassa teollisuudessa. Samalla puhutaan jo siirtymisestä Teollisuus 5.0 -aikakauteen, uuteen lähestymistapaan, jossa korostetaan ihmiskeskeisyyttä ja kestävyyttä. Monet erityisesti pienet ja keskisuuret teollisuusyritykset toimivat kuitenkin yhä Teollisuus 3.0 -periaatteiden mukaisesti, joissa automaatiojärjestelmät noudattavat ennalta määriteltyjä sääntöjä ja logiikkaa, mutta eivät hyödynnä prosessidataa reaaliaikaisesti. Datan kerääminen tapahtuu usein manuaalisesti, yksinkertaisten raporttien tai CSV-tiedostojen kautta, eikä sitä analysoida strategisesti. Tämä toimintapa ei hyödynnä datan koko potentiaalia, joka voisi mahdollistaa kustannustehokkaamman tuotannon, paremman päätöksenteon ja ympäristöystävällisemmät ratkaisut.

Siirtymistä Teollisuus 4.0 -ratkaisuihin voivat hidastaa monet tekijät, kuten suuret aloituskustannukset, teknologian monimutkaisuus tai vanhat automaatiojärjestelmät, joita voi olla vaikeaa tai kallista integroida uusiin teknologioihin. Lisäksi haasteena voivat olla kannattavuuden epävarmuus sekä osaamisen ja resurssien puute. Erityisesti pienemmät yritykset voivat kokea nämä haasteet merkittävinä esteinä. Tämä on toiminnallinen opinnäytetyö, jossa keskitytään kehittämään toimeksiantajalle konsepti kustannustehokkaasta teollisesta IoT-järjestelmästä, jonka tarkoituksena on madaltaa kynnystä siirtyä hyödyntämään Teollisuus 4.0 -ratkaisuja. Kehitettävän konseptin käyttöönotto ei vaadi suurta alkuinvestointia tai suuria resursseja. Sen avulla voidaan ottaa ensiaskeleet Teollisuus 4.0 -ratkaisuihin aloittamalla pienemmässä mittakaavassa ja kokemuksen lisääntyessä sekä tarpeiden selkiytyessä järjestelmää voidaan jatkokehittää ja laajentaa.

Konsepti koostuu reunaratkaisusta, joka kerää ja esikäsittelee dataa reunalla sekä pilvi-integraatiosta, jossa sitä analysoidaan, visualisoidaan ja tallennetaan. Toimeksiantajalla on pitkä kokemus Siemensin ohjelmoitaviin logiikkaohjaimiin (PLC) perustuvien teollisuuden automaatiojärjestelmien toimittamisesta. Näiden automaatiojärjestelmien logiikkaohjaimissa on sisäänrakennettu OPC UA -palvelin, jota konsepti hyödyntää rajapintana automaatiojärjestelmään. Reunaratkaisu koostuu yhdyskäytävälaitteesta ja kehittämästäni Java-pohjaisesta reunasovelluksesta, jota suoritetaan laitteella AWS Greengrass -ympäristössä. Reunasovellus lukee OPC UA -palvelimelta teollisuuden prosessidataa ja siirtää sen esikäsittelyn jälkeen MQTT-protokollalla Amazon Web Services (AWS) -pilvipalveluihin

jatkokäsittelyä ja analysointia varten. Konseptin suunnittelun lähtökohtana on ollut Unified Namespace (UNS) -arkkitehtuuri, joka on ohjannut kaikkia teknologisia valintoja, mukaan lukien yhdyskäytävälaiteratkaisun, tiedonsiirtoprotokollien ja pilvialustan valinnan. UNS-arkkitehtuuri mahdollistaa järjestelmällisen ja skaalautuvan datan hallinnan teollisuusympäristöissä.

Työn laajuuden vuoksi opinnäytetyön ulkopuolelle rajataan pilvipalvelussa toteutettava älykäs päätöksenteko, kehittynyt data-analytiikka, tekoälyn hyödyntäminen sekä muiden järjestelmien integraatiot arkkitehtuuriin. Lisäksi työ ei sisällä reaaliaikaista päätöksentekoa tai automaattista ohjausta, laajamittaista kyberturva- ja käyttöliittymäsuunnittelua tai järjestelmän testausta tuotantoympäristössä. Työssä sovelletaan teollisuusstandardeja konseptitasoisesti arkkitehtuurin havainnollistamiseksi, eikä niiden täydellinen toteutus kuulu työn laajuuteen. Datan käsittely on rajattu simuloituihin mittauksiin ja arkkitehtuurin testaamiseen, eikä työ sisällä valmiin tuotteen kehittämistä tai kaupallistamista. Tämä rajaus mahdollistaa kustannustehokkaan ja helposti käyttöönotettavan perusratkaisun, josta on helppo lähteä liikkeelle. Konsepti toimii kuitenkin perustana, johon voidaan laajentaa jatkokehityksenä tekoälyyn perustuva päätöksenteko, edistynyt analytiikka ja järjestelmäintegraatiot.

Tutkimusongelmat ja kehittämistehtävät voidaan tiivistää viideksi kokonaisuudeksi, joihin tämä opinnäytetyö pyrkii vastaamaan. Ensinnäkin työssä tarkastellaan, kuinka kustannustehokas ja skaalautuva teollinen IoT-arkkitehtuuri voidaan toteuttaa. Toiseksi työssä tarkastellaan, miten prosessidataa voidaan hyödyntää kustannustehokkaalla tavalla pilvessä. Kolmantena tutkimuskohteena on yhdyskäytävälaitteen rooli datansiirron kustannusten optimoinnissa. Neljänneksi tarkastellaan, kuinka AWS-pilvipalveluilla voidaan rakentaa kustannustehokas ratkaisu datan analysointiin, visualisointiin ja tallentamiseen. Viimeisenä näkökulmana on avoimen lähdekoodin lähestymistavan merkitys IoT-kehityksessä ja sen tarjoamat mahdollisuudet. Näiden kysymysten avulla työ pyrkii kehittämään ratkaisun, joka mahdollistaa kustannustehokkaan siirtymisen Teollisuus 4.0 -aikakauteen.

Työn toisessa luvussa käsitellään IoT:ta ja teollista internetiä (IIoT) ilmiönä. Luvussa tarkastellaan muun muassa IoT:n merkitystä, Teollisuus 4.0 -kontekstia, reunalaskennan hyötyjä ja haasteita, IoT:n yleisiä haasteita ja tulevia suuntaviivoja sekä IIoT-ympäristöjen teknisiä vaatimuksia ja kyberturvallisuutta. Luvussa käsitellään myös keskeisiä teollisuusstandardeja ja ajankohtaisia EU-säädöksiä, sillä ne asettavat vaatimuksia IIoT-järjestelmien turvallisuudelle ja kehitykselle. Näiden huomioon ottaminen on tärkeää, jotta suunniteltu arkkitehtuuri vastaa tulevaisuuden vaatimuksia ja tukee standardoitua toimintaympäristöä. Kolmannessa luvussa esitellään keskeiset IoT-teknologiat ja arkkitehtuurit, kuten UNS-arkkitehtuuri sekä IIoT:ssa käytettävät viestintäprotokollat ja pilvipalvelut. Neljäs luku keskittyy

työn käytännön toteutukseen. Siinä kuvataan laitteisto- ja ohjelmistototeutus, järjestelmäarkkitehtuuri ja datavirran kulku sekä kehitetty reunasovellus ja sen kyberturvaratkaisut. Lisäksi luvussa esitellään käytetyt AWS-pilvipalvelut sekä niiden rooli datan keruussa, siirrossa, tallennuksessa ja visualisoinnissa. Lopuksi työssä arvioidaan kehitetyn ratkaisun toimivuutta sekä sen tarjoamia jatkokehitysmahdollisuuksia.

Tässä työssä keskeisiä käsitteitä ovat teollisen internetin arkkitehtuureissa laajasti käytetyt protokollat ja viitekehykset. Näitä ovat erityisesti OPC UA ja MQTT, jotka muodostavat perustan reaaliaikaiselle datansiirrolle, sekä Unified Namespace (UNS) -arkkitehtuuri, joka tarjoaa hierarkkisen mallin tiedonhallintaan teollisuusympäristöissä. Lisäksi keskeisiä käsitteitä ovat yhdyskäytävälaite, jota tässä työssä edustaa SIMATIC IOT2050, sekä AWS-pilvipalvelut, joista korostuvat AWS IoT Greengrass, AWS IoT Core, AWS Lambda, AWS IoT SiteWise ja Amazon Timestream. Näiden avulla toteutettu arkkitehtuuri mahdollistaa kustannustehokkaan ja skaalautuvan IIoT-järjestelmän rakentamisen. Työn tekninen toteutus perustuu ensisijaisesti AWS:n viralliseen dokumentaatioon, Siemensin dokumentaatioon sekä Eclipsen avoimen lähdekoodin kirjastoihin ja niiden dokumentaatioon. Lisäksi työssä hyödynnetään alan tutkimuskirjallisuutta ja tieteellisiä artikkeleita, jotka käsittelevät Teollisuus 4.0 -ratkaisujen käyttöönottoa ja IoT-sovellusten kehittämistä. Näiden lähteiden yhdistelmä luo pohjan sekä työn teoreettiselle viitekehykselle että käytännön toteutukselle.

2 IOT JA TEOLLINEN INTERNET (IIOT)

Tässä luvussa tarkastellaan esineiden internetiä (IoT) ja teollista internetiä (IIoT) yleisemmällä tasolla ilmiönä ja liiketoimintaa muuttavana kehityssuuntana. Luvussa keskitytään IoT:n käsitteen syntyyn, sen merkitykseen datalähtöisessä maailmassa sekä siirtymään teollisiin sovelluksiin. Lisäksi esitellään keskeisiä Teollisuus 4.0 (Industry 4.0) -trendejä, sovelluskohteita, haasteita sekä siitä, millaisia laitteisto- ja tietoturvavaatimuksia älykkään tuotantoympäristön rakentaminen edellyttää.

IoT-käsitteen (Internet of Things) esitteli ensimmäisen kerran vuonna 1999 brittiläinen teknologia-alan pioneeri Kevin Ashton. Ashtonin ydinajatus oli, että tietokoneet ja internet ovat lähes täysin riippuvaisia ihmisten tuottamasta datasta, kuten kirjoittamisesta, kuvaamisesta tai skannaamisesta. Ongelmana on ihmisten rajallinen aika, huomio ja tarkkuus reaalimaailman datan keräämisessä. IoT:n tavoitteena on antaa tietokoneille kyky havaita fyysistä maailmaa näkemällä, kuulemalla ja tunnistamalla asioita ilman ihmistä. Tähän tarvitaan antureita ja RFID-teknologiaa, joiden avulla laitteet voivat itsenäisesti kerätä tietoa ympäristöstään. Ashton korostaa, että maailma perustuu asioihin, ei pelkästään ideoihin ja informaatioon. Jos tietokoneet tietäisivät kaiken esineistä, voisimme seurata, laskea ja optimoida kaiken, esimerkiksi vähentää hukkaa, parantaa tehokkuutta ja tietää, milloin jokin pitää huoltaa tai vaihtaa. Ashtonin mukaan kyseessä on vallankumouksellinen muutos, joka voi vaikuttaa maailmaan jopa enemmän kuin internet. (Ashton 2009.)

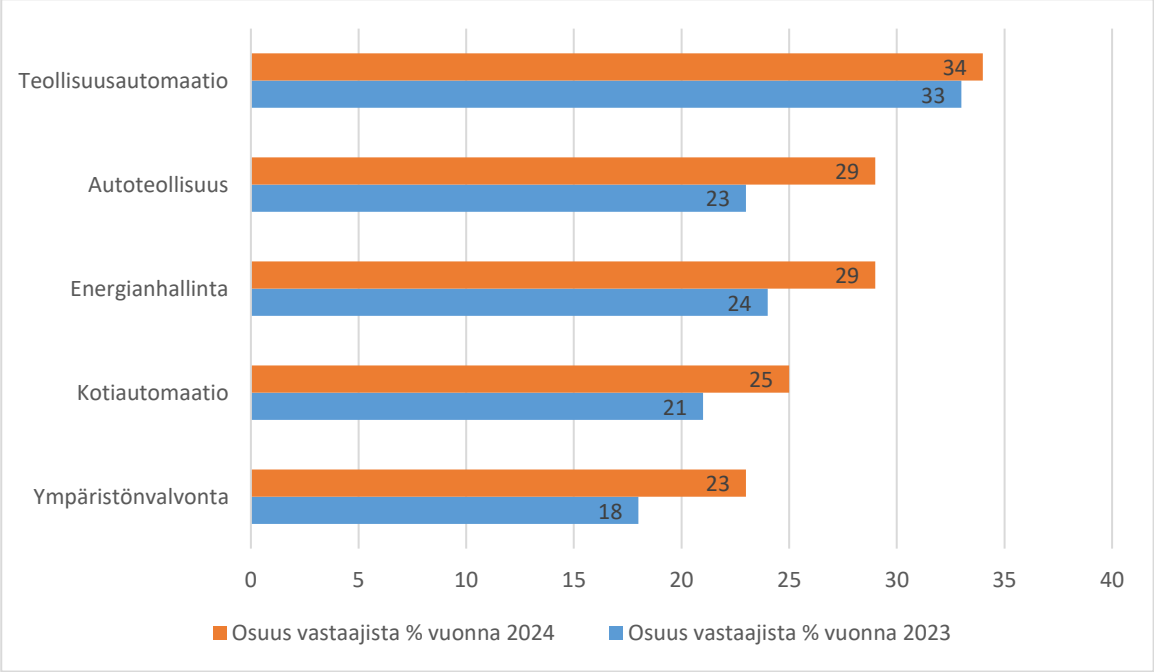
2.1 IoT:n merkitys ja datalähtöinen maailma

Liiketoiminta ei voi koskaan olla täysin irrallaan reaalimaailmasta. Vaikka organisaatio keskittyisi ainoastaan digitaalisiin palveluihin tai tuotteisiin, se hyötyy silti IoT:sta. Esimerkiksi pilvipalvelut muodostuvat miljoonista palvelimista, jotka sijaitsevat ympäri maailmaa hajautetuissa datakeskuksissa. Näitä keskuksia valvotaan jatkuvasti. Lämpötila-, kosteus-, ilmanlaatu- ja tärinäanturit varmistavat, että laitteet toimivat optimaalisissa olosuhteissa. Valvontakamerat ja muut turvajärjestelmät suojaavat keskuksia ulkopuolisilta tunkeilijoilta. (Desbiens 2023, xxiii–xxiv.)

IoT:n merkitys ei rajoitu vain infrastruktuuriin, vaan se tarjoaa perustan laajemmalle teknologiselle muutokselle. IoT yhdistää fyysiset laitteet internetiin datan keräämistä ja vuorovaikutusta varten. IoT mahdollistaa laitteiden etähallinnan ja automaattiset tehtävät. Tätä voidaan soveltaa useissa kohteissa

yksinkertaisista älykotien laitteista, kuten termostaateista tai valaistuksen ohjauksesta aina vaativampiin kohteisiin, kuten teollisuuden tai terveydenhuollon antureihin ja laitteisiin. (Quincozes, Quincozes, Kazienko, Gama, Cheikhrouhou & Koubaa 2024.) Pelkkä laitteiden yhdistäminen internetiin ei kuitenkaan vielä riitä. Todelliset hyödyt syntyvät vasta, kun kerättyä dataa osataan hyödyntää liiketoiminnan tueksi. Organisaatiot tarvitsevat dataa voidakseen tehdä parempia päätöksiä, tukeakseen prosessien automatisointia ja parantaakseen työelämän laatua. IoT toimii keinona saavuttaa tämä data. Aiempina vuosina saavuttamattomissa olleet tiedonlähteet ovat nykyään helposti saatavilla laskentatehon kehittymisen, laitteiden pienentymisen ja hintojen alenemisen ansiosta. (Desbiens 2023, xxiv.)

Tämä kehitys näkyy myös alan tilastoissa ja tutkimuksissa. Vuonna 2024 Eclipse Foundation teki vuosittaisen laajan IoT-aiheisen kyselyn: ”Eclipse IoT & Embedded Developer Survey”, johon osallistui 747 kehittäjää, arkkitehtia ja päätöksentekijää eri puolilta maailmaa. Kysely antaa kattavan kuvan siitä, mille toimialoille IoT-ratkaisuja kehitetään ja millaisia trendejä kehittäjät painottavat. Tulosten mukaan IoT-ratkaisuja kehitettiin eniten teollisuusautomaatioon (34 %), autoteollisuuteen (29 %) ja energianhallintaan (29 %) (KUVIO 1). Kuviossa 1 näkyy myös selkeä kasvu kaikilla viidellä toimialalla verrattuna vuoteen 2023. Teollisuusautomaatio on säilyttänyt vahvan aseman suurimpana markkinasegmenttinä, mutta myös muut toimialat, kuten energianhallinta ja autoteollisuus, ovat saaneet merkittävää kasvua vuoteen 2023 verrattuna. (Eclipse Foundation 2024, 8.)



KUVIO 1. IoT-sovellusten toimialat vastaajien mukaan (mukaillen Eclipse Foundation 2024, 8)

Kaikesta kehityksestä huolimatta IoT:n varhaisissa vaiheissa oli myös merkittäviä rajoitteita, jotka hidastivat sen täyttä potentiaalia. Monet varhaiset IoT-laitteet suunniteltiin hyvin kapeisiin käyttötarkoituksiin, mikä rajoitti niiden tuottaman datan käyttökelpoisuutta ja koko potentiaalin hyödyntämistä. Niiltä puuttui joustavuus laajentua uusilla ominaisuuksilla tai kehittyä seuraavan sukupolven ratkaisuihin. Tämän seurauksena jäi hyödyntämättä suuri osa IoT:n ja muiden teknologioiden, kuten tekoälyn ja lisätyn todellisuuden (augmented reality, AR) yhdistelmien tarjoamasta innovaatiopotentiaalista. Lisäksi ilman avoimia ja skaalautuvia standardeja varhaiset toteutukset eivät tarjonneet riittävän luotettavaa perustaa, joka olisi kyennyt edistämään luottamusta ja laajamittaista käyttöönottoa. Tämä rajoitti edistyneiden teknologioiden, kuten tekoälyn ja lisätyn todellisuuden hyödyntämisen lähinnä korkean lisäarvon ja kapeiden erikoissovellusten alueelle. (Figueredo, Seed & Wang 2022, 5645.)

Nämä rajoitteet heijastuvat edelleen siihen, miten IoT-dataa jaetaan ja hyödynnetään eri toimijoiden kesken. IoT-datan koko potentiaali jää usein hyödyntämättä, koska data on rajoittunut yksittäisiin käyttötarkoituksiin tai vain harvojen suljettujen käyttäjäryhmien saataville. Mahdollisuudet ovat huomattavasti laajemmat, kun dataa voidaan jakaa markkinapaikkamallin avulla. Tämä tarjoaa viitekehyksen, jossa monet datan toimittajat voivat olla vuorovaikutuksessa datan käyttäjien, kuten tekoälykehittäjien ja IoT-palveluntarjoajien kanssa. Lisäksi se mahdollistaa kysynnän tunnistamisen ja kannustaa tarjoamaan korkealaatuista ja luotettavaa dataa. (Figueredo ym. 2022, 5645.)

Kun IoT-dataa osataan hyödyntää oikein, se voi muuttaa merkittävästi yrityksen liiketoimintamallia ja arvolupausta. IoT-ominaisuuksien lisääminen yrityksen liiketoimintamalliin mahdollistaa etävalvonnan ja tiheän raportoinnin. Lisäksi IoT tukee toimitusketjujen, hajautettujen resurssien ja liikekumppaneiden tiiviimpää kytkentää. IoT:n tuottama data tarjoaa mahdollisuuden lisäarvopalveluiden luomiseen, esimerkiksi datan vaihdon avulla toimitusketjun kumppaneiden kanssa. Tämä tuo uusia ulottuvuuksia yrityksen liiketoimintamalliin, muun muassa arvolupauksiin sekä uusien asiakas- ja kumppanisuhteiden syntymiseen. IoT-laitteet tuottavat valtavia määriä dataa, mutta sen taloudellinen arvo jää usein hyödyntämättä. Monia ratkaisuja ei ole suunniteltu avoimeen datan jakamiseen, mikä estää datan kierron ja uusien arvolähtöisten palveluiden syntymisen. Markkinapaikkakonseptit ja avoimet standardit voivat ratkaista tämän mahdollistamalla skaalautuvat ja monitoimijaympäristöön perustuvat liiketoimintamallit. (Figueredo ym. 2022, 5647.)

2.2 Teollisuus 4.0 ja teollinen internet (IIoT)

Teollisuus 4.0 viittaa modernin valmistavan teollisuuden neljänteen vallankumoukseen, jossa keskiössä ovat digitalisaatio, tekoäly, data-analytiikka ja automaatio. Tavoitteena on lisätä tuotannon tehokkuutta, joustavuutta ja älykkyyttä (Attaran, Attaran & Celik 2024, 2.) Näiden tavoitteiden mahdollistamiseksi vaaditaan luotettavaa ja reaaliaikaista tiedonsiirtoa fyysisen ja digitaalisen maailman välillä. Tässä suhteessa teollinen esineiden internet (IIoT) on kriittinen teknologia, joka yhdistää älykkäät laitteet, anturit ja järjestelmät tuotantoympäristöön. Se tarjoaa infrastruktuurin, jonka avulla Teollisuus 4.0:n keskeiset periaatteet voidaan toteuttaa käytännössä. Ilman IIoT:tä Teollisuus 4.0 jäisi pitkälti abstraktiksi konseptiksi. (Ahmed, Bin Alam, Hoque, Lameesa, Afrin, Farah, Kabir, Shafiullah & Muyeen 2023, 4.)

Teollisuus 4.0:n kaksi tärkeintä konseptia ovat teollinen esineiden internet (IIoT) ja digitaalinen kaksonen (Digital Twin) (Attaran ym. 2024, 2). IIoT on IoT-teknologioiden laajennettua soveltamista teollisessa kontekstissa. IIoT hyödyntää erilaisia teknologioita, kuten antureita, pilvipalveluita, data-analytiikkaa, koneoppimista ja tekoälyä. Se mahdollistaa teollisten prosessien seurannan ja optimoinnin aina tuotteen konseptoinnista ja valmistuksesta toimitusketjuun. Lisäksi IIoT mahdollistaa laaja-alaisen tiedonkeruun ja -hyödyntämisen, vähentää virheitä, parantaa tehokkuutta ja syventää ymmärrystä prosesseista. Se on keskeinen osa Teollisuus 4.0 -kehitystä ja tarjoaa uudenlaista kokonaisvaltaista päätöksenteon tukea teollisuuden toimijoille. (Borgosz & Dikicioglu 2024, 1.) Digitaaliset kaksoset ovat virtuaalisia malleja fyysisistä laitteista, prosesseista tai järjestelmistä, jotka mahdollistavat mallintamisen ja esittämisen digitaalisessa muodossa. Ne mahdollistavat tilan, toiminnan ja käyttäytymisen reaaliaikaisen seurannan, simulaation ja analyysin. Niitä sovelletaan muun muassa tuotannon optimointiin, koneiden kunnossapitoon, suunnittelun validointiin ja toimitusketjun hallintaan. (Attaran ym. 2024, 3.)

Integroimalla IIoT:n digitaalisiin kaksosiin, organisaatiot voivat kerätä ja analysoida tietoa laitteistaan, simuloida eri tilanteita ja tukea päätöksentekoa ennakoivasti. IIoT ja digitaaliset kaksoset yhdessä mahdollistavat uusia tapoja suunnitella, valmistaa ja ylläpitää tuotteita. (Attaran ym. 2024, 5–6.) IIoT:n ansiosta valmistavat yritykset voivat kehittää parempia tuotteita, havaita fyysisiä ongelmia nopeammin ja ennustaa tuloksia tarkemmin. Vielä muutama vuosi sitten tämä ei ollut mahdollista laajassa mittakaavassa digitaalisten teknologioiden rajoittuneiden ominaisuuksien vuoksi. Lisäksi korkeat laskenta-, tallennus- ja kaistanleveyskustannukset rajoittivat laajempaa käyttöönottoa. Teknologian kehittyminen ja kustannusten merkittävä lasku ovat kuitenkin mahdollistaneet digitaalisten kaksosten ja IIoT:n laajemman käyttöönoton valmistavassa teollisuudessa. (Attaran ym. 2024, 2, 4.)

2.3 Reunalaskenta (edge computing)

Reunalaskenta on laskentamalli, jonka tavoitteena on ratkaista IoT-datan käsittelyyn liittyviä haasteita sijoittamalla laskentatehoa lähemmäksi datan syntypaikkaa. Tällä pyritään vähentämään vasteaikoja, kaistanleveyden käyttöä, verkon kuormitusta ja tiedonsiirtokustannuksia. Reunalaskennassa data prosessoidaan paikallisesti hyödyntämällä IoT-antureita, reunalaitteita ja reunasolmuja, jotka sijaitsevat lähellä datan syntypaikkaa. Lisäksi reunasolmut ja yhdyskäytävälaitteet voivat tukea eri viestintäprotokollien muuntamista ja yhdistämistä sekä tarjota paikallista laskentakapasiteettia. (Singh, Gautam & Arya 2025, 1226, 1228.)

2.3.1 Reunalaskennan hyödyt

Reunalaskenta tarjoaa useita etuja, erityisesti toimialoilla ja sovelluksissa, jotka edellyttävät reaaliaikaista prosessointia, matalaa viivettä ja paikallista datan hallintaa. Kun data prosessoidaan lähellä datan syntypaikkaa, se vähentää merkittävästi datan siirtoon kuluvaa viivettä. Tämä on kriittistä matalan viiveen sovelluksissa, kuten autonomisissa ajoneuvoissa, etäterveydenhuollossa ja teollisuusautomaatiossa. Se mahdollistaa datan nopean analysoinnin ja siihen perustuvan päätöksenteon paikallisesti, ilman keskitettyjen palvelimien tarvetta. Tämä parantaa merkittävästi sovellusten toimintaa, erityisesti tilanteissa, joissa tarvitaan reaaliaikaista vasteaikaa, kuten ennakoivassa kunnossapidossa, lisätyn todellisuuden sovelluksissa sekä älykaupunkien järjestelmissä. Lisäksi reunalaskenta mahdollistaa kriittisten toimintojen ylläpitämisen ja autonomisen toiminnan tilanteissa, joissa verkkoyhteys on hetkellisesti katkennut esimerkiksi syrjäisillä tai huonon yhteyden alueilla. (Singh ym. 2025, 1228–1229.)

Reunalaskenta vähentää pilveen siirrettävän datan määrää, kun tietoa prosessoidaan ja suodatetaan jo paikallisesti. Tämä säästää verkon kaistanleveyttä, estää verkkoa ylikuormittumasta sekä vähentää merkittävästi datan siirrosta aiheutuvia kustannuksia. Se vähentää myös tarpeita suurikaistaisille yhteyksille, jotka voivat olla kalliita. Lisäksi reunalaskennan avulla suorituskyky paranee ympäristöissä, joissa kaistanleveys on rajallinen, kuten mobiiliverkoissa, satelliittiyhteyksissä tai tilanteissa, joissa datan määrä on erittäin suuri. Reunalaskenta voi toisaalta parantaa tietosuojaa ja turvallisuutta pilveen vähentyneen datansiirron vuoksi, mikä puolestaan pienentää riskiä datan paljastumiselle siirron aikana. Se tukee myös alueellisia tietosuojasäädöksiä, jotka rajoittavat datan siirtoa EU:n ulkopuolelle. (Singh ym. 2025, 1229.)

Reunalaskennan soveltaminen arkkitehtuuritasolla mahdollistaa järjestelmien skaalautumisen lisäämällä uusia reunalaitteita ja reunasolmuja tarpeen mukaan. Reunalaskenta-arkkitehtuurin joustavuus tukee järjestelmien mukautumista kasvaviin datamääriin ja muuttuviin liiketoimintavaatimuksiin. Datan paikallinen käsittely vähentää myös energiankulutusta, koska tietoa ei tarvitse siirtää pitkiä matkoja. Reunalaitteet ja reunasolmut on yleensä optimoituja energiatehokkuuden mukaan, tehden koko järjestelmästä kestävämmän. (Singh ym. 2025, 1229.)

2.3.2 Reunalaskennan haasteet

Reunalaskenta tarjoaa paljon etuja, mutta siihen liittyy myös haasteita, kuten resurssirajoituksia, tietoturvaheikkouksia ja hallinnollista monimutkaisuutta. Reunalaitteet, kuten IoT-anturit ja yhdyskäytävälaitteet, ovat usein rajoitetuilla resursseilla varustettuja, mikä voi rajoittaa monimutkaisempien laskentatehtävien suoritusta. Reunalaskenta voi lisätä kyberhaavoittuvuutta, sillä se kasvattaa hyökkäyspinta-alaa useiden päätepisteiden vuoksi. Vaikka siirrettävän datan määrä vähenee ja riski datan paljastumiselle pienenee, toisaalta datan suojaaminen siirron aikana sekä laitteiden todennus ja päätepisteiden suojaaminen voivat olla haastavia, erityisesti hajautetuissa ympäristöissä. Reunalaskentajärjestelmien skaalaaminen ja päivittäminen voi olla monimutkaisempaa kuin pilviratkaisuissa, sillä se edellyttää fyysisen laitteiston lisäämistä tai päivittämistä kohteissa. Lisäksi reunalaskenta vaatii toimivat paikalliset verkot toimiakseen tehokkaasti, mikä voi aiheuttaa myös haasteita. Verkon ruuhkautuminen, häiriöt tai katkokset voivat vaikuttaa negatiivisesti reunalaskennan suorituskykyyn tai jopa estää sen toiminnan. (Singh ym. 2025, 1229.)

Reunalaitteiden, pilvipalvelimien ja keskitettyjen järjestelmien datan pitäminen yhdenmukaisena voi olla haastavaa. Viiveistä tai verkkoyhteyden ongelmista johtuva epäjohdonmukainen data voi johtaa virheelliseen analytiikkaan ja päätöksentekoon. Vaikka datan siirron väheneminen pienentää energiankulutusta, useiden reunalaitteiden ja reunasolmujen käyttö voi lisätä kokonaiskulutusta. Energiatehokkuuden ja suorituskyvyn tasapainottaminen on siten tärkeä haaste. Reunalaskennan täyden potentiaalin hyödyntämiseksi on tärkeää tunnistaa haasteet, jotta ne voidaan ratkaista tehokkaasti. Reunalaskennan käyttöönotossa tarvitaan vahvoja standardointiin ja turvallisuuteen perustuvia strategioita. Lisäksi tulee ottaa huomioon organisaation sisäiset skaalautuvuus- ja energiatehokkuustarpeet. Tämänhetkisistä haasteista huolimatta jatkuva kehitys tuo parannuksia reunalaitteisiin, ohjelmistoihin ja verkkoihin. Reunalaskennalla on nyt ja tulevaisuudessa merkittävä rooli digitaalisessa transformaatiossa, koska se

tarjoaa kykyä reaaliaikaiseen näkymään ja paikalliseen päätöksentekoon. (Singh ym. 2025, 1229–1230.)

2.4 IoT:n haasteet ja tulevat suuntaviivat

IIoT-järjestelmillä on yhä merkittävämpi rooli monien sovellusten mahdollistajana, mutta lukuisista eduista huolimatta niiden kehitykseen liittyy edelleen kriittisiä ongelmia erityisesti tietoturvan ja yksityisyyden näkökulmasta. Lisäksi aiheeseen liittyy useita tutkimuksellisia haasteita, kuten resurssienhallinta, oikeudenmukaisuus, taloudellisuus sekä IIoT:n rooli tulevaisuuden langattomissa verkoissa. (Aouedi, Vu, Sacco, Nguyen, Piamrat, Marchetto & Pham, 2024, 1279.)

2.4.1 Resurssienhallinta IIoT:ssa

Tulevaisuuden IIoT-järjestelmissä eri teknologiat, kuten langattomat verkot, pilvipalvelut ja tekoälyratkaisut integroituvat toimimaan yhtenäisenä järjestelmänä. Tämä kehitys tuo mukanaan haasteita sekä verkkojen käyttöönottoon että tekoälyratkaisujen toimivuuteen liittyen. Haasteet liittyvät erityisesti heterogeenisuuteen, skaalautuvuuteen, dynaamisuuteen ja tietoturvaan. Heterogeenisuudella tarkoitetaan, että IoT-järjestelmät koostuvat erilaisista laitteista, joilla on vaihtelevat laskenta-, tallennus- ja tehorajoitteet. Esimerkkejä tällaisista järjestelmistä ovat langattomien antureiden verkot, älykodit, teollisuuslaitokset, etäterveydenhuolto, älyliikenne ja älymaatalous. Heterogeenisuuden vuoksi resurssienhallinnan protokollien standardointi on välttämätöntä. Nykyään on useita protokollia saatavilla IoT-laitteille, kuten IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) sekä kevyet REST-pohjaiset ympäristöt, mutta ne ovat vielä toisistaan riippumattomia ja lisäävät monimutkaisuutta rajoitetuissa laitteissa. Uusien ratkaisujen tulisi pystyä varmistamaan yhteentoimivuus eri laitteiden ja protokollien kanssa. (Aouedi ym. 2024, 1280.)

IIoT-verkkojen odotetaan kasvavan merkittävästi tulevina vuosina, mikä tuo skaalautuvuuden kanssa haasteita. Kasvu lisää älykkäiden IoT-laitteiden, käyttäjien, sovellusten määrää. Edellä mainittujen lisäksi vaihtelevan laatuinen non-independent and identically distributed (non-IID) data lisääntyy merkittävästi. Näiden ongelmien ratkaisemiseksi uudentyyppisten integroitujen oppimiskehysten käyttöönottojen on oltava skaalautuvia, jotta ne vastaavat eri käyttöönottoskenaarioihin. Tähän mennessä mainittujen haasteiden lisäksi verkkojen on oltava dynaamisia. Laitteiden verkkoon liittyminen ja siitä

poistuminen on tiheää, joten resurssienhallintamallien tulee mukautua jatkuviin muutoksiin. Esimerkiksi token-pohjainen autentikointi eli todentaminen on yksi mahdollinen ratkaisu tähän haasteeseen. (Aouedi ym. 2024, 1280.)

Lopuksi tarkastellaan turvallisuuteen liittyviä haasteita. IIoT-laitteet ovat usein alttiita erilaisille kyberhyökkäyksille, minkä vuoksi tietoturva on keskeinen osa järjestelmien suunnittelua. Tietoturvahaasteiden vuoksi tekoälypohjaiset oppimismallit pyrkivät suorittamaan useita tehtäviä samanaikaisesti, kuten luokittelemaan dataa, havaitsemaan poikkeamia ja salaamaan datan. Koska IoT-laitteilla on rajalliset resurssit, näiden tehtävien suorittaminen voi johtaa datan synkronointiviiveisiin ja korkeaan virrankulutukseen. Näihin haasteisiin ratkaisuna voidaan käyttää yhteistyöhön perustuvia oppimismalleja, kuten hajautettua oppimista. Näiden avulla voidaan vähentää yksittäisten laitteiden kuormitusta ja parantaa energiatehokkuutta. Lisäksi lohkoketjuteknologian hyödyntäminen mahdollistaa varmennuksen ja lisää datan luotettavuutta erityisesti pilvipohjaisissa ratkaisuissa. Kevyet salausmenetelmät, kuten ISO/IEC 29192 -standardin mukaiset tekniikat, yhdistettynä oppimismalleihin voivat parantaa käyttäjätiedon suojaa heikentämättä järjestelmän suorituskykyä. Datan jakaminen varmennettaviin osiin ja niiden todentaminen lohkoketjussa tarjoaa lisäsuojaa mahdollisia laite- tai ohjelmistovikoja vastaan. (Aouedi ym. 2024, 1280.)

2.4.2 Koneoppimismallien haasteet IIoT:ssa

Turvallisten koneoppimismallien hyödyntäminen on keskeinen osa IoT-järjestelmiä, mutta eri sovellusten vaatimukset ja järjestelmän laajuus asettavat haasteita mallien monimutkaisuudelle ja muistitarpeelle. Nykyaikaiset koneoppimismenetelmät, kuten syvät neuroverkot vaativat kehittyneitä laiteratkaisuja, kuten tehokkaita grafiikkasuorittimia ja tensorisuorittimia. Erityisesti moniulotteinen data, suurikokoiset syötteet ja matalan viiveen vaatimukset edellyttävät suurta suorituskykyä. Tulevaisuuden IIoT-ratkaisuissa on tärkeää kehittää myös kevyitä koneoppimismalleja, jotka soveltuvat resurssirajoitettuihin ympäristöihin. (Aouedi ym. 2024, 1280.) Tämä korostuu erityisesti reunalaskennassa, jossa tietoa käsitellään mahdollisimman lähellä datan syntypaikkaa (Aouedi ym. 2024, 1242).

Kevyiden koneoppimismallien kehittämisessä keskeisiä huomioon otettavia tekijöitä ovat mallin koko, monimutkaisuus ja tarkkuus. Nämä tekijät vaikuttavat suoraan järjestelmän laskentatehokkuuteen, energiankulutukseen ja sopivuuteen erityisesti resurssirajoitteisissa IIoT-ympäristöissä. Tarkkuuden parantaminen vaatii usein suurempia ja monimutkaisempia malleja. Kuitenkin kevyempiä ratkaisuja

voidaan toteuttaa erilaisilla optimointimenetelmillä, kuten kvantisoinnilla ja parametrien karsinnalla. (Aouedi ym. 2024, 1280–1281.)

2.4.3 Resurssien tasavertainen jakaminen IIoT:ssa

Suurimittakaavaiset IoT-järjestelmät ja uudet teknologiat, kuten koneoppimismallit sekä lohkoketjuteknologiat ovat tuoneet mukanaan haasteita resurssien tasavertaiseen jakamiseen. On tärkeää, että jokainen IoT-laite, sovellus tai käyttäjä saa riittävästi resursseja viestintään, laskentaan, muistinkäyttöön ja kaistanleveyteen. Tämä on kuitenkin haastavaa muun muassa infrastruktuurin rajoitusten, viestintäprotokollien ja verkon kapasiteetin ja hallintapolitiikkojen vuoksi. Lisäksi tehottomasti kerätty IoT-data voi aiheuttaa merkittäviä ongelmia, kuten epätarkkoja ennusteita, tai johtaa virheellisiin päätöksiin. Tämän kaltaisten ongelmien ratkaisemiseksi yksi vaihtoehto voisi olla päätöksenteon siirtäminen lähelle reunaa sekä hyödyntää hajautettua oppimista. (Aouedi ym. 2024, 1281.)

IIoT-järjestelmissä resurssien epätasapaino voi johtaa myös epätasaisesti jakautuneeseen dataan, mikä tarkoittaa, että osa laitteista tuottaa enemmän tai laadukkaampaa dataa kuin toiset. Tämä voi vaikuttaa mallien oppimisen tasapuolisuuteen ja luotettavuuteen. Tämän vuoksi datan tasapainottamiseen on kehitetty useita menetelmiä. Esimerkiksi data-aineistoa voidaan laajentaa tuottamalla olemassa olevasta datasta uutta dataa erilaisin muunnoksin, kuten kiertämällä, skaalaamalla tai peilaamalla alkuperäistä dataa. Toinen lähestymistapa on datan painottaminen, jossa eri näytteille annetaan painokertoimia, joiden avulla voidaan tasapainottaa ominaisuuksien, piirteiden tai luokkien jakaumaa. Kolmas ratkaisu on siirto-oppiminen, jossa pienemmällä ja tasapainoisemmalla opetusdatalla koulutettuja malleja sovelletaan uuteen aineistoon. (Aouedi ym. 2024, 1281.)

2.4.4 Taloudelliset haasteet IIoT-ympäristöissä

Yksi keskeinen haaste IIoT-järjestelmissä liittyy taloudellisiin kysymyksiin. Ensimmäinen taloudellinen haaste on infrastruktuurin käyttöönotto- ja ylläpitokustannukset. Nämä sisältävät laitteistojen, kuten antureiden, yhdyskäytävien ja muiden laitteiden asennukseen liittyvät kulut, ohjelmistot sekä verkkoyhteydet. Jos verkkoa laajennetaan tulevaisuudessa laajempaan mittakaavaan tai monimutkaisempaan ympäristöön, sen pitäminen toimintakunnossa pitkällä aikavälillä voi vaatia huomattavia inves-

tointeja. Lisäksi koneoppimispohjainen rutiinitehtävien automatisointi voi johtaa työtehtävien siirtymiseen tai poistumiseen, mikä voi vaikuttaa haitallisesti työvoimaan, erityisesti teollisilla aloilla. Toinen merkittävä taloudellinen haaste liittyy kyberturvallisuuteen. Kyberhyökkäysten ja tietomurtojen riski kasvaa samaan tahtiin kuin yhdistettyjen laitteiden määrä. Vahvan suojausmekanismin varmistaminen edellyttää lisäinvestointeja infrastruktuuriin, mikä puolestaan kasvattaa koneoppimismallien koulutukseen, testaukseen ja käyttöönottoon liittyviä kustannuksia. (Aouedi ym. 2024, 1281.)

Kolmas haaste liittyy datan omistajuuteen ja yksityisyyteen. Kun IIoT-järjestelmät tuottavat suuria määriä dataa, jota jaetaan useiden toimijoiden kesken, syntyy haasteita datan omistajuuden määrittelyssä ja yksityisyydensuojassa. Näiden kysymysten ratkaiseminen edellyttää hallintamekanismeja, joiden avulla voidaan valvoa datan omistajuutta ja suojata käyttäjien arkaluonteisia tietoja. Neljäs taloudellinen haaste liittyy standardeihin ja yhteensopivuuteen. Älykkäissä IoT-järjestelmissä hyödynnetään useita erilaisia laitteita ja teknologioita, joiden välinen sujuva yhteistoiminta edellyttää yhtenäisten standardien käyttöä. Ilman yhteisiä standardeja laitteet eivät välttämättä kykene kommunikoimaan tehokkaasti keskenään, mikä voi hidastaa tai estää järjestelmien kehittämistä ja käyttöönottoa. (Aouedi ym. 2024, 1281–1282.)

2.5 IIoT-alustan vaatimukset, standardit ja kyberturvallisuus

IIoT-ympäristöjen turvallinen ja luotettava toiminta edellyttää sekä riittävää laitteistokapasiteettia että huolellisesti valittuja viestintäprotokollia, jotka mahdollistavat datan siirron tehokkaasti ja turvallisesti. Pelkkä tekninen toteutus ei kuitenkaan riitä, sillä järjestelmien suojaaminen vaatii myös kansainvälisten standardien ja EU-säädösten noudattamista. Standardit tarjoavat viitekehyksen riskienhallintaan ja turvallisuuskäytäntöihin, kun taas sääntelyt asettavat yrityksille sitovia velvoitteita tietoturvan ja datan hallintaan. Teknisten suojauskeinojen, standardien ja sääntelyn yhdistelmä muodostaa kokonaisuuden, jonka avulla IIoT-ratkaisut voidaan toteuttaa turvallisesti, skaalautuvasti ja kestävästi.

2.5.1 Tekniset laitteistovaatimukset

IIoT:n arkkitehtuurissa laitteistolla on keskeinen merkitys, sillä se muodostaa perustan sekä tiedonkeruulle että datan prosessoinnille. Laitteiden on usein toimittava ympäristöissä, joissa fyysiset olosuhteet voivat olla erittäin haastavat. IoT-reunalaitteet ja reunasolmut voivat olla käytössä vuosia tai jopa

vuosikymmeniä. Niiden on kestettävä lämpötilavaihteluita, kosteutta, pölyä, tärinää ja sähkömagneettisia häiriöitä. Tämä asettaa laitteille korkeammat vaatimukset kuin perinteisille IT-infrastruktuurin laitteille. Lisäksi energiatehokkuus on kriittinen tekijä etenkin kentällä toimivissa antureissa, jotka ovat usein paristokäyttöisiä. Teollisuusympäristöissä voidaan hyödyntää Power over Ethernet (PoE) -ratkaisuja, kun taas kenttäolosuhteissa voidaan käyttää energian keruuseen perustuvia tekniikoita, jotka mahdollistavat pienten elektroniikkakomponenttien toiminnan ilman ulkoista virtalähdettä. Virrankulutukseen vaikuttavat laitteistotason lisäksi myös antureiden tyyppi, näytteenottotaajuus ja käytetyt viestintäprotokollat, minkä vuoksi kompromissi suorituskyvyn ja energiatehokkuuden välillä on olennainen osa laitevalintoja. (Desbiens 2023, 189–190.)

Koska IIoT-ratkaisujen elinkaari on pitkä, laitteistokomponenttien saatavuus ja toimitusketjun vakaus ovat merkittäviä vaatimuksia. Yksi laite- ja komponenttivalinnan haasteista on se, että uudet laitteistoalustat tai anturit voivat vaatia ohjelmiston uudelleenkirjoittamista. Tämän vuoksi usein suositaan valmistajia, joilla on pitkäaikainen tuotetuki. Valitettavasti toimittajien tuotannossa ja yrityksen strategioissa voi tapahtua muutoksia, eikä saatavuus ja pitkäikäinen tuotetuki ole välttämättä taattua. Vaihtoehtoisesti voidaan käyttää avoimen laitteiston ratkaisuja (open hardware), jotka mahdollistavat yhteisölähtöisen jatkokehityksen. Toisena vaihtoehtona voidaan käyttää hybridimallia, jossa käytetään sekä luotettavien toimittajien että avoimen laitteiston ratkaisuja. (Desbiens 2023, 190–191.)

Käyttötapauksen erityispiirteet määrittelevät ympäristövaatimusten lisäksi IIoT-laitteiston toiminnalliset vaatimukset. Turvallisuuskriittisissä reaaliaikaisissa sovelluksissa, joissa anturidataa käsitellään päätöksenteon tukena, tarvitaan luotettavia vasteaikoja muistin ja verkon käytölle sekä luotettavaa keskeytyskäsittelyä. Käytettävissä olevat verkkoteknologiat ja energialähteet vaikuttavat suoraan laitevalintoihin ja järjestelmän kykyyn vastata sovelluksen tarpeisiin. Lisäksi ratkaisut eroavat sen mukaan, painotetaanko kustannustehokkuutta vai kestävyyttä. Massamarkkinoiden yleisratkaisut, kuten Raspberry Pi -korttitietokoneet, tarjoavat edullisen ja joustavan vaihtoehdon, mutta ne eivät sovellu vaativiin ympäristöihin yhtä hyvin kuin teollisuuteen suunnitellut, vahvistetut komponentit. (Desbiens 2023, 191–192.)

Kentälle sijoitetut IoT-laitteet, yhdyskäytävät ja reunasolmut ovat alttiimpia hyökkäyksille kuin pilvi tai perinteinen IT-infrastruktuuri, sillä ne voivat joutua fyysisen manipuloinnin tai jopa varkauden kohteeksi. Varastettua laitetta voidaan käyttää esiintymään luotettavana laitteena tai sen sisältämä data voi paljastaa liikesalaisuuksia tai asiakastietoja, mikä korostaa kokonaisvaltaisen suojausstrategian tarvetta. Laitteet voidaan suojata fyysisesti sijoittamalla ne murtoa kestäviin sekä ympäristöolosuhteita

vastaaviin IP-luokiteltuihin koteloihin, jotka tarjoavat suojaa pölyltä, nesteiltä ja korroosiolta. Yksinkertaisimmillaan myös käyttämättömien porttien ja liitäntöjen poistaminen tai peittäminen vähentää hyökkäyspintaa ja pienentää riskiä laitteen kaappaamisesta. (Desbiens 2023, 192–193.)

2.5.2 Kyberturvallisuus, standardit ja sääntelykehys

Koska IIoT yhdistää toimintaturvallisuuden ja tietoturvan vaatimukset, pelkät tapauskohtaiset tekniset ratkaisut eivät riitä. Tarvitaan yhteinen kieli, yhtenäinen turvallisuuden taso ja toimintatapoja määrittävä standardiperusta sekä sitova EU-sääntely, joka varmistaa vaatimustenmukaisuuden ja yhdenmukaiset käytännöt. Seuraavaksi käsitellään keskeisimmät viitekehykset ja säädökset sekä niiden merkityksen IIoT-arkkitehtuurin suunnittelulle ja operoinnille.

Teollisuusverkoissa lisääntyneet laitteet lisäävät hyökkäyspinta-alaa ja altistavat verkot erilaisille kyberuhille. Erityisesti lisääntyneet IIoT-järjestelmät ja niiden haavoittuvuudet altistavat jopa kriittistä infrastruktuuria. Onnistunut hyökkäys voi johtaa luvattomaan käyttöön, tietomurtoihin sekä toiminnallisiin häiriöihin. Näiden torjuminen edellyttää vahvoja puolustusmekanismeja haittaohjelmia, kiristysohjelmia sekä muita edistyneitä kyberhyökkäyksiä vastaan. Haasteisiin on vastattava kokonaisvaltaisella lähestymistavalla, joka kattaa turvallisuuden verkkoarkkitehtuureissa, salausprotokollien hyödyntämisen sekä ennakoivat uhkien havaitsemismekanismit IIoT-ympäristössä. Lisäksi on tärkeää varmistaa datan suojaus keruun, siirron ja tallennuksen aikana. Teollisissa IIoT-ympäristöissä on erityisen tärkeää turvata arkaluonteisen tiedon yksityisyys, erityisesti tiedon keruun, tallennuksen ja jakamisen vaiheissa. (Krishnan, Kalilulah, Indu, Gokuldhev, Deepa & Sushma 2024.)

Teollisuuden toimijoita ohjaavat lukuisat kansainväliset standardit ja viitekehykset, joiden tavoitteena on yhtenäistää turvallisuuden tasoa eri toimialoilla ja tarjota työkaluja riskienhallintaan. Standardien tarkoituksena on ohjata organisaatioita parhaissa tietoturvakäytännöissä, mutta ne harvoin säätelevät toteutuksia. Ratkaisu voi olla standardien mukainen, mutta se ei tarkoita, että turvallisuustaso olisi välttämättä optimaalinen tai ratkaisut olisivat vapaita haavoittuvuuksista. Kuitenkin IIoT-järjestelmissä on tavoitteena suojata saatavuus, eheys ja luottamuksellisuus vaatimusten mukaisesti. Turvallisen ympäristön toteuttamisessa tulisi tunnistaa uhat ja haavoittuvuudet sekä toteuttaa vastatoimet hyväksytyn riskitason mukaisesti. Tämän vuoksi teollisuudessa on kehitetty laajoja standardisarjoja, kuten IEC 62443, jotka tarjoavat systemaattisen viitekehyksen riskienhallintaan niin organisaatio-, järjestelmä- kuin komponenttitasolla. (Industry IoT Consortium 2023, 147.)

2.5.2.1 Kyberturvallisuusstandardit ja viitekehykset (IEC 62443, IISF ja NIST SP 800-82)

IEC 62443 on keskeinen teollisuuden kyberturvallisuuden perusviitekehys. Se ei ole yksittäinen standardi, vaan kokonainen standardisarja, joka on jaettu neljään ryhmään. Ensimmäinen ryhmä *General* määrittelee yhteisen terminologian, viitekehykset ja mittarit, joita muut osat hyödyntävät. Toinen ryhmä *Policies & Procedures* käsittelee politiikkoja ja menettelyjä, joilla organisaatiot voivat rakentaa tehokkaita kyberturvallisuusohjelmia. Kolmas ryhmä *System* kattaa kyberturvallisuusteknologiat, suunnittelumenetelmät, arviointitavat, turvallisuusvaatimukset ja varmuustasot. Neljäs ryhmä *Component* keskittyy vaatimuksiin turvalliselle kehityksen elinkaarelle sekä komponenttikehitykselle. Näin IEC 62443 tarjoaa kokonaisvaltaisen mallin, jonka avulla voidaan suojata teollisuuden automaatio- ja ohjausjärjestelmät sekä hallinnollisella että teknisellä tasolla. (Industry IoT Consortium 2023, 148.)

Vaikka IEC 62443 tarjoaa vahvan teknisen ja hallinnollisen perustan, sen rinnalle on kehitetty myös kokonaisvaltaisempia viitekehyksiä, kuten Industrial Internet Security Framework (IISF) (Industry IoT Consortium 2023, 147). Se tuo laajemman näkökulman, jossa otetaan huomioon teknologian lisäksi myös prosessit ja ihmiset. IISF määrittelee keskeiset ominaisuudet, joiden varaan IIoT-järjestelmien luotettavuus rakentuu: turvallisuus, suojaus, luotettavuus, vikasietoisuus ja yksityisyys. Näitä kutsutaan keskeisiksi järjestelmäominaisuuksiksi, joiden varmistaminen on edellytys koko järjestelmän luotettavuudelle. IISF painottaa riskianalyysin merkitystä vaarojen tunnistamisessa ja tapahtumien ennaltaehkäisyssä sekä korostaa, että jokainen ominaisuus on turvattava omalla tavallaan. (Industry IoT Consortium 2023, 15–18.) Turvallisuuden varmistamisen on ulotuttava koko järjestelmän elinkaareen, aina toimitusketjusta käytön aikaisiin prosesseihin ja käytöstä poistoon saakka. Lisäksi turvallisuus ei ole pelkkä tekninen ratkaisu, vaan se vaatii myös hallinnon, prosessien ja henkilöstön jatkuvaa yhteistyötä. (Industry IoT Consortium 2023, 41–42.)

IEC 62443 ja IISF muodostavat IIoT-järjestelmien kansainvälisten standardien perustan (Industry IoT Consortium 2023, 15–17, 147). Näiden lisäksi yhdysvaltalaisen The National Institute of Standards and Technologyn (NIST) julkaisema toinen revisio NIST SP 800-82 antaa ohjeita Industrial Control Systems (ICS) -ympäristöjen, Supervisory Control and Data Acquisition (SCADA) -valvonta- ja tiedonkeruujärjestelmien, Distributed Control System (DCS) -hajautettujen ohjausjärjestelmien ja muiden ohjausjärjestelmien, kuten ohjelmoitavien logiikkaohjainten (PLC) turvallisuuden parantamiseen. Oppaassa otetaan huomioon kyberturvallisuuden lisäksi myös suorituskyky-, luotettavuus- ja turvallisuusvaatimukset, jotka ovat keskeisiä kriittisessä infrastruktuurissa. NIST SP 800-82 tietoturvakontrollit

vastaavat muita NIST:n suosituksia, kuten SP 800-53:ssa esiteltyjä liitovaltion tietojärjestelmille ja organisaatioille tarkoitettuja tietoturvatoimenpiteitä. (Industry IoT Consortium 2023, 149.)

2.5.2.2 NIS2-direktiivi (EU 2022/2555)

Kansainväliset standardit, kuten IEC 62443, tarjoavat viitekehyksen vapaaehtoisille käytännöille, mutta EU-tason sääntely asettaa yrityksille sitovia velvoitteita, joilla on keskeinen merkitys myös IIoT-järjestelmien kyberturvallisuuden varmistamisessa. NIS2-direktiivin (EU 2022/2555) mukaiset velvoitteet astuivat Suomessa voimaan 8.4.2025 kyberturvallisuuslain myötä (Traficom 2025). Se velvoittaa kriittisten toimialojen toimijoita, kuten valmistavaa teollisuutta, energia- ja vesihuoltoa, toteuttamaan asianmukaiset ja oikeasuhteiset tekniset, operatiiviset ja hallinnolliset toimenpiteet. Näiden tavoitteena on hallita riskejä, joita heidän toiminnoissaan tai palvelujensa tarjoamisessa kohdistuu heidän käyttämiin verkko- ja tietojärjestelmien turvallisuuteen. (Directive (EU) 2022/2555, Art. 21; Annex I–II.)

Toimenpiteiden oikeasuhteisuus määritetään riskinarvioinnin perusteella. Toimijoiden on pyrittävä estämään sekä minimoimaan poikkeamien vaikutukset palvelujen käyttäjiin ja muihin palveluihin. NIS2 edellyttää, että riskienhallintatoimet kattavat muun muassa toimitusketjun turvallisuuden sekä toimijoiden ja heidän toimittajiensa tai palveluntarjoajiensa väliset suhteet. Lisäksi ne koskevat verkko- ja tietojärjestelmien hankintaa, kehittämistä ja ylläpidon turvallisuutta, johon sisältyy myös haavoittuvuuksien hallinta ja julkistaminen. Direktiivin mukaisiin toimenpiteisiin sisältyvät muun muassa riskianalyysit ja tietojärjestelmien turvallisuutta koskevat politiikat, poikkeamien käsittely sekä toiminnan jatkuvuuden varmistaminen varmuuskopioinnin, palautumissuunnittelun ja kriisinhallinnan avulla. Lisäksi NIS2 edellyttää organisaatioita arvioimaan kyberturvallisuusriskien hallintatoimien tehokkuutta, perustason kyberhygieniakäytäntöjen ylläpitämistä ja henkilöstön kouluttamista. Toimenpiteisiin sisältyvät myös kryptografian ja salauksen käyttö, henkilöstöturvallisuus, pääsynhallinnan periaatteet ja omaisuudenhallinta sekä tarvittaessa monivaiheinen tai jatkuva todennus, suojattu viestintä ja hätäviestintäjärjestelmien käyttö. (Directive (EU) 2022/2555, Art. 21.)

2.5.2.3 Data Act – EU:n datasäädös (EU 2023/2854)

EU:n datasäädös (Data Act) täydentää kyberturvallisuussääntelyä säätelemällä datan käyttöä, siirrettävyyttä ja yhteentoimivuutta EU:n sisällä (Regulation (EU) 2023/2854). Datasäädöstä alettiin soveltaa EU:ssa 12.9.2025 (Euroopan komissio 2024). Sen tavoitteena on vastata datavetoisen talouden haasteisiin, joissa erityisesti verkkoon liitettyjen tuotteiden ja palvelujen yleistyminen on lisännyt datan määrää ja arvoa. Johdantokappaleiden mukaan datan jakamista vaikeuttavat muun muassa kannustimien puute, epäselvyys oikeuksista ja velvollisuuksista. Lisäksi haasteena ovat kustannukset teknisten rajapintojen toteuttamisessa, datan hajanaisuus ja yhteentoimivuuden standardien puute. Nämä esteet vaikeuttavat erityisesti pk-yritysten pääsyä dataan ja hidastavat digitaalisen kilpailukyvyn kehittymistä, jos data on yhden toimijan hallussa järjestelmässä, tai jos datan tai datapalvelujen yhteentoimivuus on puutteellista. Säädöksen tavoitteena on luoda yhdenmukainen EU-tason kehys, joka määrittelee käyttöoikeudet ja edellytykset tuotteen tai palvelun tuottaman datan käyttöön. (Regulation (EU) 2023/2854.)

Datasäädöksen artikloissa konkretisoidaan, mitä velvoitteita säädös asettaa. Säädöksen mukaan muun muassa käyttäjällä on oikeus saada käyttöönsä tuotteen tai palvelun käytön tuloksena syntyvää dataa. Käyttäjällä on myös oikeus asettaa se kolmansien osapuolten saataville. Datasäädöksen soveltamisalaan kuuluu raakamuotoinen jalostamaton data, toisin sanoen lähde- tai primaaridata sekä sellainen esikäsitelty data, jonka käsittely parantaa datan ymmärrettävyyttä ja käytettävyyttä. Esikäsittelyksi ei kuitenkaan luokitella käsittelyä, joka vaatii datan haltijalta merkittäviä investointeja esimerkiksi datan puhdistamiseksi tai muuntamiseksi. Soveltamisalaan kuuluvalla primaaridatalla tarkoitetaan dataa, joka syntyy automaattisesti ilman jalostamista. Tällaista on esimerkiksi antureista tai yhteen liitetyistä anturiryhmistä kerätyt tiedot, jotka kuvaavat fyysisiä suureita, kuten lämpötilaa, painetta, virtausnopeutta, ääntä, pH-arvoa, nestetasoa, asentoa, kiihtyvyyttä tai nopeutta. (Regulation (EU) 2023/2854.)

Varsinaisen datan lisäksi soveltamisalaan kuuluu asiaankuuluva metadata, kuten peruskonteksti tai aikaleima. Säädös helpottaa myös pilvipalvelujen välistä vaihtoa. Lisäksi sen mukaan tulisi ottaa käyttöön turvatoimia luvattoman pääsyn estämiseksi ei-henkilökohtaiseen dataan ja asettaa velvoitteita yhteentoimivuusstandardien kehittämiselle. Soveltamisala kattaa sekä henkilö- että ei-henkilötiedot. Erityisesti teollisuudessa on keskeistä, että myös ei-henkilökohtainen prosessidata kuuluu sääntelyn piiriin. Käytännön tasolla tämä velvoittaa yrityksiä tarjoamaan käyttäjien saataville helposti, turvallisesti ja maksutta heidän tuottamansa datan kattavassa, jäsennellyssä ja yleisesti käytetyssä koneellisesti luettavassa muodossa. (Regulation (EU) 2023/2854.)

2.5.2.4 Cyber Resilience Act – EU:n kyberkestävyyssäädös (EU 2024/2847)

Cyber Resilience Act (CRA) luo vähimmäisedellytykset digitaalisia elementtejä sisältävien tuotteiden tietoturvalliselle kehittämiselle. Asetuksen tarkoituksena on varmistaa, että laitteisto- ja ohjelmistotuotteet saatetaan markkinoille vähemmillä haavoittuvuuksilla ja että valmistajat huomioivat tietoturvan vakavasti tuotteen koko elinkaaren ajan. Lisäksi asetuksen tavoitteena on lisätä markkinoiden avoimuutta esimerkiksi määrittämällä velvoitteet tuotteiden tukiajan ilmoittamisesta. CRA asettaa horisontaaliset kyberturvallisuusvaatimukset kaikille digitaalisia elementtejä sisältäville tuotteille, mukaan lukien sekä ohjelmistot että laitteistot. Soveltamisalaan kuuluu myös etäkäsittely, kuten pilvipalvelut, silloin, kun ilman niitä tuote ei voisi toimia. Asetusta sovelletaan kaupallisessa toiminnassa, riippumatta siitä toimitetaanko tuotteet maksua vastaan vai ilmaiseksi. (Regulation (EU) 2024/2847.)

CRA tuli voimaan 10.12.2024, ja sen soveltaminen alkaa vaiheittain: ilmoitettuja laitoksia koskevat velvoitteet 11.6.2026, haavoittuvuuksista raportointi 11.9.2026 ja asetuksen liitteen I olennaiset kyberturvallisuusvaatimukset 11.12.2027 alkaen. Olennaisiin vaatimuksiin sisältyvät muun muassa turvalliset oletusasetukset ja automaattiset päivitykset, luvattoman pääsyn estäminen, datan luottamuksellinen säilytys ja minimointi sekä keskeisten toimintojen turvaaminen. Valmistajan on suoritettava vaatimustenmukaisuuden arviointi ja laadittava EU-vaatimustenmukaisuusvakuutus. Tämän jälkeen tuotteeseen on kiinnitettävä CE-merkintä osoituksena siitä, että se täyttää asetuksen vaatimukset. Lisäksi valmistajan on raportoitava aktiivisesti hyödynnetyistä haavoittuvuuksista ja vakavista poikkeamista CSIRT-yksikölle ja ENISA:lle. Merkittävä muutos ohjelmistossa tai laitteessa voi johtaa uuteen vaatimustenmukaisuuden arviointiin. Tuotteella tarkoitetaan mitä tahansa digitaalisia elementtejä sisältävää ohjelmistoa tai laitteistoa, ja valmistajalla sitä toimijaa, joka kehittää tai valmistaa tuotteen ja saattaa sen markkinoille omalla nimellään tai tavaramerkillään. (Regulation (EU) 2024/2847.)

2.5.3 Tekniset suojauskeinot IIoT:ssa – yleiskuva

Vaikka kansainväliset standardit ja EU-tason sääntely määrittävät raamit IIoT-ympäristöjen turvallisuudelle, todellinen kyberturva syntyy käytännön teknisistä suojausratkaisuista, jotka kohdistuvat eri arkkitehtuurikerroksiin. IIoT-järjestelmät voidaan jäsentää useaan arkkitehtuuritasoon ja tutkimusten mukaan tietoturvapalveluita tarvitaan kaikissa tasoissa aina reunalla olevista laitteista pilveen asti (Gebremichael, Ledwaba, Eldefrawy, Hancke, Pereira, Gidlund & Åkerberg 2020, 152352–152353). Laajamittainen IIoT:n käyttöönotto tuo mukanaan myös merkittäviä kyberturvallisuushaasteita, sillä

verkkoon liitetyt laitteet ja järjestelmät ovat yhteydessä toisiinsa. Tämä lisää hyökkäyspinta-alaa ja altistaa kriittiset toiminnot erilaisille kyberuhille. Näiden haasteiden hallinta edellyttää kokonaisvaltaista lähestymistapaa, jossa yhdistyvät verkkoarkkitehtuurien turvallinen suunnittelu, salausteknologioiden hyödyntäminen sekä uhkien ennakoiva havaitseminen IIoT-ympäristöissä. (Krishnan ym. 2024.)

2.5.3.1 Haavoittuvuudet kerroksittain

IoT-järjestelmissä haavoittuvuudet voivat ilmetä eri arkkitehtuuritasoilla, minkä vuoksi riskien hallinta edellyttää kerroskohtaista tarkastelua. Lähteiden mukaan suojausta tarvitaan yhtä lailla reunalaitteissa, verkossa kuin sovelluskerroksessakin (Gebremichael ym. 2020, 152352–152353). Laitteet ja reunakerros ovat alttiita erityisesti fyysiselle manipuloinnille, resurssirajoitteiden aiheuttamille heikkouksille sekä riittämättömälle autentikoinnille eli todentamiselle ja salaukselle. Hyökkääjät voivat esimerkiksi peukaloida tai varastaa laitteita saadakseen luvattoman pääsyn tietoihin tai järjestelmiin. IIoT-laitteiden resurssirajoitteisuus puolestaan vaikeuttaa vahvojen kryptografisten ratkaisujen käyttöönottoa, jolloin viestiliikenteen salaus voi jäädä riittämättömäksi. Tämä voi johtaa tietojen sieppaukseen tai salakuunteluun. Lisäksi heikko autentikointi voi johtaa luvattomaan käyttöön sekä koko järjestelmän vaarantumiseen. (Rakshe & Dongre 2024, 2–3.)

Verkkokerroksessa keskeisiä uhkia ovat erityisesti palvelunestohyökkäykset (Denial of Service, DoS), liikenteen sieppaus, muokkaus ja salakuuntelu. Hyökkääjät voivat esimerkiksi kuormittaa verkkoa liiallisella liikenteellä lamauttaakseen sen toiminnan. Hyökkääjät voivat myös toteuttaa väliintulohyökkäyksen (Man-in-the-Middle, MitM), jossa he sieppaavat ja muokkaavat osapuolten välistä tietoliikennettä, vaarantaen näin viestien luottamuksellisuuden ja eheyden. Lisäksi verkko on altis osoitehuijauksille ja reititykseen liittyville hyökkäyksille. Näitä ovat esimerkiksi ARP-huijaukset (Address Resolution Protocol spoofing) ja IP-huijaukset (Internet Protocol spoofing), joissa hyökkääjä voi ohjata liikennettä omaan hallintaansa tai esiintyä luotettavana osapuolena. Näiden lisäksi riittämättömät tai vanhentuneet salausprotokollat voivat johtaa siihen, että liikenteen tietoja voidaan siepata ja hyödyntää luvattomasti. (Rakshe & Dongre 2024, 3.)

Sovelluskerroksessa tyypillisiä haavoittuvuuksia ovat injektiohyökkäykset (Injection Attacks), XSS-hyökkäykset (Cross-Site Scripting, XSS), CSRF-hyökkäykset (Cross-Site Request Forgery, CSRF) sekä luvaton pääsy laitteistoon. Injektiohyökkäyksissä hyökkääjät syöttävät haitallista koodia, kuten

SQL- ja käyttöjärjestelmäkomentoja hyödyntääkseen sovelluksen haavoittuvuuksia. XSS-hyökkäyksissä lisätään haitallisia komentosarjoja muiden käyttäjien katselemille verkkosivuille vaarantaen heidän tietoturvansa. CSRF-hyökkäyksissä hyökkääjät huijaavat käyttäjiä suorittamaan toimia ilman käyttäjien suostumusta, esimerkiksi muuttamaan tilin asetuksia hyödyntämällä heidän autentikoitua yhteyttänsä. Näiden riskien hallinta edellyttää sovelluskerroksessa erityisesti yksityiskohtaista pääsynhallintaa, vahvaa todennusta ja valtuutusta. Lisäksi vaaditaan API-rajapintojen suojaamista luvattomalta käytöltä ja tietovuodoilta, säännöllistä haavoittuvuuksien hallintaa sekä datan salausta niin siirrossa kuin levossa. (Rakshe & Dongre 2024, 2–3.)

2.5.3.2 Konkreettiset tekniset suojausmenetelmät

Edellä esitetyt esimerkit osoittavat, että IIoT-arkkitehtuurin jokainen taso on altis erilaisille uhille, jotka voivat vaarantaa järjestelmän toiminnan ja tietoturvan. Näiden uhkien hallinta edellyttää systemaattisia teknisiä suojauskeinoja, kuten salausprotokollia, autentikointia, valtuutusta ja uhkien ennakoivaa havaitsemista (Krishnan ym. 2024). Salausprotokollat muodostavat perustan turvalliselle tiedonsiirrolle. Salausprotokollia, kuten Transport Layer Security (TLS) ja sen muunnos Datagram Transport Layer Security (DTLS), voidaan hyödyntää tiedonsiirron salaamiseen niin yhteyspohjaisissa kuin viestipohjaisissa protokollissa (Gebremichael ym. 2020, 152357). IIoT-ympäristöissä haasteena on kuitenkin, että monet teollisuusprotokollat eivät vielä tue riittävää autentikointia. Näissä tapauksissa puutteelliset protokollat voidaan kapseloida TLS:n tai muiden alemman kerroksen protokollien sisään, jolloin saavutetaan tarvittavat turvallisuusominaisuudet, kuten autentikointi. (Industry IoT Consortium 2023, 90.)

Autentikointi ja valtuutus ovat keskeisiä mekanismeja, joilla varmistetaan, että vain luotetut käyttäjät ja laitteet pääsevät käsiksi järjestelmiin sekä hallitaan käyttöoikeuksia ja rajoitetaan luvattomia toimintoja. Sovelluskerroksen suojauksessa ovat lisäksi tärkeitä API-rajapintojen turvallisuus sekä luotettava pääsynhallinta. (Rakshe & Dongre 2024, 2.) Pelkkä salaus ja autentikointi eivät yksin riitä suojaamaan IIoT-järjestelmiä. Tarvitaan lisäksi mekanismeja, joilla havaitaan poikkeamia ja reagoidaan niihin reaaliajassa. Tätä varten hyödynnetään erityisesti tunkeutumisen havaitsemisjärjestelmiä (Intrusion Detection Systems, IDS), joiden avulla voidaan tunnistaa mahdollisia uhkia ja luvattomia käyttöyrityksiä IIoT-verkoissa. Kehittyneet havaitsemisjärjestelmät hyödyntävät myös koneoppimista ja käyttäytymisanalytiikkaa poikkeamien havaitsemiseen, mikä mahdollistaa nopean reagoinnin ja riskien lieventämisen. (Krishnan ym. 2024.)

IIoT-ympäristöissä suositellaan lisäksi syvyyssuuntaista puolustusstrategiaa (Defense in Depth, DiD), jossa useat suojausmekanismit sijoitetaan eri kerroksiin ja täydentävät toisiaan, mikä vähentää yksittäisten haavoittuvuuksien kriittisyyttä (Industry IoT Consortium 2023, 51). Lisäksi tärkeä osa ennaltaehkäisevää suojautumista on sovellusten säännöllinen tarkastaminen haavoittuvuuksien varalta ja niiden välitön korjaaminen, jotta järjestelmään ei jää hyökkäyksille altistavia heikkouksia (Rakshe & Dongre 2024, 2).

2.5.3.3 Laitetason suojaus (HSM/TPM)

Turvallisuuden toteuttaminen myös laitteistoissa tarjoaa erityisiä etuja ja haittoja IIoT:ssa. Erikoistunut fyysiseltä manipuloinnilta suojattu laitteisto tarjoaa korkeamman luottamustason erityisesti kryptografisten avainten ja operaatioiden suojaamisessa. Laitteistopohjainen turvallisuus ei kuitenkaan ole ongelmatonta, sillä se tuo mukanaan lisäkustannuksia sekä hallinta- ja päivitysprosessien monimutkaisuutta. Näistä rajoitteista huolimatta laitteistopohjaisia ratkaisuja, kuten Hardware Security Module (HSM) ja Trusted Platform Module (TPM) -moduuleja, hyödynnetään yhä useammin IIoT-ympäristöjen suojaamisessa, koska ne muodostavat vahvan perustan avainten hallinnalle, todennukselle ja luottamuksen juurelle (Industry IoT Consortium 2023, 71–73).

HSM on kovennettu ja eristetty laitekomponentti, jota käytetään tietoturvatoimintoihin, kuten kryptografisten avainten tallennukseen ja elinkaaren hallintaan sekä avainten luomiseen ja derivointiin vahvaa autentikointia varten (Industry IoT Consortium 2023, 72). Tässä yhteydessä avainten derivoinnilla tarkoitetaan, että avaimen derivointi toteutetaan avainten derivointifunktiolla (Key-Derivation Function, KDF), joka ottaa syötteenä salaisen avaimen ja muuta dataa ja tuottaa tästä avainmateriaalia kryptografisia algoritmeja varten (Chen 2022, 4). HSM mahdollistaa laitteiden turvallisen käyttöönoton ja viestikanavien autentikoinnin. Sitä hyödynnetään myös esimerkiksi laiteohjelmistojen päivityksissä, avainten kierrätyksessä ja uudelleenkonfiguroinnissa. Lisäksi HSM voi toimia luottamuksen juurena (Root of Trust, RoT) luomalla ja säilyttämällä yksityiset avaimet suojatuissa elementeissä, joita on kolme tyyppiä: erillinen komponentti, prosessoriin sulautettu tai flash-muistiin perustuva ratkaisu. (Industry IoT Consortium 2023, 72.)

TPM on yleinen laitteistopohjaisen turvallisuuden toteutus, joka voidaan ottaa käyttöön joko erillisenä laitekomponenttina tai prosessoriin sulautettuna osana Root of Trust (RoT) -arkkitehtuuria. Toteutus

voi tapahtua erillisenä laitesiruna tai erillisessä laitteistokomponentissa, joka sijaitsee samalla fyysisellä piirillä kuin keskusyksikkö, mutta eristetyssä alueessa. TPM-standardin mukainen komponentti suorittaa kryptografisia operaatioita erillään keskusyksiköstä, ja sitä käytetään erityisesti avainten luomiseen, säilyttämiseen, allekirjoitukseen ja datan sinetöintiin eli suojaamiseen. (Industry IoT Consortium 2023, 73.)

2.5.3.4 Käynnistysketjun eheys

Laitteen käynnistysprosessi on kriittinen vaihe, sillä se alustaa laitteen keskeiset laitteistokomponentit ja käynnistää käyttöjärjestelmän. Luottamus täytyy luoda jo käynnistysympäristössä, sillä muuhun ohjelmistoon ei voida luottaa ennen kuin käynnistysketjun eheys on varmistettu. Käynnistysprosessin mittaaminen mahdollistaa manipulaation havaitsemisen ja tarjoaa suojaa esimerkiksi haittaohjelmia, viruksia ja matoja vastaan. Käynnistysprosessin suojaamiseksi on käytössä useita menetelmiä: luotettu käynnistys ja mitattu käynnistys (Trusted Boot / Measured Boot), vahvistettu käynnistys (Verified Boot) sekä suojattu käynnistys (Secure Boot). (Industry IoT Consortium 2023, 79.)

Trusted Boot ja Measured Boot luovat ketjun, jossa jokainen komponentti mittaa seuraavan käynnistysvaiheen ennen sen suorittamista. Tämä luo luottamusketjun (Chain of Trust) -periaatteen ja mahdollistaa myös etätodentamisen ja myöhemmän päätepisteiden luotettavuuden arvioinnin. Verified Boot on Trusted Bootin muoto, jossa komponentit allekirjoitetaan, mutta niitä ei mitata. Jos tarkistus epäonnistuu, käynnistys pysähtyy. Secure Boot viittaa menetelmään, jossa käynnistysprosessi keskeytetään kokonaan, jos järjestelmän tila ei ole luotettu. Näin käynnistyksen suojaus voi joko sallia käynnistymisen ja havaita poikkeamat tai estää käynnistymisen epäluotettavassa tilassa. (Industry IoT Consortium 2023, 79.) Näin eri menetelmät tarjoavat vaihtoehtoisia lähestymistapoja käynnistysprosessin suojaamiseen IIoT-ympäristöissä.

2.5.3.5 Ohjelmistoturvallisuuden elinkaari (ENISA)

EU:n kyberturvallisuusvirasto ENISA painottaa ohjelmistoturvallisuuden näkökulmasta *secure by design* -periaatetta, jonka mukaan turvallisuus tulee sisällyttää järjestelmiin jo suunnitteluvaiheessa. Tämä tarkoittaa muun muassa turvallisen ohjelmistokehityksen käytäntöjen (Secure Software Development Life Cycle, SSDLC) noudattamista sekä jatkuvaa tietoturvatestausta. Lisäksi ENISA korostaa

päivitysten ja korjausten hallintaa, jossa tietoturvapäivitykset on sovellettava kohtuullisessa ajassa, testattava ennen tuotantoon viemistä ja niiden lähteen on oltava luotettava. Tähän liittyy myös haavoittuvuuksien hallinta, jossa organisaatioiden tulee hankkia tietoa teknisistä haavoittuvuuksista heidän verkossaan ja järjestelmissään sekä arvioida niiden vaikutuksia ja toteuttaa tarvittavat korjaavat toimenpiteet. Ohjelmistoturvallisuudessa on lisäksi huomioitava koko elinkaari: ohjelmistot ja laitteet, jotka eivät ole enää tuettuna, on poistettava verkosta hallitusti yrityksen riskinarvioinnin mukaisesti. (ENISA 2025, 80, 91, 94, 104.)

2.6 Data-analyysi ja datan visualisointi IIoT:ssa

Data-analyysi ja visualisointi muodostavat keskeisen osan myös IIoT-ympäristöjen arkkitehtuuria, sillä niiden avulla laitteista kerätty runsas määrä dataa muutetaan ymmärrettävään ja päätöksenteon kannalta hyödylliseen muotoon. Data-analyysi tarkoittaa raakadatasta lähtevää systemaattista prosessia, jossa dataa siivotaan, tutkitaan, muokataan ja jalostetaan, jotta siitä saadaan esiin päätöksenteon kannalta hyödyllisiä havaintoja. Analyysi paljastaa yhteyksiä ja kaavoja, joita ei muuten havaittaisi, ja luo näin pohjan strategiselle päätöksenteolle datalähtöisillä toimialoilla. Datan visualisointi on olennainen osa data-analyysiä, sillä sen avulla monimutkaiset tietoaineistot voidaan esittää ymmärrettävässä ja visuaalisesti havainnollisessa graafisessa muodossa, kuten kaavioina, pylväsdiagrammeina, hajontakuvioina tai lämpökarttoina. Tämä auttaa asiantuntijoita ja laajempaa yleisöä tunnistamaan suurista datamääristä piileviä trendejä ja riippuvuuksia. (Senthil, Nagarajan & Tayong 2024, 102.)

Datan visualisointi sai alkunsa peruskartoista ja kaavioista, mutta se on kehittynyt kohti monipuolisia ja älykkäitä analytiikka-alustoja. Teknologisten ratkaisujen ansiosta on tarjolla tehokkaita visualisointialustoja eri tasoisille käyttäjille ja rooleille organisaatioissa. Nykyään visualisointi on välttämätön osa datan hyödyntämistä erityisesti silloin, kun aineistot kasvavat suuriksi ja monimutkaisiksi. Visualisointialustat eivät enää tarjoa pelkkiä staattisia näkymiä, vaan niiden avulla voidaan tunnistaa poikkeamia, ennakoida laiterikkojen riskejä ja arvioida tulevia tuotantotarpeita. Visualisointi toimii samalla ”yhteisenä visuaalisena kielenä”, joka selkeyttää yhteistyötä tiimien ja sidosryhmien välillä. Strateginen visualisoinnin käyttö tukee sekä päätöksenteon laatua että operatiivista tehokkuutta. (Senthil ym. 2024, 102.)

3 KESKEISET IOT-TEKNOLOGIAT JA ARKKITEHTUURIT

Edellisessä luvussa tarkasteltiin IoT:n ja teollisen internetin (IIoT) taustaa, sovelluskohteita ja kehityssuuntia. Tässä luvussa siirrytään tarkastelemaan aihetta teknisemmästä näkökulmasta. Käsittely keskittyy arkkitehtuurimalleihin, standardoituihin viitearkkitehtuureihin sekä IIoT:ssa yleisesti käytettyihin viestintäprotokolliin. Näiden avulla luodaan pohja työn toteutukselle, joka perustuu reunalla ja pilvessä tapahtuvaan laskentaan soveltaen Unified Namespace (UNS) -lähestymistapaa. Luvussa tarkastellaan, miten eri arkkitehtuurivalinnat vaikuttavat ratkaisujen suorituskykyyn ja skaalautuvuuteen sekä millaisia vahvuuksia ja rajoitteita protokolliin liittyy. Lisäksi käydään läpi yleisiä arkkitehtuurimalleja sekä viite- ja standardikehyksiä, sekä niiden haasteita, joihin UNS tarjoaa ratkaisun. Lopuksi syvennytään IIoT:ssa käytettyihin protokolliin, joista työn kannalta keskeisimmät, OPC UA, MQTT ja Sparkplug B, analysoidaan yksityiskohtaisesti.

3.1 IoT-arkkitehtuurien yleiskatsaus

IoT-arkkitehtuurilla tarkoitetaan kokonaisuutta, joka koostuu monimutkaisesta verkosta toisiinsa kytketyistä komponenteista, jotka työskentelevät yhdessä kerätäkseen, prosessoidakseen, analysoidakseen ja toimiakseen IoT-laitteiden tuottaman datan perusteella. IoT:n tarpeisiin on kehitetty monenlaisia sovellusarkkitehtuureja. Niiden on kyettävä toimimaan hyvin erilaisten antureiden ja toimilaitteiden kanssa, jotka eroavat toisistaan paitsi käyttämissään viestintäprotokollissa myös siinä, millaista dataa ne tuottavat ja käsittelevät. Toistaiseksi mikään näistä arkkitehtuureista ei ole osoittautunut täydelliseksi kaikille sovellusalueille tai -tilanteille. Tämä johtuu IoT:n monitieteisestä luonteesta, sillä jokainen käyttöalue asettaa omat erityisvaatimuksensa. (Dauda, Flauzac & Nolot 2024, 2.)

3.1.1 Pilvi-, reuna- ja sumuarkkitehtuurit

IoT-arkkitehtuurit koostuvat yleensä useista kerroksista, joilla jokaisella on oma ainutlaatuinen roolinsa IoT-järjestelmässä. Ne yhdistävät keskitetyn hallinnan ja hajautetun datankäsittelyn, jossa hyödynnetään pilvilaskentaa, mutta samalla siirretään osa prosessoinnista lähemmäs laitteita reuna- ja sumulaskennan avulla. Näiden mallien yhdistelmä tarjoaa joustavan tavan hallita datavirtoja eri tasoilla,

jolloin voidaan tasapainottaa reaaliaikainen vaste, resurssien kuormitus ja keskitetyn analytiikan tarpeet. Tällainen lähestymistapa tarjoaa resurssien käytön optimointia, skaalautuvuutta, nopeaa vasteaikaa ja luotettavuutta, mikä tekee mahdolliseksi IoT-sovellusten moninaisten vaatimusten täyttämisen eri toimialoilla ja käyttötapauksissa. Näin ollen pilvi-, reuna- ja sumulaskenta muodostavat keskeisen perustan nykyaikaisille IoT-arkkitehtuureille. (Dauda ym. 2024, 2–3.)

Pilvilaskenta on laskentamalli, jossa erilaiset tietotekniikkapalvelut, kuten tallennus, palvelimet, tietokannat, verkkoyhteydet, ohjelmistot ja analytiikka tarjotaan internetin välityksellä. Sen sijaan että organisaatiot omistaisivat ja ylläpitäisivät omaa laskentainfrastruktuuriaan tai datakeskuksiaan, ne voivat vuokrata pääsyn näihin palveluihin pilvipalveluntarjoajalta, kuten Amazon Web Services (AWS), Microsoft Azure tai Google Cloud Platform (GCP). IoT-laitteet voivat olla suoraan yhteydessä pilveen esimerkiksi TCP/IP-, User Datagram Protocol (UDP)/IP- ja HTTP/REST-protokollilla. Monissa tapauksissa väliin tarvitaan kuitenkin yhdyskäytävä, joka mahdollistaa tiedonsiirron. Yhdyskäytävän tehtäviin kuuluu datan kerääminen, protokollamuunnos, tietoturvan varmistaminen sekä datan esikäsittely ennen siirtoa pilvipalveluihin. Suurimmat pilvialustat, kuten AWS IoT, Azure IoT ja Google Cloud IoT, hyödyntävätkin reunalla yhdyskäytäviä laiteliitettävyyden, tietoturvan ja hallinnan varmistamiseksi. Pilvilaskenta tuo mukanaan myös haasteita, kuten tietoturvaan, yksityisyydensuojaan, verkon viiveisiin ja järjestelmäintegraatioon liittyvät ongelmat, jotka on huomioitava arkkitehtuurien suunnittelussa. (Dauda ym. 2024, 2–3.)

Reunalaskenta on jo esitelty aiemmin luvussa 2.3 IoT:n keskeisenä ilmiönä, minkä vuoksi tässä luvussa keskitytään sen rooliin osana arkkitehtuurimallia pilvilaskennan ja sumulaskennan rinnalla. Reunalaskennassa laskentaa on siirretty reunalle lähemmäksi datan syntypaikkaa, kuten yhdyskäytäville tai IoT-laitteille. Sen sijaan, että kaikki data lähetettäisiin keskitettyyn pilvipalveluun, osa prosessoinnista ja analysoinnista suoritetaan paikallisesti, mikä muun muassa vähentää viivettä ja verkkokuormaa. Reunalaitteilla on usein rajalliset resurssit laskentatehon, muistin ja energian suhteen, koska ne suunnitellaan kannettaviksi sekä energia- ja kustannustehokkaiksi vaihteleviin ympäristöihin. Tämän vuoksi niiden rooli on yleensä täydentävä suhteessa pilvipalveluihin. (Dauda ym. 2024, 2–3.)

Vaikka pilvi on tehokas resurssi suurten tietomäärien tallentamiseen ja analysointiin, se ei aina ole optimaalinen ratkaisu IoT-sovelluksille, erityisesti niille, jotka vaativat reaaliaikaista vastekykyä. Tätä varten reuna- ja pilvilaskennan väliseen kerrokseen on kehitetty laskentamalli, jota kutsutaan sumulaskennaksi. Se täydentää ja tehostaa pilvilaskentaa hajauttamalla laskentaa lähemmäs datan syntypaik-

kaa. Sumulaskenta on hajautettu laskentainfrastruktuuri, joka jakaa pilvilaskennan pienempiin aliverkkoihin ja tuo resurssit lähemmäksi verkon reunaa, lähemmäksi datalähdettä ja käyttäjälaitteita. Sumulaskenta tarjoaa hajautettua prosessointia, tallennusta ja verkkopalveluita vastaten joihinkin pilvilaskennan haasteisiin pienentämällä viivettä, parantamalla tietoturvaa, yksityisyyttä ja skaalautuvuutta. Pilvi-, reuna- ja sumulaskennan integrointi IoT:hen on parantanut merkittävästi IoT-sovellusten luomista ja hallintaa. Nämä teknologiat tarjoavat laajat laskenta-, verkko- ja tallennusresurssit, mikä mahdollistaa kehittää räätälöityjä arkkitehtuureita vastaamaan erilaisiin vaatimuksiin ja rajoituksiin. (Dauda ym. 2024, 2–3.)

3.1.2 IoT-viitearkkitehtuurit ja standardimallit

Useita viitearkkitehtuureja ja malleja on kehitetty eri instituutioiden ja organisaatioiden yhteistyönä. Ne tarjoavat perusteellisia ohjeita ja laajan valikoiman ratkaisuja IoT-järjestelmien suunnitteluun, käyttöönottoon ja hallintaan. Näiden avulla voidaan vastata eri IoT-toteutusten erityisvaatimuksiin ja helpottaa yrityksiä hyödyntämään IoT-teknologiaa omassa toiminnassaan. Lisäksi ohjeistusten avulla voidaan varmistaa, että ratkaisut ovat luotettavia, turvallisia ja tehokkaita, mikä puolestaan tukee IoT-projektien onnistumista. Yksi esimerkki tällaisesta mallista on vuonna 2012 kehitetty International Telecommunication Union-Telecommunication Standardization Sectorin (ITU-T) suositus Y.2060. Viitemalli tarjoaa jäsennellyn tavan hahmottaa IoT-ekosysteemien eri osat ja niiden väliset vuorovaikutukset. Se esitetään kerroksellisena arkkitehtuurina, joka sisältää laitekerroksen, verkkokerroksen, palvelu- ja sovellustukikerroksen sekä sovelluskerroksen. (Dauda ym. 2024, 4.)

Y.2060-mallissa laitekerros on fyysinen kerros, jossa IoT-laitteet sijaitsevat. Nämä laitteet voivat olla antureita, toimilaitteita tai teollisia koneita. Ne keräävät dataa ympäristöstä tai suorittavat toimintoja vastaanotettujen ohjauskomentojen perusteella. Verkkokerros käsittää protokollat ja teknologiat, joita käytetään IoT-laitteiden ja verkon välisen yhteyden muodostamiseen, kuten Wi-Fi, Bluetooth, Zigbee ja matkapuhelinverkot. Palvelu- ja sovellustukikerros on arkkitehtuurin osa, joka käsittelee datankäsittelyä, laitehallintaa, tietoturvaa sekä viestintää laitteiden ja sovellusten välillä. Se sisältää usein reunalaskentaresursseja datan prosessointiin lähempänä lähdettä vähentääkseen viivettä ja kaistanleveyden käyttöä. Tässä kerroksessa käytettyjä protokollia ovat esimerkiksi MQTT ja CoAP (Constrained Application Protocol). Sovelluskerros koostuu sovelluksista tai palveluista, jotka hyödyntävät IoT-laitteiden

keräämää dataa. Sovellukset voivat vaihdella yksinkertaisista anturidatan visualisoinneista monimutkaisiin analytiikkoihin, kuten ennakoivan kunnossapidon järjestelmiin, tekoälypohjaiseen päätöksentekoon tai älykaupunkiratkaisuihin. (Dauda ym. 2024, 4–5.)

IoT-sovellusten arkkitehtuuri voi vaihdella merkittävästi käyttötapauksen, toimialan, skaalautuvuusvaatimusten ja käytettävissä olevien resurssien mukaan. Tehokkaiden arkkitehtuurien suunnittelussa ratkaisevaa on tasapaino skaalautuvuuden, turvallisuuden, viiveen ja kustannustehokkuuden välillä. Lisäksi tulee ottaa huomioon yhteentoimivuus, tietoturva, datanhallinta, reaaliaikainen prosessointi ja autonomia. Koska mikään yksittäinen arkkitehtuuri ei sovellu kaikkiin tilanteisiin, on kehitetty useita yleisiä lähestymistapoja, joista jokaisella on etunsa ja kompromissinsa. Nämä perustuvat joko keskitettyyn tai hajautettuun arkkitehtuurilähestymistapaan. Keskitetyssä mallissa kaikki datankäsittely- ja päätöksentekotehtävät suoritetaan keskitetyssä paikassa, kuten pilvipalvelimella tai datakeskuksessa. Hajautetussa mallissa prosessointitehtävät jakautuvat useiden reunalaitteiden tai yhdyskäytävien kesken, jotka sijaitsevat lähempänä datalähdettä. Jokainen arkkitehtuuri tarjoaa ainutlaatuisia hyötyjä ja haasteita käyttötapauksen mukaan, ja ne heijastavat ympäristön vaatimuksia ja rajoitteita. (Dauda ym. 2024, 5.)

3.1.3 Teollisuuden viitearkkitehtuurit (ISA-95, RAMI 4.0)

Yksi eniten käytetyistä teollisuuden viitearkkitehtuureista, erityisesti älykkään valmistuksen alkuvaiheessa, on ollut International Society of Automationin (ISA) standardi ISA-95, jota myös kutsutaan automaatiopyramidiksi. Se tarjoaa selkeän hierarkkisen lähestymistavan automaatioon ja on laajasti käytössä, sillä se määrittää kullekin tasolle selkeät tehtävät ja aikajänteet sekä kommunikaatioprotokollat eri kerrosten väliseen tiedonsiirtoon. Tasolla 0 (tuotantoprosessi) päätoimijoita ovat anturit ja toimilaitteet, jotka ovat suoraan yhteydessä tuotantoresursseihin ja tuottavat suuren määrän datapisteitä, usein vain millisekuntien välein. Tavallisesti nämä anturit ja toimilaitteet on yhdistetty suoraan prosessilogiikkaohjaimeen (PLC), joka muodostaa tason 1 (havaitseminen ja valmistus). Tasolla 1 PLC vastaanottaa antureilta ja toimilaitteilta tietoa prosessista, käsittelee sen ohjelmoitujen sääntöjen mukaisesti ja tarvittaessa ohjaa toimilaitteita. Taso 1 voi sisältää myös Human-Machine Interface (HMI) -käyttöliittymän, jonka kautta operaattori voi tehdä muutoksia tai tarkastella anturidataa reaaliaikaisesti. (Salcher, Finck & Hellwig 2024.)

Tasolla 2 useat PLC:t yhdistyvät yhdeksi SCADA-järjestelmäksi. Sen pääpaino on resurssien tilassa ja tuottavuudessa tuotantolinjalla. Tasolla tarkkaillaan esimerkiksi toimivatko koneet normaalisti asetettujen parametrien mukaisesti. Tällä tasolla kommunikaatioprotokollien merkitys korostuu, sillä SCADA-järjestelmät sijaitsevat yleensä valvomoissa, joihin koneet lähettävät prosessidatan. Protokollat, kuten OPC UA, Modbus ja Siemens S7 tarjoavat standardoidun rajapinnan tiedonsiirtoon ja ovat vakiintuneet laajasti teollisuudessa. Yhdistetty prosessidata siirtyy seuraavalle pyramidin tasolle, joka on taso 3 (valmistustoiminta ja -hallinta). (Salcher ym. 2024.)

Tason 3 pääpaino siirtyy operatiivisesta toiminnasta strategiseen ohjaukseen. Tason ytimessä on yleensä Manufacturing Execution System (MES) -valmistuksenohjausjärjestelmä, joka vastaanottaa tilaukset yrityksen Enterprise Resource Planning (ERP) -toiminnanohjausjärjestelmältä. MES muuntaa ne tuotantotilauksiksi, tarkistaa koneiden ja resurssien saatavuuden sekä laatii tuotantoaikataulun. Valmistuneet tilaukset raportoidaan takaisin ERP-järjestelmään, joka toimii tason 4 (liiketoiminnan suunnittelu ja logistiikka) ohjelmistona. ERP tarjoaa tärkeää tietoa eri osastoille, kuten logistiikalle tuotantomääristä, laadunhallinnalle hylkyprosenteista ja kunnossapidolle kokonaisvalmistusmääristä huoltotoimenpiteiden suunnitteluun. (Salcher ym. 2024.)

ISA-95-automaatiopyramidin rinnalle älykkään valmistuksen kehittyessä teollisuuteen on kehitetty uusia viitearkkitehtuureja. Näistä yksi merkittävä on Reference Architectural Model Industrie 4.0 (RAMI 4.0), joka liittyy Teollisuus 4.0 -kehitykseen. Samaan aikaan kehitettiin myös muita malleja, kuten Yhdysvalloissa SM Ecosystem sekä Kiinassa Intelligent Manufacturing System Architecture (ISMA), jotka ovat varsin samanlaisia kuin RAMI 4.0. Se eroaa edeltäjästään siinä, että se pohjautuu palvelukeskeiseen arkkitehtuuriin, kun taas ISA-95 perustui perinteiseen hierarkkiseen malliin. RAMI 4.0 esitetään tavallisesti kolmiulotteisena mallina, jonka kolme akselia ovat hierarkia, tuotteen elinkaari ja kerrokset. Hierarkia-akseli muistuttaa vahvasti ISA-95-mallia, mutta RAMI tunnustaa myös, että käytännön syistä voidaan joskus ohittaa yksi tai useampi kerros. Elinkaariakselin tarkoitus on varmistaa, että kaikki tuotteen vaiheet otetaan huomioon suunnittelusta ja kehityksestä aina ylläpitoon ja käytöstä poistamiseen asti. (Salcher ym. 2024.)

Kerrosakseli on saanut vaikutteita tietokoneverkkomalleista, kuten kerroksittaisesta jäsentelystä sekä selkeistä syötteistä ja ulostuloista. Nämä helpottavat yksittäisten osien muokkaamista ilman, että koko järjestelmä täytyy muuttaa. RAMI 4.0 pystyy kolmiulotteisuutensa ansiosta kattamaan huomattavasti laajemmin älykkään valmistuksen osa-alueita kuin yksiulotteinen ISA-95-pyramidi. Molemmat vii-

tearkkitehtuurit korostavat hierarkiaa, mutta siinä missä ISA-95 sallii kommunikoinnin vain vierekkäisten tasojen välillä, RAMI 4.0 mahdollistaa viestinnän myös muiden kerrosten kesken. Tämä huomioi älylaitteiden lisääntyneet kyvykkyydet ja tukee joustavampaa tiedonkulkua. Samalla kuitenkin todetaan, että kerrosten ohittamiseen liittyy tietoturvariskejä. Esimerkiksi tuotantokoneen ei tulisi viestiä suoraan pilvessä toimivan ERP-järjestelmän kanssa, sillä se altistaisi tuotantokoneen internetille. (Salcher ym. 2024.)

3.1.4 Arkkitehtuurien haasteet

Vaikka ISA-95 ja RAMI 4.0 tarjoavat teollisuudelle vakiintuneita viitearkkitehtuureja, ne sisältävät myös merkittäviä haasteita. ISA-95-mallin hierarkkisuus rajoittaa joustavuutta, sillä se sallii viestinnän vain vierekkäisten tasojen välillä. Tämä johtaa usein siihen, että reaaliaikainen tiedonkulku on hidasta tai monimutkaista toteuttaa. RAMI 4.0 pyrkii ratkaisemaan ongelmaa mahdollistamalla kerrosten ohittamisen ja joustavamman viestinnän, mutta samalla se luo suuren määrän suoria yhteyksiä ja rajapintoja eri järjestelmien välille. Jokainen uusi toiminto edellyttää useiden erillisten rajapintojen kehittämistä ja ylläpitoa, mikä kasvattaa monimutkaisuutta. Tämä johtaa tilanteeseen, jossa joudutaan toteuttamaan lukuisia päällekkäisiä ja ylläpidettäviä integraatioita eri järjestelmien välille. On myös huomioitava, että samankaltaisia rajoitteita ja haasteita esiintyy myös muissa teollisuuden viitearkkitehtuureissa, joita on kehitetty rinnakkain RAMI 4.0:n kanssa. Näin ollen monimutkaisuus ja integraatioiden suuri määrä eivät ole vain yhden mallin ongelmia, vaan yleisempi piirre useissa teollisuuden viitearkkitehtuureissa. (Salcher ym. 2024.)

Runsas määrä suoria yhteyksiä ja integraatioita muodostaa hallitsemattoman integraatioverkoston (KUVIO 2). Se on vaikeasti hallittava ja altistaa järjestelmät virheille. Lisäksi Salcher ym. (2024) huomauttavat, että jos tuotantokone viestii suoraan pilvipohjaisen ERP-järjestelmän kanssa, syntyy merkittävä tietoturvariski. Tällainen rakenne ei myöskään skaalaudu hyvin, sillä jokainen uusi järjestelmä lisää eksponentiaalisesti integraatioiden määrää ja kasvattaa kokonaisuuden monimutkaisuutta. Tämä tekee arkkitehtuurista jäykän ja kalliin ylläpitää sekä alttiin virheille etenkin, kun tuotantoympäristöt digitalisoituvat nopeasti. Näiden haasteiden vuoksi tarvitaan uusi lähestymistapa, joka yksinkertaistaa tiedonkulkua ja vähentää integraatioiden määrää. Näihin haasteisiin vastaa seuraavassa luvussa esiteltävä Unified Namespace (UNS) -arkkitehtuuri (KUVIO 3).

3.2 Unified Namespace (UNS) -arkkitehtuuri

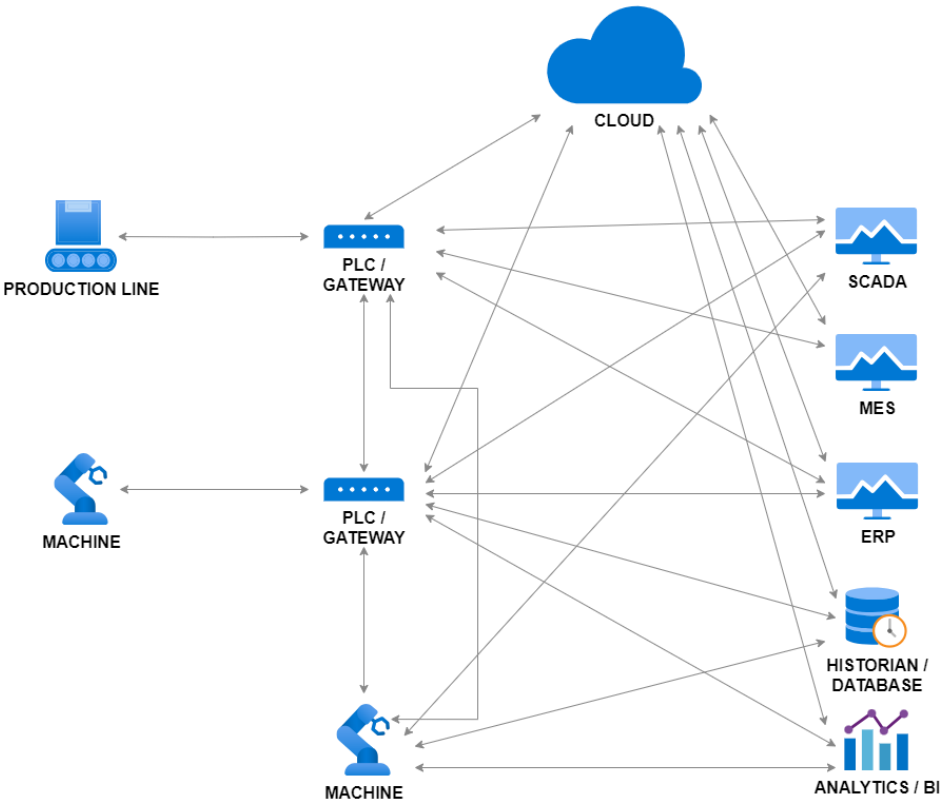
Viime aikoina Unified Namespace (UNS) -konsepti on saanut laajasti huomiota erityisesti käytännönläheisessä kirjallisuudessa, jossa sitä pidetään lupaavana ratkaisuna teollisuuden integraatiohaasteisiin. UNS voidaan määritellä tapahtumavetoiseksi arkkitehtuuriseksi lähestymistavaksi älykkääseen valmistukseen. Sen lähtökohtana on ajatus, että kaikki mitä tuotannon tasolla tapahtuu, voidaan tulkita tapahtumiksi. Tapahtumaksi määritellään havaittava muutos järjestelmässä tai resurssissa, joka tapahtuu tiettynä ajanhetkenä. Esimerkkejä tällaisista tapahtumista ovat koneen tilan muutos tai lämpötila-anturin mittaama lämpötilanvaihtelu. Tapahtuman tuottava resurssi julkaisee tiedon tapahtumavälittäjälle (event broker), joka toimii viestinvälityksen keskipisteenä, esimerkiksi MQTT välityspalvelimen muodossa. Jokainen järjestelmä, jota kyseinen tapahtuma kiinnostaa, voi tilata tapahtumavälittäjältä tiedon ja vastaanottaa sen reaaliajassa tapahtuman ilmaantuessa. (Salcher ym. 2024.)

UNS:n julkaisija–tilaajamalli poistaa tarpeen perinteisille integraatioille, jotka perustuvat suoriin yhteyksiin järjestelmien välillä. Tällaiset integraatiot ovat usein monimutkaisia ja kalliita toteuttaa, kun taas UNS yksinkertaistaa merkittävästi uusien järjestelmien liittämistä tai olemassa olevien laajentamista. (Salcher ym. 2024.) UNS-arkkitehtuurin tekniset vähimmäisvaatimukset voidaan tiivistää neljään periaatteeseen: sen tulee olla reunalähtöinen (edge-driven), eli tieto tuotetaan ja välitetään ensisijaisesti verkon reunalta; poikkeamia raportoiva (report by exception), eli dataa lähetetään vain muutosten tai poikkeamien yhteydessä; kevyt (lightweight), eli viestinvälitys toteutetaan kevyellä protokollalla, kuten MQTT:llä; sekä avoin arkkitehtuuri (open architecture), eli järjestelmä perustuu avoimiin rajapintoihin ja standardeihin. (Manditereza 2024.)

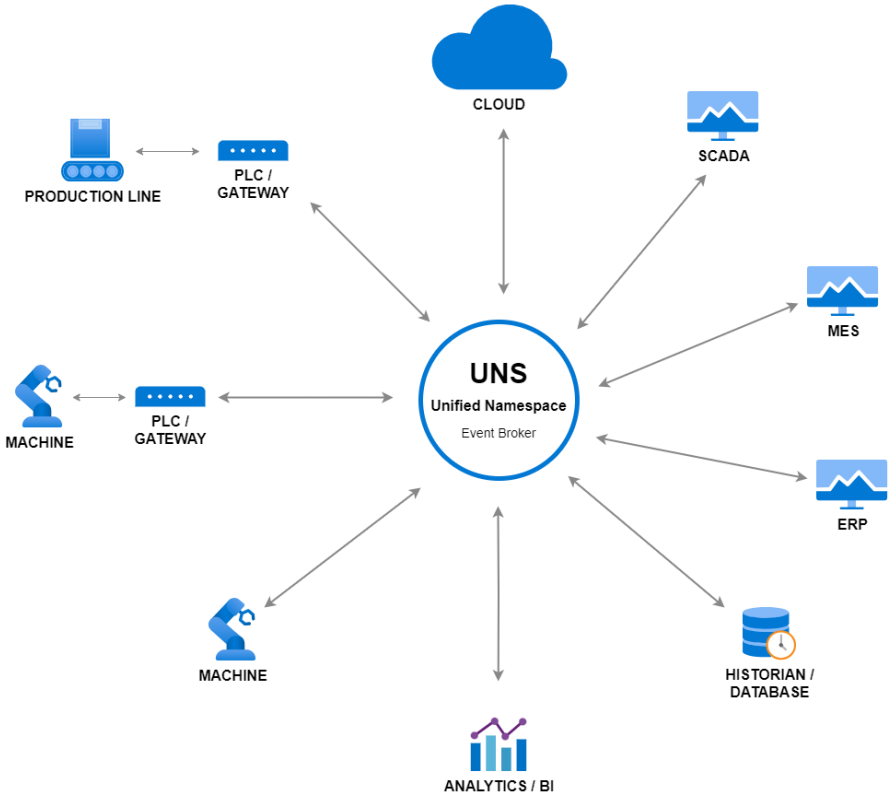
UNS määritellään teknologiariippumattomaksi viitekehykseksi, jossa tiedonvälitys perustuu tapahtumapohjaiseen lähestymistapaan. Sen ytimessä on nimeämisjärjestelmä, joka on saanut vaikutteensa ISA-95-viitearkkitehtuurista. Yhtenäinen nimeäminen varmistaa, että kaikki osapuolet löytävät tarvitsemansa tiedon tapahtumavälittäjältä. Käytännössä tämä tarkoittaa hierarkkista rakennetta, kuten *Enterprise/Site/Area/Line/Cell*, jossa tapahtumat voidaan tilata myös laajemmilla hakualueilla jokerimerkkejä hyödyntäen, esimerkiksi *Enterprise/Site/Area/Line/#.* Näin voidaan tilata kaikki tietyn tuotantolinjan tai solun tapahtumat ilman, että tarvitsee määritellä jokaista datapistettä erikseen. Hierarkkisen nimeämisen ensimmäinen taso, *Enterprise*, on tunniste koko yritykselle. Tapahtumatunnisteet pidetään yleensä mahdollisimman lyhyinä käyttämällä yrityksen nimen lyhennettä. Esimerkiksi Volkswagen voidaan esittää muodossa *VW* täydellisen nimen sijasta. Vastaavalla tavalla muut tasot, kuten *Site*, *Area*, *Line* ja *Cell*, määrittävät tarkemman sijainnin ja kontekstin tuotantoympäristössä.

(Salcher ym. 2024.)

Keskeisin ero UNS:n ja perinteisen integraatiomallin välillä on olennainen: kun perinteisessä integraatiossa käytetään suoria yhteyksiä järjestelmien välillä, joihin täytyy räätälöidä rajapinnat, tapahtumavetoisessa arkkitehtuurissa kaikki liittyvät yhteen välittäjän kautta, jolloin suoria yhteyksiä järjestelmien välillä ei tarvita. Tämä vähentää integraatioiden monimutkaisuutta ja estää hallitsemattoman integraatioverkoston muodostumisen (KUVIO 2), kun taas UNS-arkkitehtuuri kokoaa kaikki yhteydet yhden välittäjän kautta (KUVIO 3). (Salcher ym. 2024.) Yhteenvetona voidaan todeta, että UNS tarjoaa ratkaisuja niihin ongelmiin, joita aiemmissa arkkitehtuurimalleissa (ISA-95 ja RAMI 4.0) havaittiin. Tapahtumavetoinen julkaisija–tilaajamalli poistaa jäykät integraatiot, yhtenäinen hierarkkinen rakenne luo yhteisen kielen yrityksen eri tasoille ja teknologiariippumattomuus takaa, että UNS voidaan toteuttaa eri ympäristöissä joustavasti. Näin UNS muodostaa käytännönläheisen ja skaalautuvan lähestymistavan nykyaikaiseen teollisuuden datan hallintaan. Myöhemmin tämän työn konseptin toteutuksessa sovelletaan UNS-periaatteita työn rajauksen sallimissa rajoissa.



KUVIO 2. Point-to-point-integraatio (mukaillen Salcher ym. 2024)



KUVIO 3. Unified Namespace -arkkitehtuuri (mukaillen Salcher ym. 2024)

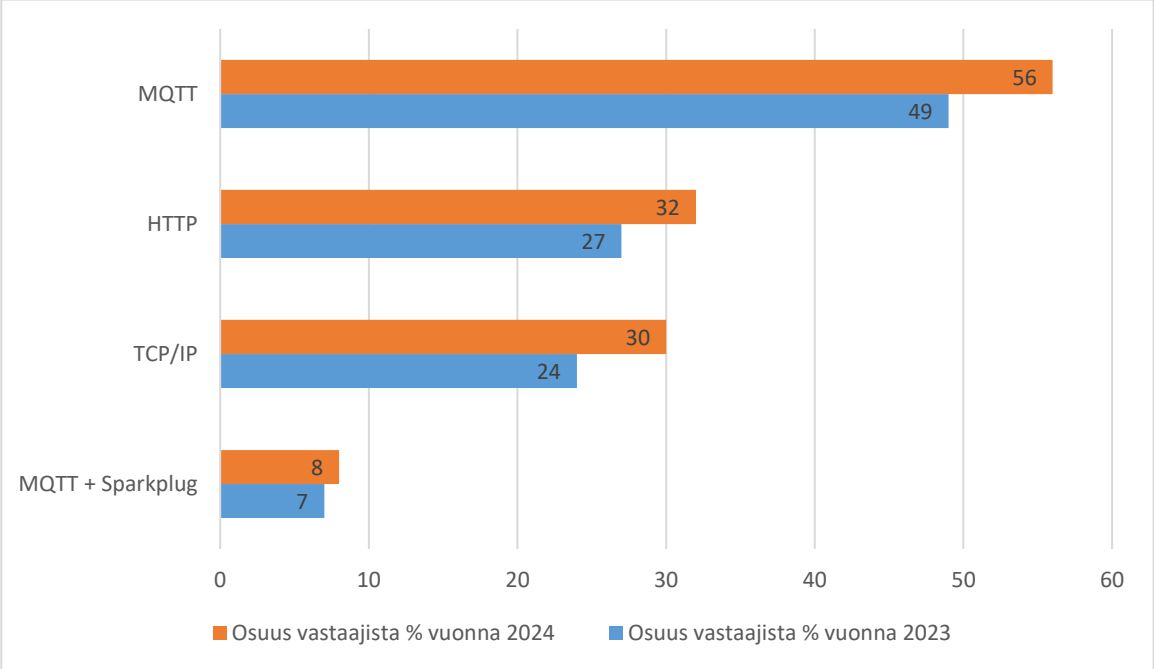
3.3 IIoT:ssa käytetyt protokollat

HTTP ja REST ovat yleiskäyttöisiä ja tehokkaita pilvipohjaisen mikropalveluarkkitehtuurin työvälineitä, mutta ne eivät yksin riitä IoT- ja IIoT-ympäristöihin, sillä laitteet toimivat usein rajoitetuissa olosuhteissa, joissa energiatehokkuus ja kaistanleveyden säästö ovat kriittisiä. Tämän vuoksi monet IoT- ja IIoT-ratkaisut hyödyntävät kevyitä ja tehokkaita protokollia, jotka tukevat kahta pääasiallista vuorovaikutusmallia: pyyntö–vastaus (request/response) ja julkaise–tilaa (publish/subscribe). Pyyntö–vastausmallissa asiakas muodostaa yhteyden palvelimeen, lähettää pyynnön ja saa vastauksen synkronisesti tai asynkronisesti, tyypillisenä esimerkkinä HTTP. Julkaise–tilaa-mallissa julkaisijat ja tilaajat toimivat toisistaan riippumatta, ja tiedonsiirto tapahtuu välityspalvelimen kautta. MQTT on yleisin julkaise–tilaa-mallin toteutus. (Desbiens 2023, 20.)

IIoT-sovelluksissa yleisesti käytettyjä protokollia ovat muun muassa HTTP, MQTT, CoAP, Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP) ja Modbus. Niiden valinta riippuu esimerkiksi verkon topologiasta, viivevaatimuksista ja sovellusalueen erityistarpeista. (Anitha, Manimurugan, Sridhar, Mathupriya & Charlyn Pushpa Latha 2022, 418.) IoT- ja IIoT-

järjestelmät voivat myös hyödyntää useampaa protokollaa samanaikaisesti, mikä parantaa yhteentoimivuutta ja mahdollistaa erilaisten viestinvälitysmallien tukemisen (Al-Masri, Kalyanam, Batts, Kim, Singh, Vo & Yan 2020, 94880). Protokollien välinen yhteentoimivuus on yhä tärkeämpää, sillä IIoT yhdistää hajautettuja ja heterogeenisia laitteita, joiden tulee kommunikoida luotettavasti sekä paikallisissa verkoissa että pilviympäristöissä (Anitha ym. 2022, 419–420).

Vuoden 2024 Eclipse IoT & Embedded Developer Survey -kyselyn mukaan MQTT on edelleen suosituin kommunikaatioprotokolla IIoT-ratkaisuissa, sitä ilmoitti käyttävänsä 56 % vastaajista. Seuraavaksi yleisimmät protokollat ovat HTTP (32 %) ja TCP/IP (30 %), joiden käyttöosuudet ovat melko tasaiset. Vähiten vastaajat ilmoittivat käyttäneensä MQTT + Sparkplug -laajennusta, jota käytti vain 8 % vastaajista. (KUVIO 4.) Tulokset osoittavat, että MQTT on vakiinnuttanut asemansa IIoT-ympäristöjen keskeisenä protokollana, kun taas Sparkplug B on vielä suhteellisen vähäisessä käytössä, mutta sen merkitys kasvaa standardoinnin (ISO/IEC 20237:2023; Harris 2023) myötä. HTTP ja TCP/IP puolestaan säilyttävät asemansa, koska ne ovat laajasti tunnettuja ja monikäyttöisiä, vaikka ne eivät ole yhtä optimoituja rajallisille laitteille ja viiveherkkiin sovelluksiin.



KUVIO 4. IIoT-sovellusten yleisimmät kommunikaatioprotokollat vastaajien mukaan (mukaillen Eclipse Foundation 2024, 16)

Koska protokollia on lukuisia, niiden kaikkien käsittely ei ole tämän työn rajauksen puitteissa mahdollista. Tässä työssä tarkastelu rajataan kolmeen IIoT:n kannalta keskeiseen protokollaan: OPC UA, MQTT ja Sparkplug B. Nämä on valittu, koska ne ovat teollisuudessa laajasti käytössä, niiden avoimet spesifikaatiot tukevat yhteentoimivuutta ja ne soveltuvat hyvin sekä paikallisverkkoihin että pilvi-integraatioon. Lisäksi rajaus on perusteltu myös toimeksiantajan näkökulmasta, sillä sen toimittamat järjestelmät hyödyntävät OPC UA -pohjaista tiedonsiirtoa, mikä tekee protokollan käsittelystä luontevan osan tätä työtä. Seuraavissa alaluvuissa tarkastellaan näitä protokollia yksityiskohtaisemmin, niiden vahvuuksia ja rajoituksia IIoT-sovellusten näkökulmasta.

3.3.1 OPC ja OPC UA

Open Platform Communications (OPC) on OPC Foundation -järjestön ylläpitämä standardi turvalliselle ja luotettavalle tiedonvaihdolle teollisuusautomaatiossa sekä muilla toimialoilla. Alkuperäinen, nykyään OPC Classicina tunnettu OLE for Process Control -standardi esiteltiin vuonna 1996. Se on sarja spesifikaatioita, jotka määrittelevät rajapinnan asiakkaiden ja palvelimien välillä sekä palvelimien kesken. Standardin kehittämiseen ovat osallistuneet teollisuuden toimittajat, loppukäyttäjät ja ohjelmistokehittäjät. Standardin tarkoituksena oli yhdistää PLC-kohtaiset protokollat, kuten Modbus ja Profibus yhdeksi yhtenäiseksi rajapinnaksi. Tämä mahdollistaa HMI- ja SCADA-järjestelmien toimimisen ”välikäden” avulla, joka muuntaa geneeriset OPC-luku- tai kirjoituspyynnöt laitekohtaisiksi pyynnöiksi ja päinvastoin. (What is OPC? 2025.) OPC Classic standardoi pääsyn prosessinohjausjärjestelmien reaaliaikaisiin ja historiallisiin tietoihin, mikä yksinkertaisti ohjelmistokehitystä ja mahdollisti yhteensopivuuden eri valmistajien järjestelmien välillä (Reddy, Maddireddy, Kumar, Prabhudesai, Reddy & Deo 2023).

OPC Classic saavutti laajan käyttöönoton useilla toimialoilla, kuten valmistavassa teollisuudessa, kiinteistöautomaatiossa, öljy- ja kaasuteollisuudessa sekä uusiutuvan energian ja energiayhtiöiden sovelluksissa. Myöhemmin palvelukeskeisten arkkitehtuurien käyttöönotto toi mukanaan uusia haasteita tietoturvaan ja tietomallinnukseen. (What is OPC? 2025.) Näiden rajoitteiden lisäksi OPC Classic oli voimakkaasti riippuvainen Windows-käyttöjärjestelmästä ja COM/DCOM-teknologiasta, mikä rajoitti sen alustariippumattomuutta ja skaalautuvuutta (Reddy ym. 2023). Näiden haasteiden ratkaisemiseksi kehitettiin OPC Unified Architecture (UA) -spesifikaatiot, jotka esiteltiin vuonna 2008 (Reddy ym. 2023; Unified Architecture 2025). OPC UA on myös kansainvälinen standardi, joka on määritelty IEC 62541 -standardisarjassa (IEC TR 62541-1:2020).

OPC UA on alustariippumaton viestintästandardi, joka mahdollistaa erilaisten järjestelmien ja laitteiden välisen tiedon- ja komentovaihdon kaikilla teollisuuden aloilla. Standardi soveltuu muun muassa teollisiin antureihin ja toimilaitteisiin, ohjausjärjestelmiin, MES-tuotannonohjausjärjestelmiin ja ERP-toiminnanohjausjärjestelmiin, sekä IIoT-ympäristöihin ja koneiden väliseen (Machine to Machine, M2M) viestintään. OPC UA määrittelee yhteisen infrastruktuurimallin tiedonsiirrolle. Määrittely koostuu neljästä pääelementistä: tietomallista, joka kuvaa tiedon rakenteen, toiminnan ja semantiikan; viestimallista, joka kuvaa sovellusten välisen vuorovaikutuksen; viestintämallista, joka määrittää tiedonsiirron päätepisteiden välillä; sekä yhteensopivuusmallista, joka varmistaa eri järjestelmien yhteensopivuuden. (OPC Foundation 2024, 7.)

OPC UA tukee kahta pääarkkitehtuurimallia: asiakas–palvelin- (Client–Server) ja julkaisija–tilaaja-mallia (Publish–Subscribe). Asiakas–palvelinmallissa asiakkaat ja palvelimet toimivat vuorovaikutteisina kumppaneina, ja sovellus voi sisältää molemmat roolit. Ne kommunikoivat palvelurajapinnan välityksellä pyyntö–vastausmenetelmillä tai tilausmekanismin avulla, mikä mahdollistaa ilmoitusten, kuten tietomuutosten tai tapahtumien toimittamisen. Palvelimen osoiteavaruus (AddressSpace) sisältää solmut (Node), jotka kuvaavat järjestelmän objektit, ominaisuudet ja suhteet, ja se tukee informaatiomalleja sekä erilaisia näkymiä. Julkaisija–tilaajamallissa julkaisijat ja tilaajat eivät ole suoraan yhteydessä, vaan viestinvälitysjärjestelmä hoitaa viestien välityksen. Lisäksi se tukee myös muita viestinvälitysjärjestelmiä. Toteutus voi olla välittäjätön, jossa viestintä tapahtuu verkon infrastruktuurin kautta esimerkiksi UDP-multicastilla, tai välittäjäpohjainen, missä viestit kulkevat välityspalvelimen kautta esimerkiksi MQTT- tai AMQP-protokollaa hyödyntäen. Mallien synergia mahdollistaa sen, että sama sovellus voi hyödyntää molempia malleja rinnakkain ja julkaisija–tilaaja konfigurointi voidaan tehdä asiakas–palvelinmallin kautta. (OPC Foundation 2024, 11–17.)

OPC UA:n tietoturvamalli käsittelee sekä asiakkaiden ja palvelinten todennusta että käyttäjien tunnistamista. Siihen sisältyy myös viestinnän eheyden ja luottamuksellisuuden varmistaminen sekä toiminnallisuuden todennettavuus. Malli on suunniteltu joustavaksi, jotta sen turvamekanismit ja -parametrit voidaan valita ja säätää käyttöympäristön tarpeiden mukaan. Turvallinen viestintä perustuu yleensä suojattuun viestintäkanavaan (SecureChannel), joka neuvotellaan istunnon (Session) alussa ja jonka avulla suojataan kaikki viestit istunnon aikana. Istunnon ja viestintäkanavan perustamiseen käytetään usein X.509-varmenteita sovellusten tunnistamisessa. Tämän lisäksi palvelin vahvistaa käyttäjän identiteetin ja valtuutuksen ennen resurssien käyttöä. Malli sisältää myös vähimmäisturvaprofiilit, jotka kaikkien OPC UA -sovellusten on tuettava. Lisäksi se tukee auditointia siten, että tapahtumat ja viestit voidaan tarvittaessa jäljittää asiakas- ja palvelinlokeissa. (OPC Foundation 2024, 9–10.)

OPC UA on keskeinen viestintästandardi IIoT:ssa, koska se yhdistää laitteiden, järjestelmien ja sovellusten välisen tiedonsiirron yhtenäisen tietomallin ja tietoturvan puitteissa. Sen arkkitehtuurimallit, asiakas–palvelin ja julkaisija–tilaaja, mahdollistavat joustavat integraatiot sekä perinteisten automaatiojärjestelmien että pilvi- ja reunaratkaisujen välillä. OPC UA soveltuu erityisen hyvin tehtaan tasolle, koska sen joustava tietomalli, vahva tietoturva ja laaja protokollatuki helpottavat eri järjestelmien ja laitteiden yhdistämistä. Kyky tukea sekä reaaliaikaista että historiallista dataa sekä monipuolisia tietomalleja tekee siitä erinomaisen perustan IIoT-sovelluksille ja UNS-arkkitehtuurille, joita hyödynnetään tämän työn toteutusosuudessa. (OPC Foundation 2024, 7–8, 16.)

3.3.2 MQTT

Message Queuing Telemetry Transport (MQTT) on OASIS Open -järjestön standardoima ja kansainväliseksi standardiksi hyväksytty kevytrakenteinen viestintäprotokolla (ISO/IEC 20922). Protokollan kehittivät vuonna 1999 IBM:llä työskennellyt Andy Stanford-Clark ja Arcomilla työskennellyt Arlen Nipper tarpeeseen valvoa öljyputkia satelliittiyhteyden kautta. Tuon ajan olemassa olevat viestintäprotokollat perustuivat jatkuvaan tiedon kyselyyn (polling), mikä kulutti liikaa kaistanleveyttä ja lyhensi kentällä olevien laitteiden akun kestoa. Ratkaisuksi kehitettiin uusi protokolla, jonka tavoitteena oli mahdollisimman pieni virrankulutus ja tehokas kaistan käyttö. MQTT perustuu julkaise–tilaa (Publish–Subscribe) -malliin, jossa viestien julkaisijat ja tilaajat ovat toisistaan täysin riippumattomia. Kaikki viestiliikenne kulkee yhden tai useamman välityspalvelimen (Broker) kautta, jotka huolehtivat viestien edelleen toimittamisesta oikeille tilaajille. Asiakas voi toimia sekä julkaisijana että tilaajana, ja useat asiakkaat voivat olla samanaikaisesti kummassakin roolissa saman välityspalvelimen kautta. (Desbiens 2023, 68.)

MQTT-viesti koostuu kolmesta osasta: kiinteästä otsakkeesta (Fixed Header), vaihtuvasta otsakkeesta (Variable Header) ja hyötykuormasta (Payload). Kiinteä otsake sisältää muun muassa tietoa paketin tyypistä esimerkiksi PUBLISH, SUBSCRIBE tai CONNECT. Vaihtuva otsake sisältää paketin ominaisuudet, kuten aiheen nimi (Topic Name), Quality of Service (QoS) -viestinvälityksen laatutaso, viestin säilytyslippu (Retain Flag) sekä yksilöivä tunniste (Packet ID), jota käytetään viestin toimituksen varmistamiseen QoS-tasoilla 1 ja 2. MQTT-protokollassa välityspalvelin välittää viestit tilaajille aiheen nimen perusteella. Jokainen viesti liitetään aiheeseen, joka on UTF-8-muotoinen merkkijono, esimerkiksi *Helsinki/1/101/temperature*. Aiheet muodostavat hierarkian, jossa tasot erotetaan vinoviivalla (/).

Tilaajat eivät tilaa yksittäisiä viestejä, vaan käyttävät aiheissa suodattimia. Suodattimet voivat sisältää jokerimerkkejä, kuten plusmerkin (+), joka vastaa mitä tahansa yksittäistä hierarkian tasoa tai risuaitamerkin (#), joka kattaa kaikki jäljellä olevat hierarkian tasot. Esimerkiksi tilaus suodattimella *Helsinki/1/+/temperature* vastaanottaa kaikki ensimmäisen kerroksen huonelämpötilat, mutta ei muiden kerrosten mittauksia. (Desbiens 2023, 69–71.)

QoS-tasot määrittävät, kuinka luotettavasti viestit toimitetaan. MQTT tukee kolmea tasoa, QoS 0–2. Tasolla 0 viesti lähetetään korkeintaan kerran ilman kuittausta, jossa viestien häviäminen on mahdollista. Tasolla 1 viesti lähetetään vähintään kerran ja lähettäjä säilyttää viestin, kunnes vastaanottaja kuittaa sen PUBACK-paketilla. Tämä lisää varmuutta viestinvälitykseen, mutta viestien duplikaatit ovat mahdollisia. Tasolla 2 viesti välitetään vain kerran ja käytetään nelivaiheista viestinvaihtoa (PUBLISH, PUBREC, PUBREL, PUBCOMP), joka varmistaa viestin toimituksen vain kerran. Tämä lisää toimitusvarmuutta, mutta ei takaa viestin perillemenon viivettä, joka voi vaihdella minuuteista tunteihin riippuen verkon ja välityspalvelimen kuormituksesta. QoS-taso tulee valita sovelluksen vaatimusten mukaan. Jokaisella tasolla on etunsa ja rajoitteensa: tasot 1–2 parantavat toimitusvarmuutta, mutta voivat lisätä viestien viiveitä ja synnyttää duplikaatteja, jotka kuormittavat järjestelmää. Tasolla 0 viestejä saattaa kadota, mutta se soveltuu hyvin sovelluksiin, joissa yksittäisten viestien perille saapuminen ei ole kriittistä. (Desbiens 2023, 73–74.)

MQTT-protokollassa tilaajat saavat oletusarvoisesti vain ne viestit, jotka julkaistaan sen jälkeen, kun tilaussuhde on muodostettu. Joskus on kuitenkin tarpeen toimittaa myös aiemmin julkaistu viesti uusille tilaajille erityisesti, jos dataliikenne on harvempaa. Tätä varten käytetään säilytettyjä viestejä (retained messages), joiden avulla uusi tilaaja saa liittyessään heti viimeisimmän viestin aiheesta. Tällöin välityspalvelin tallentaa viestin ja sen QoS-tason kyseiselle aiheelle. Kun uusi tilaaja liittyy aiheeseen, se saa tallennetun viestin välittömästi. Säilytetyt viestit eivät ole sidottuja pysyviin sessioihin, vaan ne toimitetaan tilaajille sessiotyypistä riippumatta. (Desbiens 2023, 78–79.) Lisäksi MQTT-protokollassa hyötykuorma sisältää viestin varsinaisen sisällön, jonka muoto ja merkitys ovat sovelluskohtaisia. Viestin tulkinta jää kokonaan vastaanottavan sovelluksen vastuulle. (OASIS Open 2019.) Spesifikaatio määrittelee viestin teoreettiseksi enimmäiskooksi noin 268 MB, mutta näin suuret viestit eivät ole kuitenkaan käytännöllisiä eivätkä hyvin skaalautuvia (Desbiens 2023, 74).

MQTT-protokolla sisältää muutamia tietoturvaominaisuuksia, mutta ne eivät ole käytössä oletusarvoisesti vaan jäävät kehittäjän vastuulle. Viestiliikenne ei ole oletusarvoisesti salattua, joten suosituksena

on käyttää MQTT-yhteyksiä TLS-salauksen avulla. Vaikka TLS tuo mukanaan hieman viiveitä ja resurssien kuormitusta, se on useimmissa järjestelmissä välttämätön. Erityisen rajoittuneissa laitteissa voidaan turvautua esimerkiksi hyötykuorman salaamiseen tai CONNECT-viestien salasanojen suojaamiseen erikseen. MQTT 5.0 -versio tuo mukanaan AUTH-paketin, joka mahdollistaa haaste–vastaustyyppisen autentikoinnin ja helpottaa integraatiota esimerkiksi Lightweight Directory Access Protocol (LDAP)- ja Open Authorization (OAuth)-pohjaisiin järjestelmiin. Lisäksi on suositeltavaa käyttää tuotantoympäristössä luotettavan varmentajan (CA) allekirjoittamia TLS-sertifikaatteja sekä rajoittaa viestien kokoa ja asiakaskohtaisia yhteyksiä mahdollisten hyökkäysten varalta. (Desbiens 2023, 87–88.)

3.3.3 Sparkplug B

Sparkplug on alun perin Cirrus Link Solutionsin kehittämä ja nykyisin Eclipse Foundationin ylläpitämä avoin spesifikaatio MQTT:n päälle rakennettavalle laajennukselle (Desbiens 2023, 103–104). Sen tarkoituksena on standardoida aiheavaruus, tilanhallinta ja hyötykuormat, jotta laitteiden ja järjestelmien välinen viestintä olisi yhtenäistä ja yhteensopivaa (Eclipse Sparkplug Contributors 2022, 2–4). Sparkplugin kehitti Arlen Nipper ja hänen tiiminsä Cirrus Links Solutionsilta, ja ensimmäinen versio julkaistiin 2016 (Desbiens 2023, 104). Vuonna 2023 Sparkplug® 3.0 hyväksyttiin kansainväliseksi ISO/IEC 20237:2023 standardiksi (ISO/IEC 20237:2023; Harris 2023).

MQTT-spesifikaatio ei määrittele aiheavaruuden (Topic Namespace) rakennetta, eikä julkaistavan tai tilattavan datan hyötykuorman esitystapaa. Se ei myöskään määrittele tilatiedon toteutusta ja hallintaa. Nämä määritykset puuttuivat alkuperäisestä spesifikaatiosta tarkoituksella, jotta voitaisiin tarjota mahdollisimman joustava ratkaisu. Tämä joustavuus on joissakin tilanteissa merkittävä etu, mutta toisissa se voi muodostua puutteeksi, mikäli tarvitaan tarkkaa yhteentoimivuuden määrittelyä. Tähän tarpeeseen on kehitetty Sparkplug-spesifikaatio. (Eclipse Sparkplug Contributors 2022, 3–4.)

Sparkplug-spesifikaatio määrittelee kolme keskeistä tavoitetta, joilla pyritään parantamaan MQTT:n yhteentoimivuutta ja soveltuvuutta reaaliaikaisiin SCADA- ja IIoT-ympäristöihin. Ensimmäinen tavoite on määritellä MQTT:n aiheavaruus. MQTT:n vahvuutena on vapaasti toteutettava aiheavaruuden rakenne, mikä on mahdollistanut sen laajan käytön eri IoT-ratkaisuissa. Sparkplug-spesifikaatio kuitenkin tunnistaa tarpeen yhtenäiselle ja selkeästi dokumentoidulle aiheavaruudelle, joka on optimoitu erityisesti SCADA/IIoT-sektorille. Aiheavaruus on rakennettu siten, että se mahdollistaa automaattisen

laitteiden ja sovellusten löytämisen sekä kaksisuuntaisen viestinnän järjestelmän MQTT-asiakkaiden välillä. (Eclipse Sparkplug Contributors 2022, 2.)

Toinen tavoite on määritellä MQTT:n tilanhallinta (State Management). Sparkplug hyödyntää MQTT:n alkuperäistä ominaisuutta, jatkuvaa istuntotietoisuutta (Continuous Session Awareness). Se mahdollistaa reaaliaikaisen tilatiedon välittämisen myös kaistanleveydeltään rajoittuneissa ja katkoksille alttiissa verkoissa, kuten matkapuhelin-, satelliitti- tai radiopohjaisissa yhteyksissä. Tämä toimintatapa ei ainoastaan paranna järjestelmän luotettavuutta, vaan myös optimoi kaistanleveyden käyttöä, mikä voi alentaa verkon käyttökustannuksia. Kolmas tavoite on määritellä MQTT:n hyötykuorma. Sparkplug pyrkii säilyttämään alkuperäisen MQTT:n keveyden, kaistanleveyden tehokkuuden ja alhaisen viiveen, mutta lisää samalla moderneja koodausmenetelmiä, jotka soveltuvat SCADA- ja IIoT-sovelluksiin. Hyötykuormien koodausmenetelmät on rakennettu niin, että ne ovat sekä tehokkaita että yhteensopivia laajassa teollisessa käytössä. (Eclipse Sparkplug Contributors 2022, 2–3.)

Sparkplug-spesifikaatiossa aiheavaruus on rakennettu niin, että sen avulla voidaan päätellä käytetty hyötykuorman koodausmenetelmä. Historiallisesti Sparkplugissa on ollut kaksi koodausmenetelmää: Sparkplug A ja Sparkplug B. Molemmat käyttävät tunnistetta, joka kertoo koodausmenetelmän ja sen version. Käytössä olevat tunnisteet ovat spAv1.0 ja spBv1.0, ja ne koostuvat kolmesta osasta: Sparkplug-tunniste (sp), koodausmenetelmä (A tai B) sekä versionumero, joka tällä hetkellä v1.0. Sparkplug A perustui Eclipse Kura™ -avoimen lähdekoodin projektin Google Protocol Buffer -määrittelyyn. Pian A-version julkaisun jälkeen kehitettiin Sparkplug B, joka korjasi useita A-version ongelmia. Vähäisen käyttöönoton ja B-version nopean julkaisun vuoksi A-versio poistettiin spesifikaatiosta, eikä sitä enää tueta. (Eclipse Sparkplug Contributors 2022, 3.)

Sparkplug B kehitettiin rikkaamman tietomallin pohjalta järjestelmäintegraattoreiden ja MQTT:tä käyttävien loppuasiakkaiden palautteen perusteella. Siihen lisättiin muun muassa metrikoiden aikaleimat, monimutkaisten tietotyyppien tuki, metatiedot ja muita parannuksia. Sparkplug B soveltuu erinomaisesti IIoT-ympäristöihin, joissa tarvitaan kevyttä ja reaaliaikaista, mutta standardoitua viestintäratkaisua, sillä se määrittelee yhtenäisen aiheavaruuden, hyötykuorman ja tilanhallinnan mallin reaaliaikaisia SCADA- ja IIoT-sovelluksia varten. (Eclipse Sparkplug Contributors 2022, 3–4.)

Sparkplug-arkkitehtuuri koostuu tyypillisesti kolmesta pääkomponentista: verkon reunalla olevista solmuista (Edge of Network), palvelinsovelluksista ja MQTT-välityspalvelimesta. Reunasolmut liittävät kenttälaitteet ja anturit MQTT-verkkoon, palvelinsovellukset toimivat SCADA- tai MES-järjestelminä,

ja MQTT-välityspalvelin välittää viestit näiden välillä. Sparkplugin määrittelemä standardoitu aiheavaruus ja hyötykuorma mahdollistavat eri valmistajien laitteiden ja sovellusten yhteentoimivuuden ilman räätälöityä sovitusta. Perinteiseen MQTT:hen verrattuna Sparkplug-spesifikaatio tarjoaa valmiit mekanismit automaattiseen laitteiden tunnistamiseen, tilatietojen hallintaan ja yhdenmukaiseen datan esittämiseen. (Eclipse Sparkplug Contributors 2022, 2–3.) Lisäksi Sparkplugin avoin spesifikaatio ja referenssitoteutus, Eclipse Tahu, tukevat laajaa ekosysteemiä, johon kuuluu myös kaupallisia ja avoimen lähdekoodin toteutuksia (Desbiens 2023, 104, 117–119).

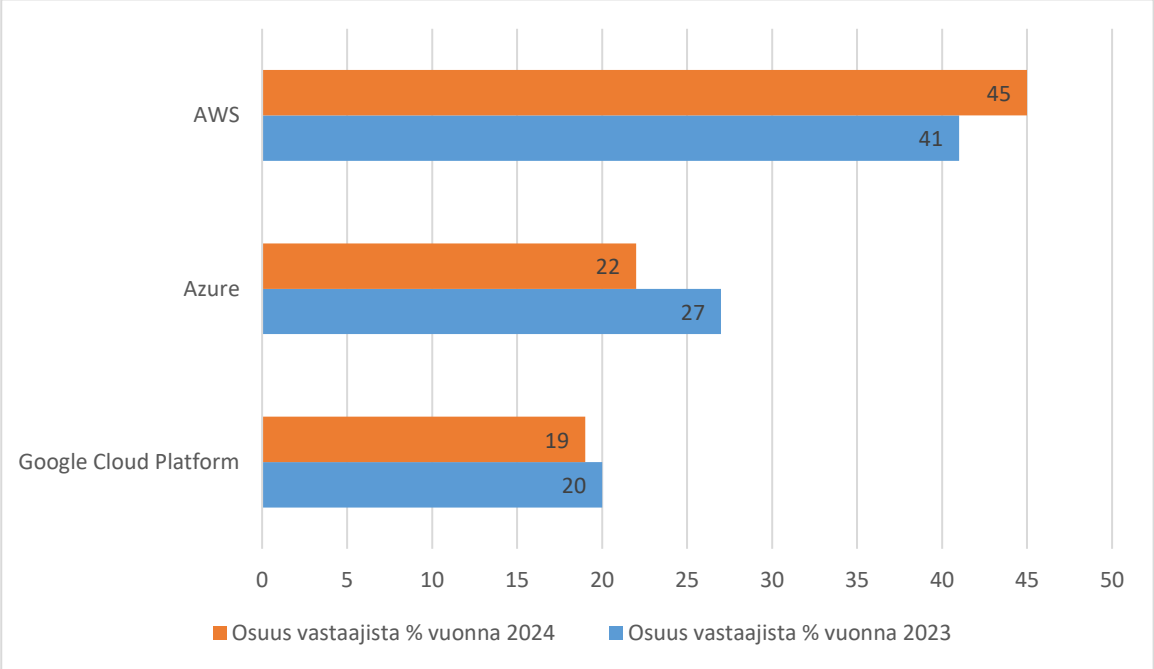
3.4 Pilvipalveluiden rooli IIoT-arkkitehtuurissa

Yksi keskeisimmistä teollisuus 4.0:n tuomista eduista on IoT- ja IIoT-antureilla kerätty reaaliaikainen data tuotantoympäristön kaikista resursseista, kuten laitteista ja prosesseista. Tämä jatkuvasti kerättävä ja välitettävä data tarjoaa arvokasta tietoa tuotannon suorituskyvystä ja sitä voidaan hyödyntää esimerkiksi tuotannon ohjauksessa, varastonhallinnassa ja ennusteiden laadinnassa. Datan avulla organisaatiot pystyvät reagoimaan nopeammin muutoksiin, tunnistamaan pullonkauloja sekä optimoimaan resurssien käyttöä. Edistyneet teknologiat, kuten tekoäly, IoT ja digitaaliset kaksoset, edellyttävät kuitenkin huomattavia laskentaresursseja ja tallennuskapasiteettia, minkä vuoksi pelkkien sisäisten ratkaisujen rakentaminen ei ole kustannustehokasta. (Attaran ym. 2024, 4.)

Pilvilaskenta tarjoaa valmistajille skaalautuvan ympäristön, jossa data voidaan tallentaa ja käsitellä tehokkaasti. Se muodostaa älykkään valmistuksen peruskiven, koska se mahdollistaa tiedon tallentamisen, prosessoinnin ja analysoinnin sekä tukee digitaalisia kaksosia. Pilvilaskennan avulla suuret datamäärät voidaan käsitellä joustavasti ja niitä voidaan hyödyntää virtuaalimallien ja fyysisten prosessien samanaikaiseen tarkasteluun. Tämä nopeuttaa tuotekehitystä ja valmistusprosesseja sekä mahdollistaa samanaikaisen yhteistyön organisaation eri tasojen ja sijaintien välillä. Lisäksi pilviteknologiat tarjoavat joustavuutta, turvallisuutta ja kustannussäästöjä verrattuna perinteisiin ratkaisuihin. (Attaran ym. 2024, 4.)

Pilvipalveluiden roolia voidaan tarkastella myös käytännön markkinadatan pohjalta. Eclipse Foundation toteuttaa vuosittain IoT & Embedded Developer Survey Report -kyselyn, ja vuoden 2024 tulosten mukaan AWS on edelleen vahvimmin käytetty pilvialusta IoT-ratkaisuissa: sitä ilmoitti käyttävänsä 45 % vastaajista. Seuraavaksi yleisimmät pilvialustat ovat Azure (22 %) ja GCP (19 %), joiden käyttö-

osuudet ovat lähes yhtä suuret. (KUVIO 5.) Kyselyn toteutti Eclipse Foundation vuonna 2024, ja siihen osallistui 747 vastaajaa eri puolilta maailmaa. Vastaajat edustivat kehittäjiä, arkkitehteja ja päätöksentekijöitä useilta toimialoilta, mikä tekee tuloksista kansainvälisesti kattavat. Suurin osa vastaajista (75 %) ilmoitti käyttävänsä avoimen lähdekoodin ohjelmistoja, ja neljännes osallistuu aktiivisesti niiden kehittämiseen (Eclipse Foundation 2024, 32.)



KUVIO 5. Käytetyimmät pilvialustan tarjoajat IoT-ratkaisuissa (mukaillen Eclipse Foundation 2024, 19)

Markkinaosuuksien lisäksi eri alustat eroavat ominaisuuksiltaan ja vahvuuksiltaan. Pilvipalvelut muodostavat perustan IoT:n tuottaman datan tallennukselle, käsittelylle ja analytiikalle, kun taas itse data syntyy IoT-laitteista. Näiden kahden integraatio mahdollistaa kehittyneempien ja älykkäämpien IoT-sovellusten toteuttamisen. Kolme keskeisintä IoT-pilvipalvelun tarjoajaa ovat AWS, Azure ja GCP, joista jokaisella on omat vahvuutensa. AWS:n vahvuuksia ovat kattava valikoima IoT-palveluita, globaali skaalautuvuus ja saumaton integraatio muihin AWS-palveluihin. Microsoft Azuren etuina on yhteensopivuus Microsoftin ekosysteemiin sekä kehittyneet analytiikka- ja tekoälyominaisuudet. GCP erottuu erityisesti datankäsittelyn, koneoppimisen ja analytiikan työkaluillaan hyödyntäen laajaa datakeskusverkostoaan. iEECON 2024 -konferenssissa julkaistun tutkimuksen mukaan pilvipalvelualustan valintaperusteet riippuvat sovelluksen erityisvaatimuksista: GCP tarjoaa parhaimmillaan nopeimmat vasteajat ja laskentatehon, AWS erottuu kustannustehokkuudellaan ja laajalla valikoimalla palveluita, ja Azure sijoittuu näiden väliin.

(Sithiyopasakul, Archevapanich, Sithiyopasakul, Lasakul, Purahong & Benjangkaprasert 2024.)

Hyötyjen ohella pilvipalveluiden käyttöön IIoT-ympäristöissä liittyy myös haasteita, jotka on huomioitava arkkitehtuurin suunnittelussa. Tietoturva on yksi merkittävimmistä huolenaiheista, sillä IoT-laitteista kerätty data siirtyy ja tallentuu pilveen usein ilman, että käyttäjät tietävät sen tarkkaa fyysistä sijaintia. Tämä heikentää luottamusta palveluihin ja korostaa vahvojen salaustekniikoiden tarvetta. Suorituskyky muodostaa toisen haasteen: IoT-sovellukset edellyttävät korkeaa suorituskykyä, jota ei kaikissa käyttöympäristöissä ole mahdollista saavuttaa. Luotettavuus on kriittinen erityisesti ajoneuvoissa, kirurgisissa instrumenteissa ja turvallisuusjärjestelmissä, joissa palvelukatkos voi vaarantaa toiminnan välittömästi. Lisäksi valtavien datamäärien hallinta on merkittävä ongelma: vuoteen 2025 mennessä jopa 50 miljardin IoT-laitteen arvioitiin tuottavan dataa, mikä asettaa suuria paineita pilvipalveluiden tallennus- ja kyselysuorituskyvylle. Samalla myös ylläpidon ja suorituskyvyn optimointi ovat keskeisiä, sillä pilviympäristön on kyettävä hallitsemaan kasvava määrä laitteita sekä varmistamaan tietoturva ja palvelun tehokkuus. (Abdulkareem, Zeebaree, Sadeeq, Ahmed, Sami & Zebari 2021, 2–3.)

4 KONSEPTIN TOTEUTUS

Tässä luvussa kuvataan opinnäytetyönäni kehittämäni teollisen IoT-järjestelmän konseptin toteutus kokonaisuutena. Toteutuksen tavoitteena oli soveltaa konseptitasolla UNS-arkkitehtuurin periaatteita käytännön ympäristössä ja osoittaa, miten avoimiin standardeihin perustuva hierarkkinen tietomalli voidaan yhdistää AWS-pilvipalveluiden tarjoamiin ratkaisuihin. UNS toimi arkkitehtuurin viitekehyksenä ja rakenteellisena mallina, jonka perusteella tein valinnat laitteesta, protokollista, viestintäratkaisuista ja pilvipalveluista. Luvussa esitellään toteutetun järjestelmän kokonaisarkkitehtuuri, laitteistototeutus sekä SIMATIC IOT2050 -yhdyskäytävälaitteessa toimiva Java-pohjainen reunasovellus, joka toteuttaa resurssitehokasta datan esikäsittelyä. Lisäksi tarkastellaan käytettyjä kehitysympäristöjä, ohjelmistokirjastoja ja AWS-pilvipalveluita, jotka mahdollistivat reuna–pilvi-integraation sekä datan tallennuksen ja analytiikan pilviympäristössä. Nämä kokonaisuudet muodostivat teknisen perustan UNS-arkkitehtuurin soveltamiselle ja reuna–pilvi-integraatiolle.

4.1 Laitteistoteutus ja viestintäratkaisut

Valitsin konseptin reuna–pilvi-integraation mahdollistajaksi Siemensin SIMATIC IOT2050 Advanced-mallin yhdyskäytävälaitteen, koska se on suunniteltu teolliseen käyttöön ja erityisesti reunalaskentaa varten (SIMATIC IOT2050 2025). Laite tukee sekä reunalaskennan (Singh ym. 2025) että UNS-arkkitehtuurin periaatteita (Manditereza 2024; Salcher ym. 2024), joissa data esikäsitellään reunalla lähellä datan syntypaikkaa ja viestinnässä käytetään kevyitä, avoimiin rajapintoihin sekä standardeihin perustuvia ratkaisuja. Se perustuu Texas Instrumentsin Sitara AM6548 -suorittimeen, jossa on kaksiytiminen ARM Cortex-A53-mikroprosessorialusta. Lisäksi laitteessa on 2 GB RAM-muistia, 16 GB eMMC-tallennustilaa sekä Linux Debian -käyttöjärjestelmä. Rajapintoina se tarjoaa teollisuuteen tarvittavat yhteydet, kuten Gigabit Ethernetin, RS232/RS485-sarjaliitännät, USB:n ja miniPCIe-liitännän. (SIMATIC IOT2050: Operating Instructions 2024.)

Laitteen keskeisiä vahvuuksia ovat avoimeen käyttöjärjestelmään perustuva joustavuus, kustannustehokkuus ja riittävä laskentateho pienimuotoisiin reunalaskentaa hyödyntäviin IIoT-sovelluksiin. Monipuoliset liitäntämahdollisuudet tukevat integraatiota erilaisiin automaatiojärjestelmiin ja alustariippumattomasti eri pilvipalvelualustoihin. Lisäksi laitteelle voidaan asentaa ja suorittaa erilaisia sovelluk-

sia, kuten kevyt MQTT-välityspalvelin. Nämä ominaisuudet tukivat konseptin arkkitehtuuriperiaatteita, joissa data käsiteltiin reunalla ennen pilveen siirtoa. Nämä seikat ja toimeksiantajan laaja kokemus Siemensin teollisuustuotteista muodostivat yhdessä perustan laitteen valinnalle tähän konseptiin.

Konseptin testaamisessa hyödynsin Siemensin SIMATIC ET200SP -sarjan PLC:tä, joka perustuu Siemensin S7-1500-logiikkaohjainsarjaan ja on suunniteltu hajautettujen ohjausratkaisujen tarpeisiin (Distributed Controllers 2025). Ohjelmoin PLC:lle simuloidun pumppausjärjestelmän, joka tuotti prosessidataa konseptin testausta varten. Simuloitu ohjelma sisälsi pumppausjärjestelmän keskeisiä komponentteja, kuten taajuusmuuttajilla ohjattuja pumppuja, venttiileitä sekä virtaus-, lämpötila-, paine- ja tärinämittareita ja energiamittarin. Se tuotti yhteensä noin 30 datapistettä, jotka muodostivat jatkuvan datavirran konseptin testaamiseen. PLC toimi OPC UA -palvelimena ja välitti prosessidatan kyberturvallisesti yhdyskäytävälaitteelle OPC UA -rajapinnan välityksellä. Yhteys suojattiin vahvalla autentikoinnilla, jossa yhdistettiin sertifikaattipohjainen varmenne sekä käyttäjätunnukseen perustuva kirjautuminen.

Yhdyskäytävälaite toimi OPC UA -asiakkaana. Asensin laitteeseen Siemensin Getting Started Guide for AWS IoT Greengrass (2024) -oppaan ohjeiden mukaisesti Siemensin tarjoaman Debian 11 -pohjaisen IOT2050 example imagen, joka sisälsi valmiiksi konfiguroidut ajurit ja perusohjelmistot (SIMATIC IOT2050: Getting Started Guide for AWS IoT Greengrass 2024). Tämä mahdollisti nopean käyttöönoton ja omien sovellusten integroinnin. Lisäksi asensin laitteelle AWS IoT Greengrass -ympäristön Siemensin oppaan ja AWS IoT Greengrass Developer Guide -dokumentaation (Tutorial: Getting started with AWS IoT Greengrass V2 2025) mukaisesti. Sen avulla voitiin suorittaa omia sovelluksia ja muodostaa yhteys AWS-pilvipalveluihin. Näiden jälkeen laiteympäristö oli valmis reuna–pilvi-integraatiota varten ja opinnäytetyön sovelluskehitystä varten.

Sovelluskehityksen jälkeen asensin osana opinnäytetyötä kehitetyn Java-pohjaisen reunasovelluksen laitteella suoritettavaan Greengrass-ympäristöön. Se vastasi yhteyden muodostamisesta PLC:n OPC UA -palvelimeen ja AWS IoT Core -palveluun. Reunasovelluksen Java-suoritusympäristönä käytettiin Amazon Corretto 21 -JDK:ta (LTS), joka tarjosi yhteensopivan suoritusympäristön Advanced RISC Machine (ARM) -arkkitehtuuriin perustavalle IOT2050-laitteelle. Lisäksi asensin laitteeseen paikallisen avoimen lähdekoodin Mosquitto MQTT -välityspalvelimen. Sen avulla voitiin toteuttaa kommunikointi pilvipalveluihin AWS MQTT Bridgen välityksellä sekä mahdollistaa paikallinen kommunikointi reunalla muiden MQTT-protokollaa tukevien laitteiden kanssa. Tämä mahdollisti UNS-arkkitehtuurin mukaisen hierarkkisen aiherakenteen ylläpitämisen koko järjestelmässä.

4.2 Käytetyt ohjelmistotyökalut ja kehitysympäristöt

Toteutuksessa käyttämäni työkalut, kehitysympäristöt ja ohjelmistokirjastot muodostivat arkkitehtuurin teknisen perustan. Ratkaisu jakautuu kahteen pääosaan: reunasovellukseen, joka toimii yhdyskäytävälaitteessa ja vastaa kenttätason datankeruusta ja esikäsittelystä, sekä pilvipalveluihin, joita AWS:n pilviympäristö tarjosi skaalautuvana alustana datan vastaanotolle, tallennukselle, analytiikalle ja visualisoinnille. Alaluvussa 4.2.1 esitellään reunatyökalut ja ohjelmistokirjastot, joiden avulla toteutin OPC UA -pohjaisen datankeruun, MQTT-viestinnän ja paikallisen esikäsittelyn. Alaluvussa 4.2.2 puolestaan keskitytään AWS-pilvipalveluihin ja hallintaympäristöihin, joissa painopiste oli palveluiden konfiguroinnissa, käyttöoikeuksien hallinnassa ja salaisuuksien turvallisessa käsittelyssä. Näin muodostui yhtenäinen, mutta selkeästi kahteen kerrokseen jäsennelty tekninen arkkitehtuuri. Reunassa toteutettiin ohjelmointi ja datankäsittely, kun taas pilvessä konfigurointi ja ohjelmalliset Lambda-ratkaisut täydensivät kokonaisuutta.

4.2.1 Reunasovelluksen kehitystyökalut ja kirjastot

Kehitin reunasovelluksen Eclipse IDE -ympäristössä Java-ohjelmointikielellä ja toteutin Java-projektin Maven-pohjaisena, mikä tarjosi valmiudet ohjelmistoriippuvuuksien hallintaan POM-tiedoston avulla. Maven mahdollisti laajasti käytettyjen avoimen lähdekoodin kirjastojen integroinnin osaksi kehitysympäristöä ja varmisti riippuvuuksien yhteensopivuuden. Käytin sovelluksen toteutuksessa useita keskeisiä kirjastoja, joiden ohjelmointirajapintoja sovelsin eri tarkoituksiin. Esimerkiksi Eclipse Milo ja Eclipse Paho tarjosivat valmiit funktiot OPC UA- ja MQTT-protokollien hallintaan. AWS Software Development Kit (SDK) v2 mahdollisti yhteydet AWS-pilvipalveluihin, kuten Secrets Manageriin, sekä Greengrass-ympäristössä laitteen sisäisen viestinnän Greengrass interprocess communication (IPC) -rajapinnan välityksellä.

Tukikirjastoja käytettiin muun muassa JSON-datan käsittelyyn, tietoturvaan ja lokien hallintaan. Kirjastojen tarjoamia luokkia ja metodeja kutsuttiin ja yhdistettiin osaksi sovelluksen omaa logiikkaa. Keskeiset kirjastot ja niiden käyttötarkoitukset on koottu taulukkoon 1. Näiden kirjastojen avulla sovelluksessa voitiin toteuttaa sekä teollisuuden standardeihin perustuvat viestintäprotokollat että tietoturvallinen integraatio AWS-pilvipalveluihin. Lisäksi valitut kehitystyökalut ja avoimen lähdekoodin

kirjastot mahdollistivat luvussa 2.3 kuvatun reunalaskennan periaatteiden toteuttamisen. Näiden mukaan laskentateho siirretään lähemmäksi datan syntypaikkaa, ja yhdyskäytävälaitteet tukevat eri viestintäprotokollien muuntamista sekä paikallisen laskentakapasiteetin hyödyntämistä (Singh ym. 2025).

TAULUKKO 1. Käytetyt ohjelmistokirjastot

Kirjasto	**Käyttötarkoitus**	**Kategoria**
Eclipse Milo	OPC UA -protokollan toteutus (IEC 62541): turvallinen yhteys, tilauspohjainen datankeruu ja komentoviestien välittäminen PLC:lle.	Teollinen viestintäprotokolla.
Eclipse Paho	MQTT-protokollan toteutus (ISO/IEC 20922): asiakas–julkaisijasovellus, joka hyödyntää julkaisu–tilaajamallia, mahdollistaa prosessidatan välityksen pilveen sekä komentoviestien vastaanoton pilvestä.	Teollinen viestintäprotokolla.
AWS SDK v2	Mahdollistaa yhteydet AWS-pilvipalveluihin, kuten Secrets Manageriin, sekä Greengrass-ympäristössä IPC-rajapinnan välityksellä.	AWS-integraatio.
Jackson & Gson	JSON-datan käsittely ja MQTT-hyötykuormien muodostaminen.	Datan serialisointi.
BouncyCastle	Kryptografiapalvelut: yksityiset avaimet (PKCS#8) ja varmenteet (PEM).	Tietoturva ja salaus.
SLF4J & Logback	Sovelluslokien hallinta ja viestintä.	Lokitus ja monitorointi.
Netty	Verkkoviestinnän taustakirjasto, jota AWS SDK ja Milo hyödynsivät.	Verkkoviestintä.
Apache HttpClient	HTTP-yhteyksien hallinta (AWS SDK:n taustariippuvuus).	Verkkoviestintä.

4.2.2 Pilvityökalut ja -palvelut

AWS-pilvipalveluiden käyttö jakautui hallintatyökaluihin ja varsinaisiin pilvipalveluihin. Pilviarkkitehtuurin toteutuksessa käytin kahta eri hallintatyökalua: verkkoselaimessa toimivaa AWS Management Console -käyttöliittymää sekä yhdyskäytävälaitteella Linux-terminaalissa käytettävää AWS Command Line Interfacea (AWS CLI). Määritin näiden avulla käyttöoikeudet, loin ja hallitsin resursseja sekä otin käyttöön ja konfiguroin pilvipalveluita. Järjestelmän arkkitehtuurissa hyödynsin useita AWS-pilvipalveluita, joita tarkastellaan tarkemmin luvussa 4.6. Ohjelmointia vaativat pilvipalveluratkaisut toteutin pääosin Lambda-funktioilla Node.js-ohjelmointikielellä hyödyntäen AWS SDK:ta sekä erillistä Lambda Layeria kirjastojen hallintaan. Lisäksi AWS IoT Coren Rules Enginessä ja Amazon Timestream-aikasarjatietokannassa käytin SQL-tyyppistä kyselykieltä datan edelleen välittämiseen, suodattamiseen ja analysointiin.

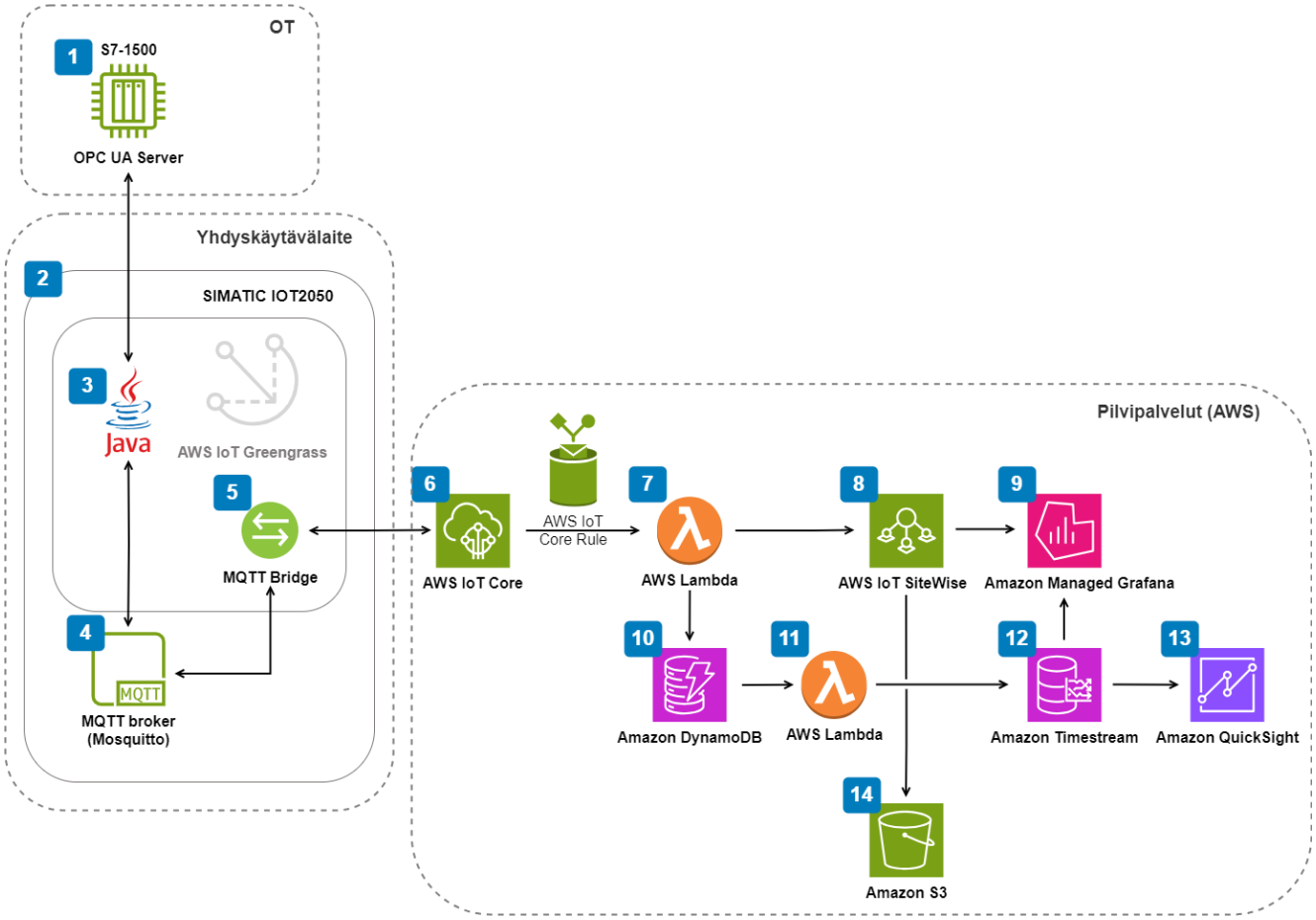
Pilviarkkitehtuurin rakentamisessa tarvittiin ohjelmointityön lisäksi laajaa ympäristön konfigurointia. Jokainen pilvipalvelu vaati erilliset määritykset ja käyttöoikeudet. Niiden yhteensovittaminen edellytti AWS-roolien ja -politiikkojen hallintaa AWS Identity and Access Managementin (IAM) avulla sekä pilvi- ja sovellusrajapintojen turvallista käyttöönottoa. Yhteenvetona voidaan todeta, että pilvipalvelut muodostivat arkkitehtuurin pilvikerroksen. AWS Management Console ja AWS CLI -hallintatyökalut mahdollistivat konfiguraation ja valvonnan, kun taas ohjelmointikielet ja kirjastot loivat pilvilogiikan, joka täydensi reunasovelluksen tuottamaa datavirtaa. Näin pilvipalveluiden käyttö toteutti luvussa 3.4 esiteltyä IIoT-arkkitehtuurin mallia, jossa pilvi toimii keskeisenä tasona datan tallennuksessa, käsittelyssä ja analytiikassa. Ratkaisu mahdollisti skaalautuvan ja kustannustehokkaan ympäristön, jossa AWS:n tarjoamat palvelut tukivat reaaliaikaista tiedonkulkua, turvallista käyttöoikeuksien hallintaa ja prosessidatan jatkoanalytiikkaa (Attaran ym. 2024, 4; Sithiyopasakul ym. 2024).

4.3 Järjestelmäarkkitehtuuri ja datavirran kulku

Konseptin arkkitehtuuri perustuu luvussa 3.1 kuvattuun malliin, jossa reuna- ja pilvilaskenta muodostavat toisiaan täydentävän kokonaisuuden (Dauda ym. 2024, 2–3). Lisäksi arkkitehtuuri noudatti luvussa 3.2 käsitellyn UNS-arkkitehtuurin periaatteita, jotka mahdollistavat standardoidun ja skaalautuvan tiedonhallinnan (Salcher ym. 2024). Arkkitehtuuri ja datavirran päävaiheet reunasta pilveen havainnollistetaan kuviossa 6, ja tarkempi kuvaus jokaisesta vaiheesta on esitetty liitteessä 1. Paikallinen Mosquitto MQTT -välityspalvelin (4) toimii järjestelmän yhteisenä tietotilana eli *single source of truth* -ratkaisuna, johon Java-ohjelma (3) julkaisee prosessidatan standardoidussa aiherakenteessa. Reunalla sovelletaan UNS:n keskeisiä periaatteita: ratkaisu on reunalähtöinen, sillä data kerätään ja esikäsitellään SIMATIC IOT2050-laitteella (2); kevyt, koska viestintä toteutetaan MQTT-protokollalla; poikkeamia raportoiva, sillä vain olennaisesti muuttuneet arvot välitetään eteenpäin; sekä avoimeen arkkitehtuuriin perustuva, sillä tiedonsiirto hyödyntää avoimia standardeja, kuten MQTT ja OPC UA. Näin mahdollistettiin järjestelmän laajennettavuus ja eri toimittajien ratkaisujen yhteensopivuus.

Kuviossa 6 havainnollistetaan, että prosessidata kerätään PLC:n OPC UA -palvelimelta (1), esikäsitellään SIMATIC IOT2050-laitteella (2) ja julkaistaan MQTT-välityspalvelimelle (4) UNS-rakenteen mukaisesti. MQTT Bridge (5) siirtää datan AWS IoT Coreen (6), joka toimii keskitettynä välityskerroksena pilvessä ja ohjaa datavirran edelleen AWS-pilvipalveluihin IoT Coren sääntöjen ja AWS Lambda-funktioiden (7, 11) avulla. AWS IoT SiteWise (8) vastaa prosessidatan mallinnuksesta ja Key Performance Indicator (KPI) -suorituskykymittareiden laskennasta. Amazon DynamoDB (10) tarjoaa

edullisen välivaraston lyhytaikaiselle datalle, kun taas Amazon Timestream (12) skaalautuvan tallennusratkaisun pitkäaikaiselle aikasarjadatalle. Amazon Managed Grafana (9) sekä Amazon QuickSight (13) mahdollistavat datan reaaliaikaisen ja pitkän aikavälin visualisoinnin. Lisäksi Amazon Simple Storage Service (Amazon S3) (14) toimii pysyvänä tallennusalustana esimerkiksi raakadatalle ja konfiguraatioille.



KUVIO 6. Datavirta teollisuusprosessista yhdyskäytävälaitteen kautta AWS-pilvipalveluihin

4.4 Reunasovellus

Kehitin osana opinnäytetyötä yhdyskäytävälaitteessa toimivan reunasovelluksen Java-ohjelmointikielellä hyödyntäen avoimen lähdekoodin kirjastoja. Käytetyt kirjastot on esitelty luvussa 4.2.1 (TAULUKKO 1). Sovellus suoritetaan yhdyskäytävälaitteella AWS IoT Greengrass -ympäristössä omana Greengrass-komponenttinaan. Sovelluksen arkkitehtuuri perustuu luvuissa 2.3 ja 3.2 mainittuihin reunalaskennan (Singh ym. 2025) ja UNS-arkkitehtuurin periaatteisiin (Mandidereza 2024; Salcher ym.

2024). Seuraavaksi kuvataan, miten nämä periaatteet toteutuivat käytännössä. Ensinnäkin dataa kerätään ja esikäsitellään reunalla, mikä edustaa reunalähtöisyyden periaatetta reunalaskennan mukaisesti. Toiseksi tiedonsiirto toteutetaan kevyellä ja resurssitehokkaalla julkaise–tilaa-mallia hyödyntävällä MQTT-protokollalla, joka on tyypillinen ratkaisu kevyeen viestintään. Kolmanneksi vain merkittävästi muuttuneet arvot lähetetään eteenpäin, mikä vähentää datan määrää ja resurssien kulutusta. Neljänneksi viestit noudattavat hierarkkista nimeämismallia, joka mahdollistaa skaalautuvuuden ja yhtenäisen tiedonhallinnan eri järjestelmäkomponenttien välillä.

Päädyin käyttämään sovelluksessa modulaarista arkkitehtuuria, jossa kukin luokka hoitaa selkeästi rajattua tehtävää. Tavoitteena oli osoittaa, kuinka reunasovellus voidaan toteuttaa konfiguroitavana ja uudelleen käytettävänä Greengrass-ympäristössä toimivana komponenttina. Ratkaisu mahdollistaa sovelluksen helpon ylläpidon, testattavuuden ja jatkokehityksen. Lisäksi ratkaisu tukee laajennettavuutta muihin protokolliin tai tietolähteisiin ilman suuria rakenteellisia muutoksia. Ohjelman rakenne on jaettu toiminnallisiin kokonaisuuksiin (TAULUKKO 2), jotka kattavat datankeruun, viestinnän, konfiguraation hallinnan ja tietomallien käsittelyn. Näiden kokonaisuuksien tarkempi toteutus avataan alaluvuissa 4.4.1 ja 4.4.2 sekä havainnollistetaan liitteessä 2 esitettävässä arkkitehtuurikaaviossa.

TAULUKKO 2. Luokat Java-ohjelmassa.

Toiminnallisuus	**Luokat**	**Kuvaus**
Pääluokka	MainRunner	Käynnistää sovelluksen ja alustaa palvelut (toiminta kuvattu pseudokoodina liitteessä 3).
OPC UA	OpcUaClientConnector	Luo ja ylläpitää OPC UA -yhteyttä palvelimeen.
	ManagedOpcUASubscription	Hallinnoi tilausten luontia, datan vastaanottoa ja suodattaa vain merkittävästi muuttuneet arvot.
	OpcUaPublisher	Julkaisee komentoviestit OPC UA -palvelimelle.
MQTT	MqttPubSubClient	Yhdistää MQTT-välityspalvelimeen ja hallitsee viestinvälityksen.
	MqttMessageHandler	Käsittelee ohjelman muodostamat MQTT-viestit.

(jatkuu)

TAULUKKO 2. Luokat Java-ohjelmassa (jatkuu).

Toiminnallisuus	Luokat	Kuvaus
Konfiguraatio	GreengrassConfigReader	Lukee sovelluksen asetukset AWS Greengrass -konfiguraatiosta.
	ProgramSettings	Ylläpitää sovelluksen sisäisiä asetuksia.
	TagReader	Lukee ja tulkitsee tagikonfiguraation tiedostosta.
	AwsSecretReader	Hakee tunnistetiedot AWS Secrets Managerista.
Tietomallit	Tag	Määrittelee yksittäisen mittaus- tai ohjaustiedon rakenteen.
	CommandMessage	Kapseloi komentoviestit OPC UA -kirjoituksia varten.

4.4.1 Sovelluksen konfigurointi ja asetukset

Sovelluksen toiminta on määritettävissä JSON-pohjaisella konfigurointitiedostolla sekä AWS IoT Greengrass Coren IPC-rajapinnan välityksellä AWS Management Consolesta. Tämä mahdollistaa sovelluksen toiminnan optimoinnin sekä uudelleen käytettävyyden eri ympäristöissä. IPC-rajapinnan välityksellä sovellus hakee suoritusaikaiset asetukset, kuten yhteysosoitteet, varmenteiden polut, MQTT-aiheet ja UNS-hierarkian tunnisteet. Näiden avulla varmistetaan, että sovellus voi lukea ja käyttää ajantasaisia konfiguraatioita ilman, että niitä tarvitsee määrittää pysyvästi ohjelmakoodiin. JSON-tiedostolla konfiguroidaan OPC UA -palvelimelta tilattavat solmut ja niiden asetukset. Nämä asetukset vaikuttavat tilausten muodostamiseen ja datan käsittelyyn ohjelmassa. Konfiguraatiotiedosto sisältää JSON-muodossa olevan listan kaikista tilattavista solmuista ja niiden yksilöllisistä asetuksista. Liitteessä 4 esitellään esimerkki yksittäisestä tietueesta ja sen attribuuteista.

Sovelluksen konfiguroitavuus tukee luvuissa 2.3 ja 3.2 esiteltyjä reunalaskennan ja UNS-arkkitehtuurin periaatteita, joissa datan keruu ja lähetys optimoidaan resurssien säästämiseksi ja siirtokustannusten pienentämiseksi (Salcher ym. 2024; Singh ym. 2025). Tällä tavalla voidaan esimerkiksi säätää datan päivitystaajuutta ja kynnysarvoja siten, että vain oleelliset muutokset lähetetään eteenpäin, mikä vähentää pilveen siirrettävän datan määrää ja verkon kuormitusta. Ohjelman konfiguraationhallinta perustuu sekä laitteella olevan Greengrass-ympäristön sisäiseen IPC-rajapintaan, että JSON-muotoisten konfiguraatiotiedostojen käsittelyyn. *GreengrassConfigReader*-luokka lukee ohjelman käynnistyessä Greengrass-komponentin asetukset. *ProgramSettings*-luokka ylläpitää sisäisiä parametreja sovelluksen

suorituksen aikana. *TagReader*-luokka puolestaan lukee konfiguraatiotiedostot, ja muodostaa sen perusteella tilauslistan OPC UA -solmuista. Vahvaa autentikointia varten tarvittavat tunnistetiedot, kuten OPC UA -käyttäjätunnukset ja salasanat, haetaan turvallisesti *AwsSecretReader*-luokan avulla suoraan AWS Secrets Manager -palvelusta.

4.4.2 Datankeruu ja tiedonsiirto: OPC UA ja MQTT

Toteutin sovelluksen datankeruun OPC UA -protokollalla ja tiedonsiirron MQTT-välityspalvelimelle MQTT-protokollalla, joka toimi järjestelmän tapahtumienvälittäjänä UNS-arkkitehtuurin mukaisesti. Näin varmistin, että prosessidata saatiin luotettavasti sekä tehokkaasti PLC:ltä ja välitettyä edelleen AWS IoT Coreen sekä muille mahdollisille reunalla oleville tilaajille. Konseptissa käytetyt OPC UA- ja MQTT-protokollat perustuvat luvussa 3.3 esiteltyihin teollisen internetin viestintäprotokolliin, joissa korostuvat avoimuus, yhteentoimivuus ja keveys.

OPC UA soveltuu erityisesti luotettavaan ja tietoturvalliseen tiedonsiirtoon sekä mahdollistamaan integraation perinteisten automaatiojärjestelmien että pilvi- ja reunaratkaisujen välillä (OPC Foundation 2024). MQTT puolestaan on kevyt ja skaalautuva tapahtumapohjainen protokolla, joka sopii hyvin reunalaskentaympäristöihin, joissa datansiirron tehokkuus ja resurssien säästö ovat keskeisiä tavoitteita (Desbiens 2023; Singh ym. 2025). Lisäksi MQTT mahdollistaa UNS-arkkitehtuurin mukaisen tapahtumapohjaisen viestinvälityksen (Salcher ym. 2024). Näiden protokollien yhdistäminen mahdollisti arkkitehtuurin, joka noudattaa luvuissa 2.3 ja 3.2 kuvattuja reunalaskennan ja UNS-arkkitehtuurin periaatteita, jossa data suodatetaan ja siirretään vain tarvittaessa, mikä vähentää verkon kuormitusta ja pilvipalvelun kustannuksia.

Toteutin protokollien käsittelyn ohjelmassa protokollakohtaisilla kirjastoilla. OPC UA -asiakasohjelman toteutuksessa käytin avoimen lähdekoodin *Eclipse Milo* -kirjastoa, joka on Eclipse Foundationin ylläpitämä Java-kirjasto OPC UA -protokollan toteuttamiseen ja tarjoaa tehokkaan tuen sekä OPC UA -asiakas- että palvelintoiminnallisuuksille (Eclipse Milo Project). Asiakasohjelma tilasi OPC UA -palvelimelta solmuja, jotka välittivät simuloitua prosessidataa PLC:ltä. OPC UA -asiakasohjelmassa *OpcUaClientConnector*-luokka huolehtii yhteyden muodostamisesta ja ylläpidosta palvelimeen. *ManagedOpcUASubscription*-luokka hallitsee tilaukset ja suodattaa datan UNS-periaatteen mukaisesti: vain merkittävästi muuttuneet välitetään eteenpäin. Erillinen *OpcUaPublisher*-luokka välittää ohjauskomennot takaisin palvelimelle.

Eclipse Milo -kirjasto tarjosi valmiit mekanismit muun muassa yhteyksien hallintaan, yksittäisten arvojen luku- ja kirjoitusoperaatioihin sekä jatkuvien tilausten toteuttamiseen ja eri tietotyyppien käsittelyyn. Tämä nopeutti kehitystä ja varmisti OPC UA -protokollan standardinmukaisen toiminnan. Lisäksi kirjasto tuki OPC UA:n turvaprofiileja ja sertifikaattipohjaista autentikointia, kuten Basic256Sha256 + Sign&Encrypt -profiilia, jota sovellus hyödyntää yhteyden salaamiseen. Kirjaston valintaan vaikutti sen laaja dokumentaatio, aktiivinen kehitys sekä hyvä yhteensopivuus Java-pohjaisen reunasovelluksen kanssa. Näiden ominaisuuksien ansiosta Eclipse Milo soveltui hyvin tämän konseptin OPC UA -asiakasohjelman perustaksi.

Vaikka Sparkplug B -protokollalla on etuja UNS-arkkitehtuurin näkökulmasta, päädyin kuitenkin perinteiseen MQTT-protokollaan sen yksinkertaisuuden ja keveyden vuoksi. Sparkplug B:n lisäominaisuudet, kuten tilanhallinta ja binäärimuotoinen viestirakenne, eivät olleet välttämättömiä tämän työn rajatussa konseptissa. MQTT mahdollisti nopeamman integraation, selkeän virheenkorjauksen ja sujuvan testauksen kehitysvaiheessa. Käytin työssä MQTT 3.1.1-protokollaa, vaikka uudempi MQTT 5 olisi ollut saatavilla sekä Mosquitto-välityspalvelimessa (Eclipse Mosquitto) että AWS IoT Coressa (MQTT 2025). MQTT 3.1.1 on edelleen yleisesti käytetty IoT-sovelluksissa ja sen tueksi on saatavilla runsaasti työkaluja ja dokumentaatiota. Lisäksi sovellus hyödyntää QoS 0 -tasoa, jossa viesti lähetetään kerran ilman vastaanottajan kuittausta (MQTT 2025). QoS 0 soveltui hyvin konseptin tavoitteisiin, sillä dataa siirretään tiheään tahtiin, yhteys on vakaa, eikä yksittäisen viestin katoaminen ole kriittistä. Tämän etuna oli myös se, että uudelleenlähetyksiä ei muodostu, mikä vähentää duplikaatteja ja mahdollistaa nopean viestinvälityksen ilman ylimääräistä kuormitusta.

Käytin sovelluksen MQTT-toteutuksessa avoimen lähdekoodin Eclipse Paho -kirjastoa, joka on Eclipse Foundationin ylläpitämä kirjasto MQTT:n julkaisu- ja tilausmallin toteuttamiseen (Eclipse Paho Project). Kirjasto tarjosi sovellukseen valmiit mekanismit yhteyden muodostamiseen, tilausten hallintaan ja viestien välittämiseen. Tämä nopeutti kehitystyötä ja varmisti MQTT-protokollan standardinmukaisen toiminnan. MQTT-asiakasohjelmassa *MqttPubSubClient*-luokka muodostaa yhteyden paikalliseen Mosquitto MQTT-välityspalvelimeen, ylläpitää yhteyttä ohjelman suoritusajan ja hallitsee julkaisut sekä tilaukset. *MqttMessageHandler*-luokka puolestaan keskittyy viestien ja hyötykuormien muodostamiseen UNS-rakenteen mukaisesti, mahdollistaen yhtenäisen ja skaalautuvan aiherakenteen. Näiden ominaisuuksien ansiosta Eclipse Paho soveltui hyvin tämän konseptin MQTT-asiakasohjelman perustaksi.

Sovellus julkaisee dataa kahteen MQTT-aiheeseen: toinen aihe käsittelee simuloidun prosessin mittaus- ja ohjaustietoja, ja toinen sovelluksen tilatietoja (TAULUKKO 3). Lisäksi MQTT-asiakasohjelma toimii tilaajana AWS IoT Corelta tulevalle ohjausaiheelle (TAULUKKO 3). Tämän aiheen välityksellä vastaanotetaan esimerkiksi prosessia optimoivia ohjauskomentoja, jotka perustuvat mittausdataan. Käytetyt MQTT-aiheet noudattavat UNS-arkkitehtuurin mukaista hierarkkista rakennetta. Lisäksi niissä hyödynnetään Sparkplug B -spesifikaation mukaisia viestityyppien nimeämiskäytäntöjä, kuten *DDATA*, *DCMD*, *STATE*, mutta ilman varsinaista Sparkplug B -protokollan tarkkaa aiherakennetta tai binaarimuotoista hyötykuormaa. Taulukossa 3 esitetään toteutuksessa käytetyt MQTT-aiheet, jotka noudattavat UNS-hierarkkista nimeämismallia ja Sparkplug B -viestityyppien käytäntöjä.

TAULUKKO 3. Esimerkki toteutuksen MQTT-aiheista.

Aihealue	**MQTT-aihe (topic)**
Prosessin mittaus- ja ohjaustiedot.	mqtt/Company1/Site1/Production/PumpSystem1/IOT2050/DDATA
Sovelluksen tilaviestit, kuten OPC UA -palvelimen tila.	mqtt/Company1/Site1/Production/PumpSystem1/IOT2050/STATE
AWS IoT Corelta tulevat ohjauskomennot.	mqtt/Company1/Site1/Production/PumpSystem1/IOT2050/DCMD

Jokainen datapiste ja MQTT-aihe noudattaa yhtenäistä UNS-hierarkiaan perustuvaa nimeämismallia. Esimerkiksi *Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/ActivePower_kW* ilmaisee selkeästi mittauspisteen kontekstin ja sijainnin hierarkiassa. Tämä rakenne varmistaa, että tieto on yksiselitteisesti tunnistettavissa ja liitettävissä oikeaan kontekstiin sekä reunasovelluksessa että pilvipalveluissa. UNS-hierarkkinen nimeäminen mahdollistaa lisäksi järjestelmän laajentamisen uusiin laitetyyppeihin tai tuotantolinjoihin lisäämällä uusia aihetasoja ilman muutoksia olemassa olevaan viestintämalliin.

Lisäksi sovellus on konfiguroitavissa julkaisemaan data joko yksittäisinä MQTT-viesteinä tai eräviesteinä, joissa useampi mittausarvo yhdistetään yhdeksi viestiksi (LIITE 5). Tämä ratkaisu tukee UNS-arkkitehtuurin periaatteita, sillä se mahdollistaa datan siirron joko mahdollisimman reaaliaikaisesti tai kustannustehokkaasti suurempina erinä. Liitteiden esimerkeissä konkretisoidaan, kuinka prosessidata siirtyy PLC:ltä OPC UA -palvelimen avulla MQTT-välityspalvelimen kautta pilvipalveluun (LIITE 6) ja palautuu tarvittaessa ohjauskomentona takaisin prosessiin (LIITE 7). Luvussa esitetyt ratkaisut osoittavat, kuinka Eclipsen Milo- ja Paho-kirjastojen yhdistäminen UNS-mukaiseen aiherakenteeseen

mahdollistaa kokonaisuuden, joka on sekä teknisesti toimiva että laajennettavissa muihin ympäristöihin.

4.5 Kyberturvallisuuden ratkaisut

IIoT-yhdyskäytävälaitteen ja pilvipalveluiden välisessä integraatiossa kyberturvallisuus on keskeinen osa kokonaisarkkitehtuuria (Gebremichael ym. 2020, 152352–152353). Konseptissa huomioitiin sekä reuna- ja pilvikerroksen tietoturvavaatimukset, jotta prosessidatan siirto ja ohjauskomennot voidaan toteuttaa luotettavasti ja turvallisesti. Kyberturvallisuuden ratkaisut kattoivat sertifikaattien ja avainten hallinnan, yhteyksien salauksen, käyttöoikeuksien rajauksen sekä tunnistetietojen turvallisen käsittelyn. Toteutin OPC UA -yhteyden Eclipse Milo -kirjaston turvaprofiililla Basic256Sha256 + Sign&Encrypt, joka varmistaa viestien luottamuksellisuuden, eheyden ja alkuperän, kuten luvussa 3.3.1 käsiteltiin. Loin yhdyskäytävälaitteelle Siemensin TIA Portalin Certificate Manager -työkalulla X.509-pohjaisen asiakasvarmenteen ja muunsin siihen liittyvän yksityisen avaimen TIA Portalin ulkopuolella PKCS#8-yhteensopivaan muotoon Milo-kirjaston kanssa yhteensopivaksi. Sertifikaattien luottoketjun hallinnan toteutin Milo-kirjaston DefaultTrustListManager-mekanismilla, joka mahdollistaa luotettujen ja estettyjen varmenteiden hallinnan kansiorakenteen avulla.

AWS IoT Core -yhteys käyttää TLS-suojausta, joka on yleisesti käytetty tiedonsiirron salaamiseen (Gebremichael ym. 2020, 152357). Yhteyden muodostamisessa laite tunnistautuu AWS-pilvipalvelussa Greengrass-komponentille rekisteröidyn X.509-varmenteen ja yksityisen avaimen avulla. MQTT bridge -ratkaisu käyttää salattua TLS-yhteyttä portissa 8883, mikä estää salaamattoman liikenteen kulun ja varmistaa, että dataliikenne pysyy suojattuna myös välityspalvelimen kautta. Tunnistetietojen käsittelyssä reunasovellus hyödyntää AWS Secrets Manager -palvelua. Sen avulla OPC UA -palvelimen käyttäjätunnukset ja salasanat tallennetaan turvallisesti pilveen ja haetaan ajonaikaisesti yhdyskäytävälaitteelle sovelluksen käyttöön. Näin varmistetaan, että kiinteitä salasanoja ei tarvitse säilyttää konfiguraatiotiedostoissa tai koodissa. Saadut tunnistetiedot käsitellään vain muistissa, ja ne nollataan käytön jälkeen tietovuotoriskin minimoimiseksi.

Tallensin sertifikaatit ja yksityiset avaimet yhdyskäytävälaitteella suojattuun tiedostojärjestelmään sekä suojasin ne Linux-käyttöjärjestelmän käyttöoikeusrajoituksilla ja erillisellä kansiorakenteella. Tällöin vain Greengrass-komponentti ja järjestelmänvalvojakäyttäjät pääsevät niihin käsiksi. Tämä todettiin konseptitason ratkaisuun käytännössä toimivaksi vaihtoehdoksi HSM-pohjaisille toteutuksille,

vaikka HSM olisi parempi ratkaisu teolliseen tuotantokäyttöön. Kokeilin AWS:n tarjoamaa testaukseen tarkoitettua ohjelmallista HSM:ää (SoftHSM), mutta integrointi MQTT-välityspalvelimeen ja Milo-kirjastoon osoittautui haasteelliseksi. Edellä mainittujen teknisten ratkaisujen lisäksi toteutuksessa hyödynnettiin Linux-käyttöjärjestelmän käyttöoikeusmalleja, kuten user/group-jako ja oikeuksien rajaus sekä Greengrass-komponentin omaa eristystä. Näiden avulla varmistettiin, että vain oikeutetut prosessit pääsivät käsiksi salaisiin tietoihin ja että sovellus pysyi eristettynä muista laitteella ajettavista prosesseista. Ratkaisu tarjosi käytännössä kahden tason suojauksen: käyttöjärjestelmän tasolla rajatut käyttöoikeudet sekä Greengrass-ympäristön tarjoaman komponenttikohtaisen eristyksen.

4.6 Pilviarkkitehtuuri ja käytetyt pilvipalvelut

Järjestelmän kokonaisarkkitehtuuri reunasta pilveen esiteltiin luvussa 4.3. Tässä osuudessa syvennytään toteutuksen pilviosuuteen sekä käytettyihin AWS-pilvipalveluihin. Valitsin pilvialustaksi Amazon Web Services (AWS), koska se tarjoaa laajan valikoiman IoT- ja teollisuusratkaisuihin soveltuvia palveluita. Se mahdollistaa UNS-arkkitehtuurin toteuttamisen vapaasti määriteltävällä aiherakenteella ja MQTT-pohjaisella viestinnällä. Lisäksi se tarjoaa laajan MQTT-tuen ja kustannustehokkaat mallit datan hallintaan. Valintaa vahvisti se, että käytettyyn SIMATIC IOT2050 -yhdyskäytävälaitteeseen on saatavilla valmis dokumentaatio AWS-integraatiota varten (SIMATIC IOT2050: Getting Started Guide for AWS IoT Greengrass 2024). AWS:n valintaa tuki myös sen markkina-asema, sillä se on tutkimusten mukaan (Eclipse IoT & Embedded Developer Survey 2024) eniten käytetty pilvialusta IoT-ratkaisuissa. Tämä viittaa myös sen vakauteen ja luotettavuuteen. Lisäksi AWS tarjoaa laajasti dokumentaatiota ja aktiivisen yhteisön tuen IoT- ja IIoT-ratkaisuille.

Konseptin pilviarkkitehtuuri rakentuu kerroksittain reunalaitteelta pilveen. Reunalla SIMATIC IOT2050 kerää OPC UA -palvelimelta prosessidataa. Laitteessa data suodatetaan UNS-periaatteen mukaisesti lähettämällä vain merkittävästi muuttuneet arvot. Data siirretään AWS IoT Coreen, jossa se käsitellään AWS Rules Enginen avulla ja ohjataan edelleen AWS Lambda-funktioihin. Näiden avulla data tallennetaan puskurina ja välivarastona toimivaan DynamoDB-tietokantaan. Pysyvään säilytykseen data tallennetaan Amazon Timestream-aikasarjatietokantaan. Lisäksi data ohjataan IoT SiteWise -palveluun hierarkkisen mallinnuksen ja KPI-laskennan mahdollistamiseksi. Amazon S3 toimii pitkäaikaissäilytyksenä ja konfiguraatioiden tallennuspaikkana. Pilviresurssien rooleja ja käyttöoikeuksia hallitaan IAM- ja IAM Identity Center -palveluiden avulla. Nämä yhdessä AWS Secrets Managerin

kanssa tukevat datan hallintaa ja turvallisuutta. Palveluiden vianhaussa ja seurannassa käytetään CloudWatchia. Visualisointi toteutettiin Amazon Managed Grafanan ja QuickSightin avulla.

4.6.1 IoT-reunapalvelut – AWS IoT Greengrass

Valitsin AWS IoT Greengrassin reunaratkaisun keskeiseksi palveluksi, koska se mahdollistaa oman Java-komponentin suorittamisen SIMATIC IOT2050 -laitteessa. Lisäksi se tarjosi valmiin integraation AWS IoT Core -palvelun avulla muihin AWS-pilvipalveluihin. Greengrassin avulla voitiin toteuttaa kokonaisuus, jossa OPC UA -palvelimelta kerätty prosessidata esikäsiteltiin paikallisesti UNS-periaatteiden mukaisesti ja siirrettiin kustannustehokkaasti pilveen. Tämä mahdollisti joustavan arkkitehtuurin, jossa paikallinen Java-ohjelma ja pilvipalveluiden tarjoamat resurssit täydensivät toisiaan.

AWS IoT Greengrass on avoimen lähdekoodin ajonaikainen ympäristö ja pilvipalvelu, jonka avulla voidaan rakentaa, ottaa käyttöön ja hallita reunasolmuihin ja yhdyskäytävälaitteisiin asennettavia ohjelmistoja ja palveluita. Se on laajasti käytetty IoT-sovelluksissa, kuten älykodeissa, tehtaissa, ajoneuvoissa ja yrityksissä. Greengrass mahdollistaa datan paikallisen prosessoinnin, viestinnän, tiedonhallinnan ja koneoppimiseen (ML) perustuvan päätöksenteon. Lisäksi se tarjoaa valmiita komponentteja sovelluskehitykseen. Greengrassin avulla reunasolmut ja yhdyskäytävälaitteet voivat kommunikoida turvallisesti AWS-pilvipalveluiden sekä kolmannen osapuolen palveluiden kanssa. Lisäksi ohjelmistoja voidaan hallita ja päivittää etänä ilman tarvetta laiteohjelmistopäivityksille. Greengrass koostuu pilvipalvelusta ja ohjelmistojakelusta, joiden keskeiset osat on koottu taulukkoon 4 (AWS IoT Greengrass FAQs 2025.) Taulukossa 5 on esitelty konseptissa käytetyt AWS IoT Greengrass -komponentit, mukaan lukien kehitetystä Java-ohjelmasta toteutettu Greengrass-komponentti.

TAULUKKO 4. AWS IoT Greengrass -ohjelmistoarkkitehtuurin komponentit (AWS IoT Greengrass FAQs 2025).

Komponentti	**Käyttötarkoitus**	**Suoritusympäristö**
AWS IoT Greengrass Core	Tarjoaa paikallisia palveluita, kuten laskenta, viestintä, tila ja tietoturva sekä kommunikoi SDK-laitteiden kanssa.	64-bittiset x86- tai ARM-prosessorit, joissa on yleiskäyttöinen käyttöjärjestelmä (esim. Linux).
AWS IoT Device SDK	Mahdollistaa laitteiden paikallisen kommunikoinnin Greengrass Core -komponentin kanssa.	Lähes kaikki laitteet, jotka tukevat C++, Node.js-, Java- tai Python-ohjelmointikieliä.
AWS IoT Greengrass SDK	Mahdollistaa Lambda-funktioiden tai SDK:lla kehitetyn ohjelman vuorovaikutuksen paikallisten palveluiden kanssa Greengrass Core -laitteessa.	Lambda-funktioiden tai SDK:lla kehitettyjen ohjelmistojen sisällä, jotka on asennettu Greengrass Core -laitteelle.

TAULUKKO 5. Konseptissa käytetyt AWS IoT Greengrass -komponentit.

Komponentti	**Käyttötarkoitus**	**Huomio**
aws.greengrass.Nucleus	Greengrassin ydinkomponentti, joka mahdollistaa Core-ympäristön suorittamisen laitteessa.	Pakollinen osa Greengrass-järjestelmää.
aws.greengrass.Cli	Komentorivityökalu komponenttien paikalliseen hallintaan ja käynnistämiseen.	Helpottaa kehitystä ja testauksia.
aws.greengrass.clientdevices.Auth	Todentaa ja valtuuttaa Greengrass-asiakaslaitteiden toiminnot.	Turvallisuuden peruskomponentti.
aws.greengrass.clientdevices.IPDetector	Tunnistaa automaattisesti IoT Coreen yhdistettyjen asiakaslaitteiden IP-osoitteet ja julkaisee tiedot paikalliseen MQTT-aiheeseen.	Vähentää manuaalista IP-hallintaa ja tukee dynaamisia ympäristöjä.
aws.greengrass.clientdevices.mqtt.Bridge	Mahdollistaa kaksisuuntaisen MQTT-viestinnän Greengrassin ja AWS IoT Coren välillä.	Toimii siltana paikallisen verkon ja pilven välillä.
iiot.greengrass.edge.opcua.mqtt (kehitetty Java-ohjelma)	Opinnäytetyössä kehitetty Java-ohjelma, joka lukee OPC UA -palvelimen prosessidataa, suodattaa sen ja välittää pilveen vain muuttuneet arvot.	Kehitetty tätä konseptia varten.

4.6.2 IoT-yhteydet ja viestinvälitys – AWS IoT Core

Käytin konseptissa AWS IoT Corea keskeisenä välityskerroksena, joka yhdistää yhdyskäytävälaitteen ja pilvipalvelut suojatun MQTT-yhteyden (MQTT over TLS) kautta. Autentikointi perustuu AWS IAM -käyttöoikeuksiin ja X.509-sertifikaatteihin. AWS IoT Core vastaanottaa datan yhdyskäytävälaiteelta ja reitittää datan määritettyjen Rules Engine -sääntöjen avulla Lambda-funktioille. Näiden välityksellä data välittyy edelleen muihin pilvipalveluihin jatkokäsittelyä varten. Rules Engine hyödyntää SQL-tyyppistä syntaksia viestien suodattamiseen ja reitittämiseen. Määritin jokaiselle säännölle oman kyselyn ja toimintona Lambda-funktion kutsun. Esimerkki yhdestä määritellystä säännöstä, joka reitittää kaiken prosessidatan DynamoDB-tietokantaan, on esitetty taulukossa 6.

TAULUKKO 6. Esimerkkisääntö AWS IoT Coren Rules Enginessä.

Osa-alue	**Arvo**
Rule description	Writes incoming MQTT data to DynamoDB as temporary storage before batch writing to Timestream.
SQL version	2016-03-23
SQL statement	SELECT *, topic() as topic FROM 'mqtt/Company1/Site1/Production/PumpSystem1/IOT2050/DDATA'
Action 1	Lambda (Send a message to a Lambda function)
Lambda function	MQTTTODynamoDB
Lambda function version	$LATEST

AWS IoT Core on hallinnoitu alusta, joka mahdollistaa IoT-laitteiden turvallisen ja skaalautuvan yhteydenpidon pilvisovelluksiin ja muihin laitteisiin. Amazonin mukaan palvelu kykenee hallinnoimaan globaalisti jopa miljardeja laitteita ja käsittelemään biljoonia viestejä, mikä mahdollistaa massiivisten IoT-ratkaisujen rakentamisen ilman omia palvelinresursseja. AWS IoT Coreen voidaan liittää erilaisia laitteita, kuten antureita, toimilaitteita, sulautettuja järjestelmiä (esimerkiksi SIMATIC IOT2050) sekä älykkäitä kodinkoneita, jotka muodostavat yhteyden IoT Coreen suojatun yhteyden kautta. Se tukee standardien mukaisia kommunikaatioprotokollia, kuten HTTP, MQTT, WebSocket ja LoRaWAN. Kommunikointi IoT Coren kanssa on turvallista, sillä yhteydet käyttävät TLS-salausta tietoturvallisen yhteyden varmistamiseksi. Kaikki viestintä on salattua, ja palvelu vaatii myös asiakkaita käyttämään vahvaa autentikointia, kuten X.509-sertifikaattia, AWS IAM -käyttöoikeuksia tai kolmannen osapuolen autentikointia AWS Cogniton avulla. (AWS IoT Core FAQs 2025.)

IoT Core tarjoaa kaksisuuntaisen, turvallisen ja matalaviiveisen tiedonsiirron yhdistettyjen laitteiden ja pilvisovellusten välillä. Lisäksi sen sisäänrakennettu Rules Engine mahdollistaa yhdistettyjen laitteiden lähettämän datan jatkuvan käsittelyn, suodattamisen, muuntamisen sekä uudelleenreitittämisen sääntöjen perusteella. Data voidaan reitittää muihin AWS-pilvipalveluihin, kuten DynamoDB-, Kinesis-, Lambda-, SNS-, SQS- ja CloudWatch-palveluihin, jatkokäsittelyä, tallennusta tai analytiikkaa varten. Tarvittaessa dataa voidaan Lambda-funktioiden avulla välittää myös AWS:n ulkopuolisiin palveluihin. (AWS IoT Core FAQs 2025.) AWS IoT Core sallii oletusarvoisesti yhdellä tilillä jopa 500 000 aktiivista MQTT-tilausta kaikkien laitteiden ja yhteyksien välillä. Palvelu voi oletusarvoisesti vastaanottaa ja lähettää jopa 20 000 viestiä sekunnissa, ja mikäli tämä kapasiteetti ei riitä, rajoja voidaan nostaa AWS:n tuen kautta tarpeen mukaan. (AWS IoT Core endpoints and quotas 2025.)

IoT Coren Device Shadow -palvelu mahdollistaa yhdistettyjen laitteiden tilatietojen säilyttämisen ja hallinnan pilvessä. Pilvi- ja mobiilisovellukset voivat kysellä laitteen tilaa ja lähettää laitteelle komentoja REST API -rajapinnan kautta, vaikka laite ei olisi juuri sillä hetkellä yhteydessä verkkoon. Kun laite muodostaa yhteyden uudelleen, se vastaanottaa Device Shadow -palvelusta viimeisimmän tilatiedon ja päivittää oman tilansa vastaamaan sitä. Tämän jälkeen laite julkaisee päivitetyn tilan takaisin pilveen. Tämä mahdollistaa sovelluksille jatkuvan pääsyn laitteen tilatietoihin myös katkonaisessa verkkoyhteydessä. (AWS IoT Device Shadow service 2025.)

4.6.3 Datan siirto pilvipalvelujen välillä – AWS Lambda

Datan saavuttua pilveen keskitettynä välikerroksena toimivalle AWS IoT Corelle, tarvitaan vielä palvelu, joka siirtää ja muokkaa tätä dataa ennen sen siirtämistä muihin AWS-palveluihin. Tähän tarpeeseen käytin AWS Lambda -funktioita. Data ohjataan IoT Coren Rules Enginen -sääntöjen perusteella eri Lambda-funktioille käyttötarkoituksen mukaan. Hyödynsin Lambdaa useissa tiedonkäsittelyn vaiheissa: datan jatkokäsittelyssä ja siirtämisessä AWS IoT SiteWise -palveluun, Amazon DynamoDB-tietokantaan (LIITE 8) ja Amazon Timestream -aikasarjatietokantaan ajastetusti. Toteutin ajastuksen Amazon EventBridgen avulla, joka mahdollistaa tapahtumapohjaisten ja aikaperusteisten työnkulkujen rakentamisen (Amazon EventBridge 2025). Lisäksi hyödynsin Lambdaa tietokantojen hallinnassa ja datan aggregoinnissa, kuten kumulatiivisen energiamittarin kulutuksen laskennassa. Käytetyt Lambda-funktiot on esitetty taulukossa 7. Toteutin ne Node.js versiolla 22.x hyödyntäen AWS SDK -kirjastoa.

TAULUKKO 7. Konseptissa käytetyt AWS Lambda -funktiot.

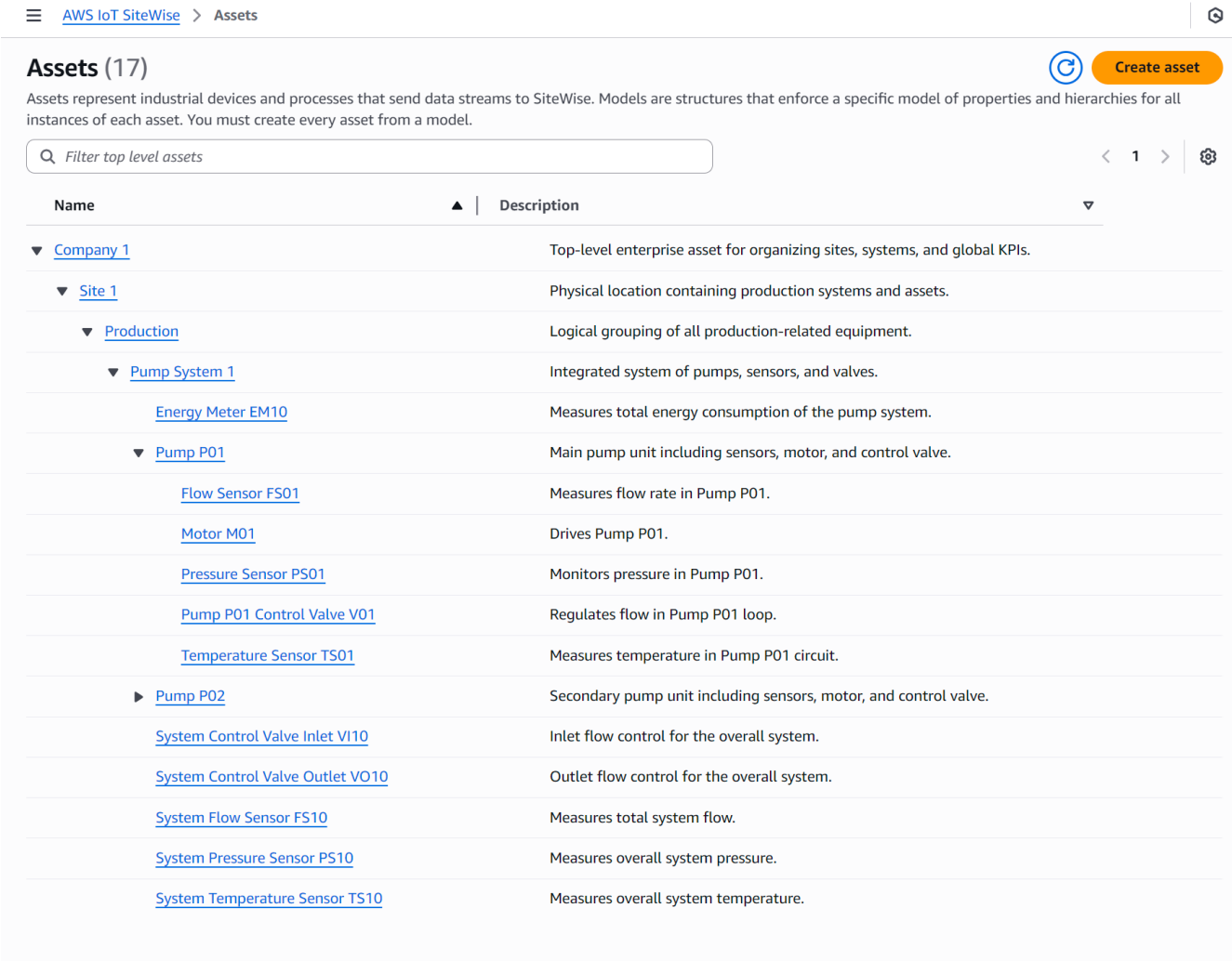
Funktio	Käyttötarkoitus	Kohdepalvelu	Lisäselvitys
MQTTToIoTSiteWise	Siirtää IoT Coreen saapuvan datan IoT SiteWise -palveluun.	AWS IoT SiteWise	Toteutuksen datamallien päivitys.
MQTTToDynamoDB	Siirtää IoT Coreen saapuvan datan välivarastoon DynamoDB-tietokantaan.	Amazon DynamoDB	Data tallennetaan välimuistiin eräajoa varten (LIITE 8).
DynamoDBDataToTimestream	Siirtää datan eräajona DynamoDB:stä Timestream-aikasarjatietokantaan.	Amazon Timestream	Mahdollistaa kustannustehokkaan pitkäaikaistallennuksen.
DynamoDBPreviousValues	Laskee energiankulutuksen muutoksen, suorittaa arvojen aggregoinnin ja ylläpitää edellisten arvojen tilaa.	Amazon Timestream & DynamoDB	Vertailee nykyistä ja edellistä arvoa tietojen jatkuvaa seurantaa varten.

AWS Lambda on palvelimettomaan arkkitehtuuriin (serverless) perustuva pilvipalvelu, joka mahdollistaa koodin kehittämisen ja suorittamisen ilman, että kehittäjän tarvitsee huolehtia palvelimien käyttöönotosta tai hallinnasta. Lambda käyttää AWS:n hallinnoimia palvelimia, mutta kehittäjän ei tarvitse huolehtia niistä. Kustannukset muodostuvat vain käytetystä laskenta-ajasta ilman ylläpitokuluja, eli kustannuksia ei synny silloin, kun koodia ei suoriteta. Lambdan avulla voidaan suorittaa koodia lähes minkä tahansa tyyppiselle sovellukselle tai taustapalvelulle. Palvelu huolehtii automaattisesti kaikesta, mitä koodin suorittaminen ja skaalaus edellyttävät, ja tarjoaa korkean käytettävyyden. Lambda-koodi voidaan määrittää käynnistymään automaattisesti muista AWS-palveluista, kuten AWS IoT Core Rules Enginestä tai mistä tahansa verkko- tai mobiilisovelluksesta. Lambda tukee natiivisti Java-, Go-, PowerShell-, Node.js-, C#-, Python- ja Ruby-koodia ja tarjoaa suorituksenaikaisen API-rajapinnan, jonka avulla voidaan käyttää muitakin ohjelmointikieliä. (AWS Lambda FAQs 2025.)

4.6.4 Teollisuusdatan mallinnus – AWS IoT SiteWise

Käytin konseptissa teollisuuteen tarkoitettua AWS IoT SiteWise -palvelua prosessidatan mallintamiseen, tallentamiseen ja visualisoinnin datalähteenä. Palvelun avulla voidaan kerätä, tallentaa, järjestää ja visualisoida tuhansien antureiden ja laitteiden tietovirtoja useista teollisuuslaitoksista (AWS IoT SiteWise FAQs 2025). IoT SiteWise -palvelun avulla sain muodostettua konseptista UNS-periaatteiden mukaisen hierarkkisen mallin laitekohtaisine tietomalleineen, kuten kuvassa 1 on havainnollistettu.

Hyödynsin palvelua myös mittausarvojen reaaliaikaiseen ja historialliseen tarkasteluun Amazon Managed Grafana -palvelun avulla. Lisäksi IoT SiteWise toimi keskeisenä osana datan jatkokäsittelyssä ja aggregoinnissa.



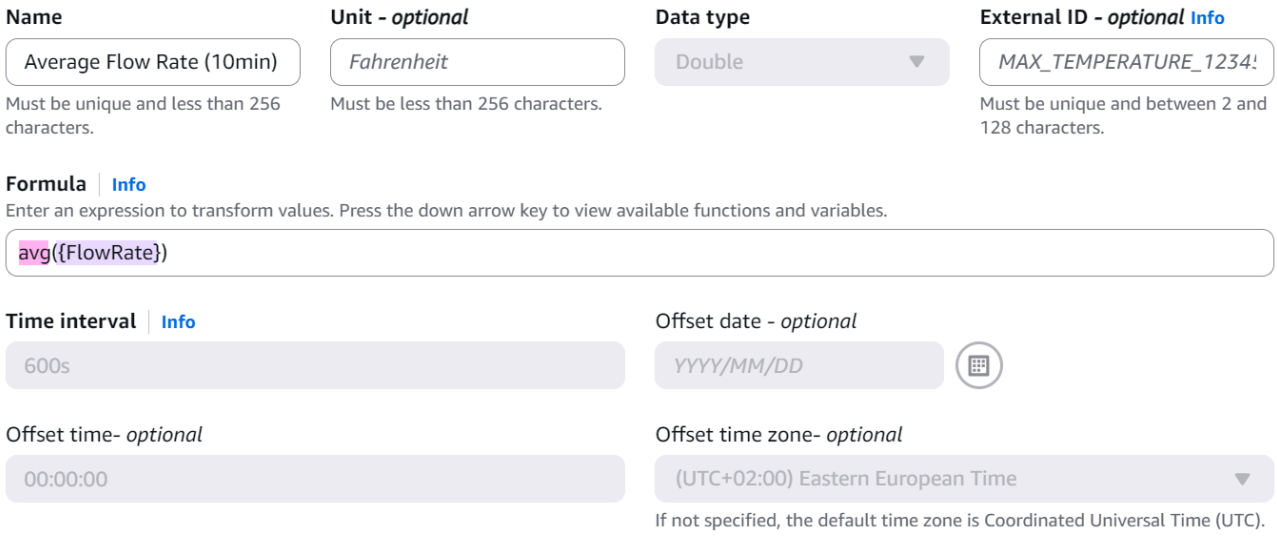
KUVA 1. AWS IoT SiteWise -palvelun käyttöliittymä, mallinnettu UNS-hierarkia.

Pilvipalvelun lisäksi AWS tarjoaa myös AWS IoT SiteWise Edge -ohjelmistoa, jota suoritetaan paikallisesti reunaympäristössä laitteessa, jossa on riittävä suorituskyky. Sitä ei kuitenkaan käytetty tässä konseptissa. Ohjelmisto mahdollistaa paikallisen datan keruun, prosessoinnin, tallennuksen, visualisoinnin sekä datan välityksen paikallisille asiakkaille tai AWS-pilveen. SiteWise Edge sisältää valmiit ominaisuudet datan keräämiseen aikasarjatietovirroista, laitteista ja tietokannoista käyttämällä teollisuuden protokollia, kuten OPC UA, Ethernet/IP ja Modbus-TCP. (AWS IoT SiteWise Edge 2025.) Pilvessä toimivan AWS IoT SiteWisen datalähteenä voivat toimia AWS IoT SiteWise Edge, AWS IoT Core tai REST API -rajapinta.

SiteWise-palvelussa jokaiselle laitteelle tai anturille luodaan digitaalinen datamalli (asset model). Näiden määrittely voidaan tehdä käyttöliittymästä ilman koodaamista tai SQL-komentoja, hyödyntäen valmiita funktioita. Datamalliin voidaan määrittää muun muassa muunnoskaavoja (transform), kuten yksikkömuunnoksia Celsius-asteista Fahrenheit-yksiköiksi tai aikaperusteisia metriikoita (metrics), kuten keskiarvon laskenta. Lisäksi voidaan määrittää hierarkkisia suhteita, esimerkiksi pumppu voidaan määrittää moottorin emomalliksi. (Create asset models in AWS IoT SiteWise 2025.) Taulukossa 8 on esitetty datamallin keskeiset attribuutit, ja kuvassa 2 on esimerkki datamallin metriikan konfiguroinnista AWS Management Consolessa.

TAULUKKO 8. AWS IoT SiteWise datamallin keskeiset attribuutit. (Create asset models in AWS IoT SiteWise 2025).

Attribuutit	**Kuvaus**	**Esimerkki**
Name	Datamallin yksilöllinen nimi.	Pump
External ID	Käyttäjän määrittämä ulkoinen tunnus, jota voidaan hyödyntää API-toiminnoissa.	Pump_1234
Measurement	Edustaa laitteistosta tulevaa tietovirtaa (esim. mittausta).	FlowRate
Transform	Kaavoja, jotka muuntavat tietovirran arvot muodosta toiseen.	{FlowRate} * 3,6
Metric	Kaavoja, jotka kokoavat tietoja aikavälein.	avg({FlowRate})
Hierarchy	Datamallien välisiä hierarkkisia suhteita.	Child Motor



KUVA 2. AWS IoT SiteWise -palvelun käyttöliittymä, metriikan konfigurointi.

4.6.5 Datantallennus ja käsittely – DynamoDB, Timestream ja S3

IoT-järjestelmissä syntyy jatkuvasti suuria määriä dataa, jota on tarpeen säilyttää eri ajanjaksojen ja käyttötarkoitusten mukaan. Pilvipalveluissa datan hallintaa jäsennetään usein kuuma- ja kylmävarasto-mallilla. Kuuma varasto on tarkoitettu usein päivittyvälle datalle, johon tarvitaan nopea kirjoitus ja välitön pääsy esimerkiksi reaaliaikaista prosessin seurantaa varten. (Hsu, Irie, Murata & Matsuoka 2018, 492–493.) Lämmin varasto, jota esimerkiksi AWS IoT SiteWise hyödyntää, soveltuu kustannustehokkaaseen historiallisen datan tallennukseen ja sen hakemiseen esimerkiksi analytiikkaa, raportointia ja koneoppimisen koulutusta varten (Manage data storage in AWS IoT SiteWise 2025). Kylmä varasto (cold storage) on kustannustehokas, mutta hitaampi ratkaisu pitkän aikavälin säilytykseen, jossa pääpaino on kapasiteetissa ja turvallisuudessa, ei nopeassa vasteajassa. Usein mitä lämpimämpi varasto on, sitä suuremmat ovat sen säilytyskustannukset. (Hsu ym. 2018, 492–494.)

AWS-palvelut soveltavat näitä varastomalleja palvelukohtaisilla ratkaisuilla. Amazon DynamoDB tarjoaa Standard-luokan, joka soveltuu usein käytettävälle datalle (hot storage), sekä Standard-IA-luokan (infrequent access), joka toimii kustannustehokkaampana vaihtoehtona harvemmin käytetylle historialliselle datalle (DynamoDB table classes 2025). Tässä konseptissa DynamoDB toimii kuumana välivarastona ja puskurointipaikkana IoT Corelta tulevalle datalle ennen eräajoa Amazon Timestream -tietokantaan. Välivaraston avulla voidaan suorittaa eräajoja Timestream-tietokantaan, mikä vähentää merkittävästi kustannuksia verrattuna siihen, että jokainen datapiste kirjoitettaisiin tietokantaan yksitellen. Amazon DynamoDB on suorituskykyinen ja joustava palvelimeton NoSQL-tietokantapalvelu, joka on helppo ja nopea ottaa käyttöön. Palvelun käyttö voidaan aloittaa pienestä, mutta tarvittaessa se skaalautuu automaattisesti horisontaalisesti lähes rajattomasti. Kustannukset määräytyvät käytön mukaan, mikä tukee kustannustehokasta käyttöä. DynamoDB soveltuu erityisesti palvelimettomiin tapahtumapohjaisiin sovelluksiin. (Amazon DynamoDB features 2025.)

Konseptin esikäsitellyn prosessidatan kuumana ja lämpimänä varastona käytin Amazon Timestream -aikasarjatietokantaa. Lisäksi käytin Timestream-tietokantaa Amazon Managed Grafanan ja Amazon QuickSight -visualisointipalvelujen datalähteenä. Nämä palvelut hakevat dataa Timestream-tietokannasta SQL-kyselyillä, jotka mahdollistavat monimutkaisemman analytiikan kuin AWS IoT SiteWise -palvelun omat ominaisuudet, ja näin palvelut täydentävät toisiaan. Tulevaisuuden jatkokehitysvaiheessa Timestreamin dataa voidaan analysoida koneoppimissovelluksissa, kuten Amazon SageMakerilla, joka soveltuu datankäsittelyyn ja SQL-pohjaiseen analytiikkaan (Amazon Timestream FAQs 2025).

Amazon Timestream hyödyntää kahta tallennustasoa: Memory store -taso on tarkoitettu lyhytaikaiselle ja nopeaa pääsyä vaativalle datalle, kun taas Magnetic store soveltuu pitkän aikavälin säilytykseen ja kustannustehokkaaseen analytiikkaan (Storage 2025). Se on nopea ja skaalautuva aikasarjadatalle optimoitu tietokantapalvelu, joka soveltuu sekä nopeisiin kyselyihin että suurten datamäärien tallennukseen. Koska kyseessä on täysin hallinnoitu palvelu, käyttäjän ei tarvitse huolehtia esimerkiksi asennuksista, päivityksistä, tallennustilasta, korkean käytettävyyden varmistamisesta tai varmuuskopioista. (Amazon Timestream 2025.) Kustannukset määräytyvät käytön mukaan (Amazon Timestream FAQs). Timestreamin yleisimmät käyttökohteet ovat IoT-, DevOps- ja analyytiikkasovellukset (Amazon Timestream 2025).

Käytin konseptissa Amazon S3 -palvelua kylmänä varastona harvemmin käytettävälle datalle, kuten reunasovelluksen konfiguraatiotiedostojen, historiadatan ja AVRO-muotoisen SiteWise-datan pitkäaikaiseen säilytykseen. Tämä mahdollistaa jatkoanalytiikan esimerkiksi AWS Glue- ja Athena -palveluissa. Amazon S3 sopii laajasti eri käyttötarkoituksiin, kuten datalake-ratkaisuihin, pilvinatiiveihin sovelluksiin, kriittisen datan varmuuskopiointiin sekä kustannustehokkaaseen pitkäaikaissäilytykseen (Amazon S3 FAQs 2025). Se on skaalautuva pilvitallennuspalvelu, joka on suunniteltu säilyttämään rajattomia määriä dataa ja mahdollistamaan sen hakemisen turvallisesti internetin yli (Amazon S3 FAQs 2025).

Pääsy Amazon S3 -resursseihin on hallittavissa tarkasti esimerkiksi IAM-poliitikoilla ja bucket-kohtaisilla säännöillä. Tiedot tallennetaan objekteina säilöihin (bucket), joissa jokaisella objektilla on yksilöllinen tunniste. (Amazon S3 FAQs 2025.) Tallennusarkkitehtuuria täydensi AWS IoT SiteWisen sisäinen kuuma–lämmin–kylmävarastomalli. Lisäksi eräajot DynamoDB:stä Timestream-tietokantaan toteutettiin AWS Lambda -funktioilla ja ajastettiin Amazon EventBridgen avulla. Näin kokonaisuus yhdistää eri tallennusmallien vahvuudet: kuuma varasto mahdollistaa nopean käytön, lämmin varasto tehokkaan analytiikan ja kylmä varasto kustannustehokkaan pitkäaikaissäilytyksen (Hsu ym. 2018, 492–493).

4.6.6 Turvallisuus ja käyttäjähallinta

Turvallisuus ja käyttäjähallinta ovat olennainen osa pilviarkkitehtuuria, sillä datan ja pilvipalvelujen käyttö perustuu tunnistamiseen, pääsynhallintaan ja salaisuuksien suojaamiseen. Tässä konseptissa painopiste on käyttöoikeuksien hallinnassa AWS Identity and Access Management (IAM) -palvelun

avulla sekä tunnistetietojen turvallisessa säilytyksessä AWS Secrets Manager -palvelun avulla. Lisäksi IAM Identity Center -palvelua käytettiin käyttäjähallinnan ja kirjautumisprosessin hallintaan. Näistä palveluista IAM on keskeisin, sillä sen avulla hallitaan käyttöoikeuksia AWS-resursseihin määrittelemällä käyttäjiä, rooleja ja poliitikkoja (AWS Identity and Access Management (IAM) FAQs 2025).

IAM on keskeinen osa tämän konseptin pilviarkkitehtuuria, sillä AWS:n kaikki resurssit ovat oletuksena estettyjä, ja käyttöoikeudet on määritettävä täsmällisesti käyttötapauksen mukaan. AWS suosittelee käyttöoikeuksien rakentamista vähiten sallittujen oikeuksien periaatteella (least privilege). Testausvaiheessa voidaan aloittaa laajemmilla oikeuksilla, mutta lopullista käyttöä varten myönnetään vain välttämättömät oikeudet. AWS-pilvipalveluissa käyttöoikeuksia hallitaan kolmella keskeisellä käsitteellä: käyttäjä (user) on yksittäinen IAM-identiteetti, joka on yleensä henkilö tai palvelu; rooli (role) on joukko käyttöoikeuksia AWS-palvelupyyntöjen tekemiseen eikä ole sidottu tiettyyn käyttäjään tai ryhmään; politiikka (policy) on JSON-muotoinen dokumentti, joka määrittelee mitä toimintoja missäkin resursseissa saa suorittaa. (AWS Identity and Access Management (IAM) FAQs 2025.)

Käytin IAM Identity Center -palvelua konseptin Amazon Managed Grafanan käyttäjähallinnan ja kirjautumisen hallintaan. Identity Center mahdollistaa keskitetyn pääsynhallinnan AWS-resursseille ja tukee kolmannen osapuolen sovelluksia, kuten Grafanaa (AWS IAM Identity Center FAQs 2025). Alla on esimerkki AWS:n hallinnoimasta *AWSLambdaBasicExecutionRole*-politiikasta JSON-muodossa (KOODI 1), joka mahdollistaa Lambda-funktion kirjoittaa lokit AWS CloudWatch-lokeihin. Tämä on tyypillinen käyttöoikeus Lambda-funktion perustoiminnallisuuteen testausvaiheessa. Tuotantoympäristössä Resource kannattaa rajata tarkemmin kuin ”*”, joka sallii käytännössä kaikki tilin resurssit.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

KOODI 1. IAM-roolin esimerkkipolitiikka CloudWatch-lokitukseen.

Käyttöoikeuksien lisäksi salaisuuksien suojaus on olennainen osa kokonaisarkkitehtuuria. Käytin konseptissa AWS Secrets Manageria paikallisesti IOT2050-yhdyskäytävälaitteella AWS Greengrassin kautta, jotta OPC UA -palvelimeen voidaan tunnistautua ilman, että tunnistetietoja koodataan ohjelmaan tai säilytetään suojaamattomasti tiedostojärjestelmässä. Secrets Manager on salaisuuksien hallintapalvelu, joka auttaa suojaamaan sovellusten, palveluiden ja IT-resurssien käyttöoikeuksia. Sen avulla voidaan hallita ja kierrättää salaisuuksia, joita käytetään resurssien käyttämiseen AWS-pilvessä, kolmannen osapuolen palveluissa ja paikallisesti. Palvelu mahdollistaa tietokannan tunnistetietojen, API-avainten ja muiden salaisuuksien turvallisen tallennuksen ja hallinnan koko niiden elinkaaren ajan. (AWS Secrets Manager FAQs 2025.)

Hyödynsin toteutuksessa kahta erillistä salaisuutta: (1) käyttäjätunnus ja salasana, joita käytetään OPC UA -yhteyden käyttäjätunnus–salasana-autentikointiin (Sign & Encrypt). Nämä haetaan ohjelmallisesti Greengrass-komponentissa ja liitetään OPC UA -asiakkaan kirjautumispyyntöön. (2) Keystore-salasana, jota käytetään yksityisen avaimen purkamiseen ja lataamiseen OPC UA -asiakasohjelmassa. Tämä mahdollistaa turvallisen X.509-pohjaisen autentikoinnin ilman, että salasana tallennetaan suoraan laitteen tiedostojärjestelmään. Yhdistämällä IAM Identity Centerin (pääsynhallinta) ja Secrets Managerin (salaisuuksien hallinta) toteutus noudattaa least privilege -periaatetta, välttää kovakoodatut tunnukset ja mahdollistaa hallitun käytön myös reunalaitteessa.

4.6.7 Valvonta ja lokitus

Pilviarkkitehtuurin toimivuuden kannalta pelkkä käyttöoikeuksien hallinta ja salaisuuksien suojaaminen eivät riitä, vaan tarvitaan myös jatkuvaa valvontaa ja lokitietojen seurantaa. AWS:n tarjoama CloudWatch-palvelu mahdollistaa sekä sovellusten että infrastruktuurin tilan tarkkailun ja virheiden analysoinnin reaaliaikaisesti (Amazon CloudWatch FAQs 2025). Tässä konseptissa käytin CloudWatchia AWS Lambda -funktioiden tapahtumien lokitukseen ja virheiden analysointiin. Jokainen Lambda-funktion suoritus kirjoittaa lokit automaattisesti CloudWatch-palveluun, mikä mahdollistaa muun muassa virheviestien, poikkeusten ja käsitellyn datan tarkastelun jälkikäteen. CloudWatch-lokit ovat olennainen osa konseptin Lambda-funktioiden valvontaa ja vianhakua. Amazon CloudWatch on AWS:n hallinnoima valvontapalvelu, jonka avulla voidaan kerätä ja seurata mittareita, tarkastella lokitiedostoja sekä asettaa hälytyksiä järjestelmien ja sovellusten tilasta (Amazon CloudWatch FAQs 2025). Palvelu soveltuu erityisesti tapahtumapohjaisen arkkitehtuurin toiminnan tarkkailuun ja vikatilanteiden selvittämiseen.

4.6.8 Data-analyysi ja datan visualisointi AWS-ympäristössä

Kuten luvussa 2.6 käsiteltiin, data-analyysi ja visualisointi ovat olennainen osa IIoT-ratkaisua, sillä niiden avulla prosessidataa voidaan seurata, tulkita ja hyödyntää päätöksenteossa. Konseptissa näiden käytännön toteutus perustui AWS:n tarjoamiin visualisointi- ja analysointipalveluihin. Käytin Amazon Managed Grafanaa ja Amazon QuickSightia, jotka tarjoavat toisiaan täydentävät lähestymistavat: Grafana mahdollistaa reaaliaikaisen seurannan ja teknisen analyysin, kun taas QuickSight tukee raportointia ja KPI-laskentaa esimerkiksi energiankulutuksen tarkastelussa. Näin muodostui kaksitasoinen kokonaisuus, jossa Grafana palvelee operatiivista hallintaa ja QuickSight strategista päätöksentekoa.

Hyödynsin konseptissa Amazon Managed Grafanaa prosessin reaaliaikaiseen seurantaan ja tekniseen analyysiin. Rakensin toteutuksessa hallintanäkymiä (dashboards), joilla esitettiin pumppausjärjestelmän yleisnäkymä sekä reaaliaikaiset mittaukset esimerkiksi moottorin tehosta (kW), virrasta (A), lämpötilasta (°C) ja värähtelystä (LIITE 9). Molemmat näkymät on esitetty liitteessä 9. Datan lähteinä käytin Amazon Timestreamia ja AWS IoT SiteWisea, joista kerätty prosessidata tuotiin Grafanaan lähes viiveettömästi. Timestreamin kohdalla käytin SQL-kyselyitä (LIITE 10), kun taas SiteWisen kohdalla riitti konfiguroida viittaamaan haluttuun datapisteeseen. Näkymien avulla voitiin seurata järjestelmän tilaa ja havaita poikkeamat, kuten äkilliset kuormituksen muutokset tai epänormaali käyttäytyminen moottoreissa. Grafanan rooli oli ensisijaisesti operatiivinen, tarjoten työkalun prosessin tilan jatkuvaan valvontaan ja häiriöiden nopeaan tunnistamiseen. Palvelun etuna oli lähes reaaliaikainen näkymä, joka oli kustannustehokas ja soveltui hyvin jatkuvaan seurantaan. Käytännön toteutusta tuki Amazon Managed Grafanan tarjoama palvelukokonaisuus, joka kuvataan seuraavassa kappaleessa.

Amazon Managed Grafana on AWS:n hallinnoima palvelu, joka soveltuu usean pilviympäristön ja projektin käyttöön. Se sisältää monipuoliset ja interaktiiviset visualisointimahdollisuudet, joiden avulla voidaan analysoida, valvoa ja muodostaa hälytyksiä metriikoista, lokitiedoista ja jäljistä eri tietolähteiden yli. Palvelu perustuu avoimen lähdekoodin Grafanaan, joka on datavisualisointiin sekä toiminnan seurantaan ja hallintaan tarkoitettu ratkaisu. Grafanaa käyttävät sadat tuhannet organisaatiot ja miljoonat käyttäjät maailmanlaajuisesti. Sen vahvuuksia ovat laaja visualisointikirjasto ja tuki useille tietolähteille, mikä mahdollistaa erilaisten operatiivisten tietojen, kuten metriikoiden, lokien ja jälkien tarkastelun sekä hälytysten muodostamisen yhdessä konsolissa hallintanäkymien avulla. Amazon Managed Grafana tarjoaa täysin hallittuja Grafana-työtiloja, jotka ovat yhteensopivia avoimen lähdekoodin version kanssa ja kehitetty yhteistyössä Grafana Labsin, avoimen lähdekoodin projektin emoyhtiön, kanssa. (Amazon Managed Grafana FAQs 2025.)

Amazon Managed Grafana soveltui paremmin operatiiviseen reaaliaikaiseen seurantaan ja tekniseen analyysiin, kun taas toteuttamani Amazon QuickSight -ratkaisu tarjosi luontevamman ympäristön liiketoimintaraportointiin ja BI-näkymiin. Näin palvelut täydensivät toisiaan eri käyttötarkoituksissa. Amazon QuickSight -toteutus rajattiin energiankulutuksen laskentaan ja raportointiin (LIITE 11). Palvelussa käytettiin datalähteenä Amazon Timestreamia, josta prosessidata haettiin SQL-kyselyillä (LIITE 10). Tulokset ladattiin palvelun Super-fast, Parallel, In-memory Calculation Engine (SPICE) -moottoriin, mikä mahdollisti aggregoinnin ja raportoinnin tehokkaasti ilman jatkuvia kyselyjä lähdejärjestelmään. Tämän ansiosta dataa voitiin käsitellä eri aikatasoilla, kuten tunneittain, päivittäin ja viikoittain, mikä mahdollisti trendien tunnistamisen ja muutosten seuraamisen.

Amazon QuickSight on AWS:n hallinnoima palvelu, joka tarjoaa datalähtöisille organisaatioille yhtenäisen liiketoimintatiedon (BI) ratkaisun laajassa mittakaavassa. Palvelun avulla voidaan toteuttaa pikselintarkkoja raportteja ja interaktiivisia hallintanäkymiä (dashboards), joiden taustalla analytiikka rakennetaan suoraan tietolähteistä, kuten Amazon Timestreamista. QuickSight sisältää myös Amazon Q -ominaisuuden, jonka avulla liiketoiminta-analyytikot ja käyttäjät voivat hyödyntää luonnollisen kielen kyselyitä (Natural Language Querying, NLQ) merkityksellisten havaintojen tekemiseen, löytämiseen ja jakamiseen sekunneissa. QuickSight on palvelimeton ja automaattisesti skaalautuva ratkaisu, joka mukautuu jopa kymmeniin tuhansiin käyttäjiin ilman erillistä infrastruktuurin hallintaa tai kapasiteetin suunnittelua. Palvelulla on lisäksi laaja käyttäjäyhteisö ja yli 100 000 asiakasta maailmanlaajuisesti. (Amazon QuickSight 2025.)

Rakennetuissa hallintanäkymissä esitettiin esimerkiksi kokonaisenergiankulutus sekä muutosprosentit tunnin ja vuorokauden tasolla (LIITE 11). Näiden avulla saatiin selkeä kuva prosessin energiankäytön kehityksestä ja sen vaihteluista eri ajanjaksoilla. QuickSightin rooli oli ensisijaisesti strateginen, sillä sen avulla tuotettiin KPI-laskentoja ja raportteja, joita voidaan hyödyntää todellisessa ympäristössä johdon päätöksenteossa ja energiankäytön optimoinnissa. Palvelun etuina olivat monipuoliset laskentafunktiot sekä raporttien helppo jakaminen eri sidosryhmille. Kustannustehokkuuden vuoksi QuickSightia ei käytetty sekuntitason reaaliaikaiseen seurantaan, sillä Grafana tarjosi tähän tarkoitukseen paremmin soveltuvan ratkaisun.

5 ARVIOINTI JA POHDINTA

Tässä luvussa arvioidaan, miten opinnäytetyön toteutus onnistui suhteessa asetettuihin tavoitteisiin sekä luvuissa 2 ja 3 käsiteltyyn teoriaperustaan. Pohdin projektin aikana saamiani kokemuksia ja havaintoja sekä sitä, mitä opin työn edetessä. Arvioinnissa tarkastellaan sekä työn vahvuuksia että sen rajoitteita ja haasteita, jotka vaikuttivat lopputulokseen. Lopuksi esittelen jatkokehitysideoita, joiden avulla vastaavia IIoT-ratkaisuja voidaan kehittää tulevaisuudessa.

5.1 Omat havainnot ja kokemukset

Opinnäytetyön aikana opin laajasti uusia taitoja, jotka liittyivät pilvipalveluihin, ohjelmistokehitykseen ja teollisissa IoT-ratkaisussa käytettyihin viestintäprotokolliin. Erityisesti AWS-pilvipalveluiden arkkitehtuurin suunnittelu ja käyttöönotto olivat keskeisiä oppimiskohteita. Opinnäytetyön aikana syvensin osaamistani AWS:n pilvipalveluista myös suorittamalla AWS:n itseopiskelukursseja, kuten AWS Educate ja Skill Builder. Kurssit ja AWS:n virallinen dokumentaatio antoivat perustason ymmärryksen palveluista ja tukivat käytännön toteutusta, jossa pääsin soveltamaan oppimaani käytännössä. Tämä yhdistelmä teoriaa ja käytännön harjoittelua vahvisti oppimisprosessia ja auttoi kokonaisuuden hallinnassa.

Opin käyttämään AWS Greengrass -ympäristöä reuna–pilvi-integraation mahdollistajana jonka avulla reunalla oleva yhdyskäytävälaite integroitiin osaksi pilvipalveluita. Lisäksi sain käytännön kokemusta monista muista AWS:n palveluista, kuten Lambdasta, Timestreamista, DynamoDB:stä, SiteWisesta, Grafanasta ja QuickSightista. Työ kehitti myös ohjelmointiosaamistani. Kehitin teolliseen IoT-tarkoitukseen Java-ohjelman, joissa hyödynsin avoimen lähdekoodin kirjastoja. Samalla opin käyttämään Linux-käyttöjärjestelmää yhdyskäytävälaitteen roolissa sekä ymmärtämään paremmin keskeisiä teollisuudessa käytettyjä IoT-protokollia ja erilaisia IoT-arkkitehtuurimalleja. Teknisten taitojen lisäksi opin prosessin aikana tutkimuksellista näkökantaa ja tutkitun tiedon hyödyntämistä käytännön toteutuksen ohella. Tutkitun teorian ja oman toiminnallisen toteutuksen kautta ymmärsin, että IoT ja sen soveltamiskohteet, kuten teollinen IoT ovat hyvin monitasoisia ja laajoja kokonaisuuksia, joista saisi useita tutkimus- ja kehityskohteita.

Projektin aikana tein myös useita käytännön havaintoja, jotka liittyivät erityisesti kyberturvallisuuteen ja resurssien hallintaan. Kyberturvallisuuden parantaminen osoittautui monitasoiseksi kokonaisuudeksi, joka kattoi niin ohjelmakoodin, laitteistoratkaisut, konfiguroinnin, sertifikaatit kuin autentikoinninkin. Yksi konkreettinen esimerkki oli OPC UA -kommunikoinnin suojaaminen, jossa huomasin, että vaikka TIA Portalissa luotu sertifikaatti oli Eclipse Milon tukemassa formaatissa, kuitenkin kryptografinen toteutus oli väärässä muodossa. TIA Portalissa luodut sertifikaatit olivat PKCS#1-standardin muodossa, kun taas Eclipse Milo -kirjasto edellytti PKCS#8-muotoa. Tämä tarkoitti, että sertifikaatti täytyi vielä muuntaa Eclipsen Milon tukemaan PKCS#8-muotoon, jotta sain kommunikoinnin toimimaan. Havainto antoi arvokasta käytännön oppia sertifikaattien hallinnasta ja standardien eroista.

Toinen tärkeä havainto liittyi resurssien ja kustannusten arviointiin pilviarkkitehtuurin suunnittelussa. Työn edetessä ymmärsin, kuinka tärkeää on ottaa huomioon paitsi järjestelmän tekninen toimivuus myös sen kustannustehokkuus. AWS tarjoaa monipuolisia vaihtoehtoja eri tarpeisiin, ja järjestelmän arkkitehti voi painottaa valinnoissa esimerkiksi joko suurten datavirtojen tehokasta käsittelyä tai kustannustehokkuutta, mikä voi heikentää suorituskykyä. AWS:n monipuolinen palveluvalikoima mahdollistaa arkkitehtuurin optimoinnin eri tavoitteiden mukaisesti. Tämä monipuolisuus osoitti, että sama IIoT-järjestelmä voidaan toteuttaa useilla eri tavoilla riippuen tavoitteista ja käytettävissä olevista resursseista. Kolmas havainto liittyi reunalla toimivan yhdyskäytävälaitteen resurssien rajoituksiin. Vaikka laitteessa oli suhteellisen paljon laskentatehoa kokoon ja hankintakustannuksiin nähden, prosessori ei tue tehokkaasti monisäikeisen ohjelman suorittamista. Laitteen rajallinen prosessoriteho, RAM-muisti ja tallennustilan vähäisyys eivät tue laitteen käyttöä suurempien reuna-arkkitehtuurien keskitettynä MQTT-välityspalvelimena tai datan paikallisena varastona.

Havainnot tukivat myös luvuissa 2.3 ja 2.4 käsiteltyä teoreettista taustaa, jossa IIoT-arkkitehtuureissa keskeiseksi haasteeksi tunnistetaan kustannusten, resurssien ja suorituskyvyn tasapainottaminen. Työn toteutuksessa tehdyt havainnot vahvistivat käytännössä sen, että tämä tasapaino on ratkaiseva tekijä järjestelmän suunnittelussa ja toteuttamisessa. Lisäksi havaitsin, että kyberturvallisuuden varmistaminen kaikilla tasoilla vastasi luvussa 2.5 esitettyä näkemystä IIoT:n monikerroksisista suojaustarpeista. Koen, että havaintojeni luotettavuutta tukee se, että testasin ratkaisuja toistuvasti ja dokumentoin tulokset opinnäytetyöhön. Vertailin havaintoja AWS:n viralliseen dokumentaatioon sekä Siemensin tuotteisiin liittyviin ohjeisiin.

Eettisestä näkökulmasta työssä ei käsitelty arkaluonteista dataa, ja tietoturvakokeilut toteutettiin turvallisesti rajatussa testausympäristössä. Kokonaisuutena havaitsin, että opinnäytetyön aikana sain yhdistettyä automaatiotaustani ja uudet pilvipalveluihin liittyvät taidot yhdeksi kokonaisuudeksi. Tämä kokemus antoi varmuutta siitä, että pystyn jatkossa suunnittelemaan ja toteuttamaan laajoja IIoT-ratkaisuja, joissa yhdistyvät sekä reuna- että pilviteknologiat. Kirjallisessa työssä opin jäsentämään laajaa teknistä kokonaisuutta ja arvioimaan kriittisesti sekä lähteiden käyttöä että oman tekstin selkeyttä ja johdonmukaisuutta.

5.2 Työn onnistuminen ja haasteet

Työn tavoitteena oli kehittää kustannustehokas teollisen IoT-järjestelmän konsepti, jonka avulla voidaan madaltaa siirtymisen kynnystä Teollisuus 4.0 -ratkaisuihin erityisesti pienemmissä yrityksissä. Tavoitteen saavuttamiseksi valitsin AWS:n pilvipalveluntarjoajaksi, sillä sen tarjoamat palvelut tukevat IIoT-arkkitehtuurin toteuttamista ja soveltuvat erityisesti UNS-lähestymistavan hyödyntämiseen. Tässä työssä ei ollut keskeistä millisekuntien vasteaika, vaan joustavuus, skaalautuvuus ja kustannusten hallinta, joissa AWS soveltui parhaiten tavoitteisiin. Avoimen lähdekoodin kirjastot ja AWS:n pilvipalvelut mahdollistivat arkkitehtuurin toteuttamisen ilman suuria alkuinvestointeja tai ylläpitokuluja. Lisäksi työn tulokset osoittivat, että järjestelmä on skaalattavissa tarpeen mukaan. Käyttöönotto voidaan aloittaa pienemässä mittakaavassa ja laajentaa myöhemmin suurempaan. Tämä vastasi hyvin tavoitteeseen madaltaa kynnystä siirtyä hyödyntämään Teollisuus 4.0 -ratkaisuja. Kehitetty konseptijärjestelmä osoitti, että yhdistämällä edullinen yhdyskäytävälaite ja AWS:n pilvipalvelut voidaan rakentaa toimiva ja kustannustehokas IIoT-ratkaisu.

Työn aikana kohdatut haasteet vastasivat tietoperustassa tunnistettuihin haasteisiin, kuten teknologian monimutkaisuus, resurssien rajallisuus ja kustannusten hallinta. Teknologian monimutkaisuus ilmeni esimerkiksi AWS IoT Greengrass -ympäristön käyttöönotossa, joka koostui useasta vaiheesta, kuten laiteympäristön asentamisesta, AWS-resurssien luomisesta, komponenttien määrittelystä ja IAM-oikeuksien hallinnasta. Monimutkaisuutta lisäsi myös OPC UA -sertifikaattien yhteensopivuusongelmat. Resurssirajoitteet korostuivat esimerkiksi yhdyskäytävän rajallisessa prosessoritehossa, muistissa ja tallennustilassa, jotka estäisivät laitteen käytön suuremmassa UNS-arkkitehtuurissa keskitettynä viestinvälityspalvelimena tai historian tallennusalustana. Lisäksi vaikka konsepti toteutettiin kustannustehokkaasti AWS:n hinnoittelumalli puolestaan paljasti, että pitkäaikainen käyttö edellyttää huolellista suunnittelua ja optimointia.

Työn tulokset konkretisoivat teoriassa kuvatut ongelmat käytännön toteutuksen tasolla. Luotettavuutta vahvistaa se, että ratkaisut testattiin useaan kertaan ja järjestelmää ajettiin useamman viikon ajan. Seurasin aika ajoin lokeja, palveluiden toimintaa ja datan päivittymistä sekä AWS-palveluiden kustannuksia, jotta pystyin varmistamaan ratkaisun toimivuuden ja arvioimaan sen taloudellista kestävyyttä. Työssä ei käsitelty arkaluonteista dataa, ja kaikki testaus toteutettiin rajatussa testiympäristössä. Haasteista huolimatta voidaan todeta, että työn tavoitteet saavutettiin erinomaisesti. Toteutus osoitti, että kustannustehokas ja skaalautuva IIoT-järjestelmä on mahdollista rakentaa pienin resurssein, ja että tällaiset konseptit voivat toimia tärkeänä välineenä yrityksille, jotka haluavat ottaa käyttöön Teollisuus 4.0 -ratkaisuja hallitusti ja vaiheittain.

5.3 Jatkokehitysideat

Työssä kehitetty konseptijärjestelmä osoitti, että kustannustehokas IIoT-ratkaisu on mahdollinen myös rajallisilla resursseilla. Jatkossa järjestelmää olisi kuitenkin mahdollista laajentaa ja kehittää useilla eri tavoilla. Kuten luvussa 2.4 todettiin, IIoT:n tuleviin suuntaviivoihin liittyy erityisesti resurssirajoitteiden, monimutkaisuuden ja skaalautuvuuden hallinta. Nämä samat teemat heijastuivat myös tämän työn toteutuksessa ja muodostavat luonnollisia jatkokehityskohteita. Ensimmäinen jatkokehityskohde liittyy kyberturvallisuuden syventämiseen. Vaikka konseptissa käytettiin vahvaa salausprofiilia ja sertifikaattipohjaista autentikointia, laitetason tietoturvaa voitaisiin vahvistaa edelleen esimerkiksi TPM- tai HSM-ratkaisuilla. Lisäksi järjestelmään olisi mahdollista liittää auditointimekanismeja, joiden avulla voitaisiin havaita poikkeava toiminta aikaisessa vaiheessa ja siten parantaa järjestelmän kokonaisvaltaista turvallisuutta.

Laitteistovalinta on aina tasapainottelua ympäristövaatimusten, suorituskykytarpeiden ja sovelluksen kriittisyyden välillä. Toinen jatkokehityskohde liittyy yhdyskäytävälaitteen ja viestinvälityksen arkkitehtuuriin. SIMATIC IOT2050 ja avoimen lähdekoodin Mosquitto-välityspalvelin soveltuvat hyvin konseptitason ja pienemmän arkkitehtuurin toteutukseen, mutta näiden rajalliset resurssit estävät käytön keskitettynä välityspalvelimena suuremmissa arkkitehtuureissa. Jatkossa voisi olla perusteltua arvioida vaihtoehtoisten yhdyskäytävälaitteiden käyttöä tai siirtymistä kehittyneempiin MQTT-välityspalvelin-ratkaisuihin. Suuremmissa arkkitehtuureissa on mahdollista tuoda reunalle tehokkaampi PC-pohjainen ratkaisu, jossa ajetaan keskitetysti kehittyneempää MQTT-välityspalvelinta, ja SIMATIC IOT2050 -yhdyskäytävälaitteet toimivat sille julkaisijoina sekä tilaajina. Tällainen ratkaisu mahdollis-

taisi myös paikallisen datan tallentamisen, jolloin kriittinen prosessidata voitaisiin säilyttää ja analysoida ilman jatkuvaa pilviyhteyttä. Tämä parantaisi järjestelmän skaalautuvuutta, luotettavuutta ja suorituskykyä laajemmissa käyttökohteissa.

Kolmas kehityssuunta liittyy datan hyödyntämiseen. Pilvipalveluissa voitaisiin tulevaisuudessa hyödyntää koneoppimista ja tekoälyä datan analysointiin esimerkiksi energiankulutuksen optimointiin tai ennakoivaan kunnossapitoon. Samalla tekoälyä voisi tuoda myös reunalle, jolloin nopea päätöksenteko voitaisiin toteuttaa lähellä prosessia ilman viivettä. Tämä parantaisi järjestelmän reagointikykyä ja vähentäisi riippuvuutta pilviyhteydestä. Lisäksi järjestelmän arkkitehtuuri olisi mahdollista kehittää kohti täysimittaista UNS-toteutusta. Tämä tarkoittaisi hierarkkisen datamallin laajentamista koko organisaation kattavaksi sekä Sparkplug B -protokollan käyttöönottoa. Vaikka Sparkplug B tarjoaa merkittäviä etuja, päädyin tämän työn toteutuksessa klassiseen MQTT-protokollaan sen joustavuuden sekä AWS-integraatioiden ja selkeästi luettavien viestien vuoksi. Nämä helpottivat käyttöönottoa ja testausta. Sparkplug B toisi kuitenkin vahvemman yhteyden teollisuuden standardeihin ja mahdollistaisi tiedonvaihdon standardoidulla viestirakenteella eri järjestelmien ja toimijoiden välillä.

Lisäksi työn aikana havaittiin, että IIoT-ratkaisujen toteutus ei voi nojata pelkästään tekniseen onnistumiseen. IEC 62443, IISF ja NIST tarjoavat riskiperustaisen, kerroksittain jäsennellyn viitekehyksen IIoT-turvallisuudelle, kun taas NIS2 ja datasäädös tuovat velvoittavan tason riskienhallintaan, poikkeamien käsittelyyn, toimitusketjun turvaan ja datan käyttöoikeuksiin. EU:n kyberkestävyyssäädös (CRA) asettaa markkinoille saatettavien tuotteiden valmistajille vaatimuksia siitä, että digitaalisia elementtejä sisältävät tuotteet, kuten ohjelmistot, laitteistot ja niihin liittyvät pilvipalvelut, suunnitellaan, kehitetään ja ylläpidetään koko elinkaaren ajan kyberturvallisuus huomioon ottaen. Tämä sisältää muun muassa olennaisten kyberturvallisuusvaatimusten täyttämistä, vaatimustenmukaisuuden arviointia sekä CE-merkinnän kiinnittämistä tuotteeseen ennen markkinoille saattamista. Käytännön tasolla nämä merkitsevät sitä, että IIoT-tuotteiden ja ratkaisujen suunnittelu ja operointi on vietävä standardienmukaisiksi ja sääntelyn mukaisiksi, minkä lisäksi tarvitaan arkkitehtuurin jokaisessa kerroksessa toteutettavia teknisiä ratkaisuja, joilla vaatimuksiin vastataan todellisessa ympäristössä.

6 YHTEENVETO JA JOHTOPÄÄTÖKSET

Opinnäytetyössä kehitettiin konseptitason teollinen IoT-järjestelmä, jonka avulla voidaan madaltaa siirtymisen kynnystä Teollisuus 4.0 -ratkaisuihin erityisesti pienemmissä yrityksissä. Työssä toteutettu konseptijärjestelmä osoitti, että yhdistämällä edullinen yhdyskäytävälaite, kuten SIMATIC IOT2050 ja AWS:n pilvipalvelut, voidaan rakentaa toimiva ja skaalautuva ratkaisu ilman suuria alkuinvestointeja. Ratkaisu perustui avoimen lähdekoodin kirjastoihin ja AWS:n palveluihin, joiden avulla voitiin yhdistää reuna- ja pilviteknologiat yhtenäiseksi kokonaisuudeksi. Tulokset vahvistivat, että järjestelmää voidaan hyödyntää aluksi pienessä mittakaavassa ja laajentaa tarpeiden kasvaessa, mikä vastasi asetettuihin tavoitteisiin.

Työn aikana opittiin teknisiä taitoja, kuten pilviarkkitehtuurin suunnittelua ja käyttöönottoa AWS-pilvipalveluilla, ohjelmistokehitystä avoimen lähdekoodin kirjastoilla sekä IIoT-protokollien hyödyntämistä. Lisäksi syvennettiin ymmärrystä IoT:sta ilmiönä ja teollisessa kontekstissa sekä sovellettiin teoriaa käytännön toteutukseen. Keskeinen oppi oli, että kustannustehokkuuden, resurssien rajallisuuden ja skaalautuvuuden huomioon ottaminen muodostavat toisiaan täydentävän kokonaisuuden, joka ohjaa IIoT-arkkitehtuurien suunnittelua. Vaikka laitteiston ja palveluiden rajoitteet asettavat haasteita, ne voidaan hallita huolellisella suunnittelulla ja oikeilla palveluvalinnoilla. Konseptimalli on sovellettavissa myös muihin yrityksiin, jotka haluavat aloittaa digitaalisen muutoksen pienimuotoisesti. Malli tarjoaa käytännön esimerkin siitä, miten teollisuuden yritykset voivat yhdistää perinteisen automaation ja pilvipalvelut hallitusti sekä hyödyntää AWS:n palveluita kustannustehokkaalla tavalla.

Työn välittömät vaikutukset näkyvät ennen kaikkea siinä, että toimeksiantaja sai konkreettisen esimerkin kustannustehokkaasta teollisesta IoT-järjestelmästä. Laajemmin työ voi toimia pohjana jatkokehitykselle ja innoittaa uusiin sovelluksiin. Konsepti tarjoaa toimeksiantajalle lähtökohdan, jonka pohjalta voidaan arvioida ja kehittää omia ratkaisuja kohti Teollisuus 4.0 -ratkaisujen hyödyntämistä. Työ osoittaa selvästi, että myös rajallisilla resursseilla on mahdollista rakentaa toimiva ja skaalautuva järjestelmä. Samalla se tuottaa esimerkin, jota voidaan soveltaa muihinkin yrityksiin, jotka haluavat kokeilla IIoT-ratkaisujen käyttöönottoa pienessä mittakaavassa ennen laajempia investointeja. Työ tarjoaa mallin, joka voi toimia taustamateriaalina sekä teknisessä että taloudellisessa arvioinnissa. Lisäksi opinnäytetyö vahvistaa käsitystä siitä, että avoimen lähdekoodin komponenttien, edullisten laitteiden ja pilvipalveluiden yhdistäminen voi madaltaa kynnystä siirtyä Teollisuus 4.0 -ratkaisuihin vaiheittain ja hallitusti.

LÄHTEET

Abdulkareem, N. M., Zeebaree, S. R. M., Sadeeq, M. A. M., Ahmed, D. M., Sami, A. S. & Zebari, R. R. 2021. IoT and Cloud Computing Issues, Challenges and Opportunities: A Review. *Qubahan Academic Journal*, 1(2), 1–7. Saatavissa: https://doi.org/10.48161/qaj.v1n2a36. Viitattu 23.8.2025.

Ahmed, S. F., Bin Alam, M. S., Hoque, M., Lameesa, A., Afrin, S., Farah, T., Kabir, M., Shafiullah, G. M. & Muyeen, S. M. 2023. Industrial Internet of Things enabled technologies, challenges, and future directions. *Computers and Electrical Engineering*, 110, 108847. Elsevier. Saatavissa: https://doi.org/10.1016/j.compeleceng.2023.108847. Viitattu 23.6.2025.

Al-Masri, E., Kalyanam, K. R., Batts, J., Kim, J., Singh, S., Vo, T. & Yan, C. 2020. Investigating messaging protocols for the Internet of Things (IoT). *IEEE Access*, 8, 94880–94911. Saatavissa: https://doi.org/10.1109/ACCESS.2020.2993363. Viitattu 12.8.2025.

Amazon CloudWatch FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/cloudwatch/faqs/. Viitattu 6.5.2025.

Amazon DynamoDB features. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/dynamodb/features/. Viitattu 24.4.2025.

Amazon EventBridge. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/eventbridge/. Viitattu 6.5.2025.

Amazon Managed Grafana FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/grafana/faqs/?nc=sn&loc=5. Viitattu 6.5.2025.

Amazon QuickSight. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/quicksight/. Viitattu 18.9.2025.

Amazon S3 FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/s3/faqs/?nc=sn&loc=7#topic-0. Viitattu 4.5.2025.

Amazon Timestream. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/timestream/. Viitattu 4.5.2025.

Amazon Timestream FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/timestream/faqs/?nc=sn&loc=5. Viitattu 4.5.2025.

Anitha, T., Manimurugan, S., Sridhar, S., Mathupriya, S. & Charlyn Pushpa Latha, G. 2022. A review on communication protocols of Industrial Internet of Things. *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, 25–27.1.2022, FCIT/UT/KSA. Saatavissa: https://doi.org/10.1109/ICCIT52419.2022.9711544. Viitattu 12.8.2025.

Aouedi, O., Vu, T.-H., Sacco, A., Nguyen, D. C., Piamrat, K., Marchetto, G. & Pham, Q.-V. 2025. A Survey on Intelligent Internet of Things: Applications, Security, Privacy, and Future Directions. *IEEE Communications Surveys & Tutorials*, 27(2), 1238–1292. Saatavissa: https://doi.org/10.1109/COMST.2024.3430368. Viitattu 25.6.2025.

Ashton, K. 2009. That ‘Internet of Things’ Thing. *RFID Journal*. Saatavissa: https://www.rfidjournal.com/expert-views/that-internet-of-things-thing/73881. Viitattu 20.6.2025.

Attaran, S., Attaran, M. & Celik, B. G. 2024. Digital twins and Industrial Internet of Things: Uncovering operational intelligence in industry 4.0. *Decision Analytics Journal*, 10, 100398. Saatavissa: https://doi.org/10.1016/j.dajour.2024.100398. Viitattu 8.6.2025.

AWS IAM Identity Center FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/iam/identity-center/faqs/. Viitattu 6.5.2025.

AWS Identity and Access Management (IAM) FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/iam/faqs/. Viitattu 6.5.2025.

AWS IoT Core endpoints and quotas. 2025. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/general/latest/gr/iot-core.html#message-broker-limits. Viitattu 6.5.2025.

AWS IoT Core FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/iot-core/faqs/. Viitattu 6.5.2025.

AWS IoT Device Shadow service. 2025. *AWS IoT Core Developer Guide.* Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/iot/latest/developerguide/iot-device-shadows.html. Viitattu 6.5.2025.

AWS IoT Greengrass FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/greengrass/faqs/. Viitattu 10.5.2025.

AWS IoT SiteWise Edge. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/iot-sitewise/sitewise-edge/. Viitattu 15.5.2025.

AWS IoT SiteWise FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/iot-sitewise/faqs/. Viitattu 15.5.2025.

AWS Lambda FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/lambda/faqs/. Viitattu 10.5.2025.

AWS Secrets Manager FAQs. 2025. Amazon Web Services. Saatavissa: https://aws.amazon.com/secrets-manager/faqs/. Viitattu 6.5.2025.

Borgosz, L. & Dikicioglu, D. 2024. Industrial Internet of Things: What does it mean for the bioprocess industries? *Biochemical Engineering Journal*, 201, 109122. Saatavissa: https://doi.org/10.1016/j.bej.2023.109122. Viitattu 8.6.2025.

Chen, L. 2022. *Recommendation for Key Derivation Using Pseudorandom Functions*. U.S. Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 800-108r1-upd1. Saatavissa: https://doi.org/10.6028/NIST.SP.800-108r1-upd1. Viitattu 30.8.2025.

Create asset models in AWS IoT SiteWise. 2025. *AWS IoT SiteWise User Guide*. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/iot-sitewise/latest/userguide/create-asset-models.html. Viitattu 18.5.2025.

Dauda, A., Flauzac, O. & Nolot, F. 2024. A Survey on IoT Application Architectures. *Sensors*, 24(16), 5320. Saatavissa: https://doi.org/10.3390/s24165320. Viitattu 16.8.2025.

Desbiens, F. 2023. *Building Enterprise IoT Solutions with Eclipse IoT Technologies: An Open Source Approach to Edge Computing*. Embrun: Apress. Saatavissa: https://doi.org/10.1007/978-1-4842-8882-5. Viitattu 7.6.2025.

DIRECTIVE (EU) 2022/2555 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on measures for a high common level of cybersecurity across the Union, amending Regulation (EU) No 910/2014 and Directive (EU) 2018/1972, and repealing Directive (EU) 2016/1148 (NIS 2 Directive). 14.12.2022. Saatavissa: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32022L2555. Viitattu 27.8.2025.

Distributed Controllers. 2025. siemens.com. Saatavissa: https://www.siemens.com/global/en/products/automation/systems/industrial/io-systems/distributed-controller.html#SIMATICET200SPCPUs. Viitattu 24.4.2025.

DynamoDB table classes. 2025. *Amazon DynamoDB Developer Guide*. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.TableClasses.html. Viitattu 13.9.2025.

Eclipse Foundation. 2024. *2024 IoT and Embedded Developer Survey Report*. Eclipse Foundation. Saatavissa: https://outreach.eclipse.foundation/iot-embedded-developer-survey-2024. Viitattu 15.6.2025.

Eclipse Milo Project. Eclipse Foundation. Saatavissa: https://projects.eclipse.org/projects/iot.milo. Viitattu 24.5.2025.

Eclipse Mosquitto. Eclipse Foundation. Saatavissa: https://mosquitto.org/. Viitattu 27.10.2025.

Eclipse Paho Project. Eclipse Foundation. Saatavissa: https://projects.eclipse.org/projects/iot.paho. Viitattu 24.5.2025.

Eclipse Sparkplug Contributors. 2022. *Sparkplug 3.0.0 – Sparkplug Specification*. Version 3.0.0. Eclipse Foundation. Saatavissa: https://sparkplug.eclipse.org/specification/version/3.0/documents/sparkplug-specification-3.0.0.pdf. Viitattu 10.8.2025.

ENISA. 2025. *Technical Implementation Guidance*. European Union Agency for Cybersecurity. Version 1.0, June 2025. Saatavissa: https://doi.org/10.2824/2702548. Viitattu 30.8.2025.

Euroopan komissio. 2024. *Datasäädös tulee voimaan: mikä muuttuu?* commission.europa.eu. Saatavissa: https://commission.europa.eu/news-and-media/news/data-act-enters-force-what-it-means-you-2024-01-11_fi. Viitattu 20.9.2025.

Figueredo, K., Seed, D. & Wang, C. 2022. A scalable, standards-based approach for IoT data sharing and ecosystem monetization. *IEEE Internet of Things Journal*, 9(8), 5645–5652. Saatavissa: https://doi.org/10.1109/JIOT.2020.3023035. Viitattu 18.6.2025.

Gebremichael, T., Ledwaba, L. P. I., Eldefrawy, M. H., Hancke, G. P., Pereira, N., Gidlund, M. & Åkerberg, J. 2020. Security and Privacy in the Industrial Internet of Things: Current Standards and Future Challenges. *IEEE Access*, 8, 152351–152366. Saatavissa: https://doi.org/10.1109/ACCESS.2020.3016937. Viitattu 20.7.2025.

Harris, J. 2023. *The Eclipse Foundation Announces Sparkplug as an International Standard for a "Plug and Play" Industrial IoT*. 7.11.2023. Saatavissa: https://newsroom.eclipse.org/news/announcements/eclipse-foundation-announces-sparkplug-international-standard-%E2%80%9Cplug-and-play%E2%80%9D. Viitattu 11.8.2025.

Hsu, Y.-F., Irie, R., Murata, S. & Matsuoka, M. 2018. A novel automated cloud storage tiering system through hot-cold data classification. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. San Francisco: IEEE, 492–499. Saatavissa: https://doi.org/10.1109/CLOUD.2018.00069. Viitattu 14.9.2025.

IEC TR 62541-1:2020. OPC Unified Architecture – Part 1: Overview and Concepts. 2020. Saatavissa: https://sales.sfs.fi/fi/index/tuotteet/IEC/IEC/ID9985/6/945861.html.stx. Viitattu 9.8.2025.

Industry IoT Consortium. 2023. *Industry Internet of Things Security Framework (IISF)*. Version 2.0, 12.6.2023. Industry IoT Consortium. Saatavissa: https://www.iiconsortium.org/wp-content/uploads/sites/2/2023/06/IISF-Version-2.pdf. Viitattu 26.8.2025.

ISO/IEC 20237:2023. *Information technology — Sparkplug® version 3.0*. 2023. Saatavissa: https://www.iso.org/standard/86204.html. Viitattu 11.8.2025.

Krishnan, R. S., Kalilulah, S. I., Indu, V. T., Gokuldhev, M., Deepa, S. & Sushma, G. 2024. Advancements and Security Challenges in the Industrial Internet of Things (IIoT). *2024 International Conference on Integration of Emerging Technologies for the Digital World (ICIETDW)*. IEEE. Saatavissa: https://doi.org/10.1109/ICIETDW61607.2024.10939109. Viitattu 26.8.2025.

Manage data storage in AWS IoT SiteWise. 2025. *AWS IoT SiteWise User Guide*. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/iot-sitewise/latest/userguide/manage-data-storage.html. Viitattu 13.9.2025.

Manditereza, K. 2024. HiveMQ. *Implementing Unified Namespace (UNS) With MQTT Sparkplug*. Saatavissa: https://www.hivemq.com/blog/implementing-unified-namespace-uns-mqtt-sparkplug/. Viitattu 22.9.2025.

MQTT. 2025. *AWS IoT Core Developer Guide*. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html. Viitattu 24.5.2025.

OASIS Open. 2019. *MQTT Version 5.0*. OASIS Standard. Saatavissa: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Viitattu 7.6.2025.

OPC Foundation. 2024. OPC 10000-1 – OPC Unified Architecture, Part 1: Overview and Concepts, Release 1.05.04. OPC Foundation. Saatavissa: https://opcfoundation.org/developer-tools/documents/?type=Specification. Viitattu 9.8.2025.

Quincozes, V. E., Quincozes, S. E., Kazienko, J. F., Gama, S., Cheikhrouhou, O. & Koubaa, A. 2024. A survey on IoT application layer protocols, security challenges, and the role of explainable AI in IoT (XAIoT). *International Journal of Information Security*, 23, 1975–2002. Saatavissa: https://doi.org/10.1007/s10207-024-00828-w. Viitattu 18.6.2025.

Rakshe, A. & Dongre, N. 2024. Survey on Security Protocols for IoT. *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*. IEEE. Saatavissa: https://doi.org/10.1109/I2CT61223.2024.10544115. Viitattu 30.8.2025.

Reddy, G. P., Maddireddy, S. R., Kumar, Y. V. P., Prabhudesai, S., Reddy, Y. J. & Deo, S. 2023. OPC UA Implementation for Industrial Automation – Part 1: Evaluating the Performance of PubSub Model. *2023 1st International Conference on Circuits, Power and Intelligent Systems (CCPIS)*. IEEE, 1–6. Saatavissa: https://doi.org/10.1109/CCPIS59145.2023.10291862. Viitattu 7.8.2025.

REGULATION (EU) 2023/2854 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on harmonised rules on fair access to and use of data and amending Regulation (EU) 2017/2394 and Directive (EU) 2020/1828 (Data Act). 13.12.2023/2854. Saatavissa: https://eur-lex.europa.eu/eli/reg/2023/2854/oj. Viitattu 28.8.2025.

REGULATION (EU) 2024/2847 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) No 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act). 23.10.2024/2847. Saatavissa: https://eur-lex.europa.eu/eli/reg/2024/2847/oj. Viitattu 24.9.2025.

Salcher, F., Finck, S. & Hellwig, M. 2024. A smart shop floor information system architecture based on the unified namespace. *2024 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC)*. IEEE. Saatavissa: https://doi.org/10.1109/ICE/ITMC61926.2024.10794387. Viitattu 17.8.2025.

Senthil, R., Nagarajan, V. & Tayong A. 2024. Data Visualization and Predictive Analytics in Manufacturing: A New Paradigm in Maintenance. *2024 International Conference on Progressive Innovations in Intelligent Systems and Data Science (ICPIDS)*. IEEE, 102–107. Saatavissa: https://doi.org/10.1109/ICPIDS65698.2024.00025. Viitattu 18.9.2025.

SIMATIC IOT2050. 2025. siemens.com. Saatavissa: https://www.siemens.com/global/en/products/automation/industrial-computing/iot-gateways/simatic-iot2050.html. Viitattu 24.4.2025.

SIMATIC IOT2050: Getting Started Guide for AWS IoT Greengrass, A5E52619353-AB, 05/2024. 2024. siemens.com. Saatavissa: https://support.industry.siemens.com/cs/document/109972453/simatic-iot2050-getting-started-guide-for-aws-iot-greengrass?dti=0&lc=en-FI. Viitattu 24.4.2025.

SIMATIC IOT2050: Operating Instructions, 07/2024, A5E39456816-AF. 2024. siemens.com. Saatavissa: https://support.industry.siemens.com/cs/document/109974073/simatic-iot2050?dti=0&lc=en-FI. Viitattu 6.9.2025.

Singh, P., Gautam, P. & Arya, N. 2025. Role of edge computing in IoT. *2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 1226–1230. Saatavissa: https://doi.org/10.1109/CSNT64827.2025.10968010. Viitattu 29.6.2025.

Sithiyopasakul, J., Archevapanich, T., Sithiyopasakul, S., Lasakul, A., Purahong, B. & Benjangkaprasert, C. 2024. Implementation of Cloud Computing and Internet of Things (IoT) by Performance Evaluation. *2024 International Electrical Engineering Congress (iEECON 2024)*. IEEE. Saatavissa: https://doi.org/10.1109/IEECON60677.2024.10537945. Viitattu 23.8.2025.

Storage. 2025. *Amazon Timestream Developer Guide*. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/timestream/latest/developerguide/storage.html. Viitattu 13.9.2025.

Traficom. 2025. *Kyberturvallisuuslaki on hyväksytty eduskunnassa – NIS2-direktiivin mukaiset velvoitteet astuvat voimaan 8.4.2025*. traficom.fi. Saatavissa: https://traficom.fi/fi/ajankohtaista/kyberturvallisuuslaki-hyvaksytty-eduskunnassa-nis2-direktiivin-mukaiset-velvoitteet#:~:text=Kyberturvallisuuslaki%20on%20hyv%C3%A4ksytty%20eduskunnassa%20%2D%20NIS2%2Ddirektiivin%20mukaiset%20velvoitteet%20astuvat%20voimaan%208.4.2025%20%7C%20Traficom. Viitattu 20.9.2025.

Tutorial: Getting started with AWS IoT Greengrass V2. 2025. *AWS IoT Greengrass V2 Developer Guide*. Amazon Web Services. Saatavissa: https://docs.aws.amazon.com/greengrass/v2/developerguide/getting-started.html. Viitattu 24.4.2025.

Unified Architecture. 2025. OPC Foundation. Saatavissa: https://opcfoundation.org/about/opc-technologies/opc-ua/. Viitattu 9.8.2025.

What is OPC? 2025. OPC Foundation. Saatavissa: https://opcfoundation.org/about/what-is-opc/. Viitattu 7.8.2025.

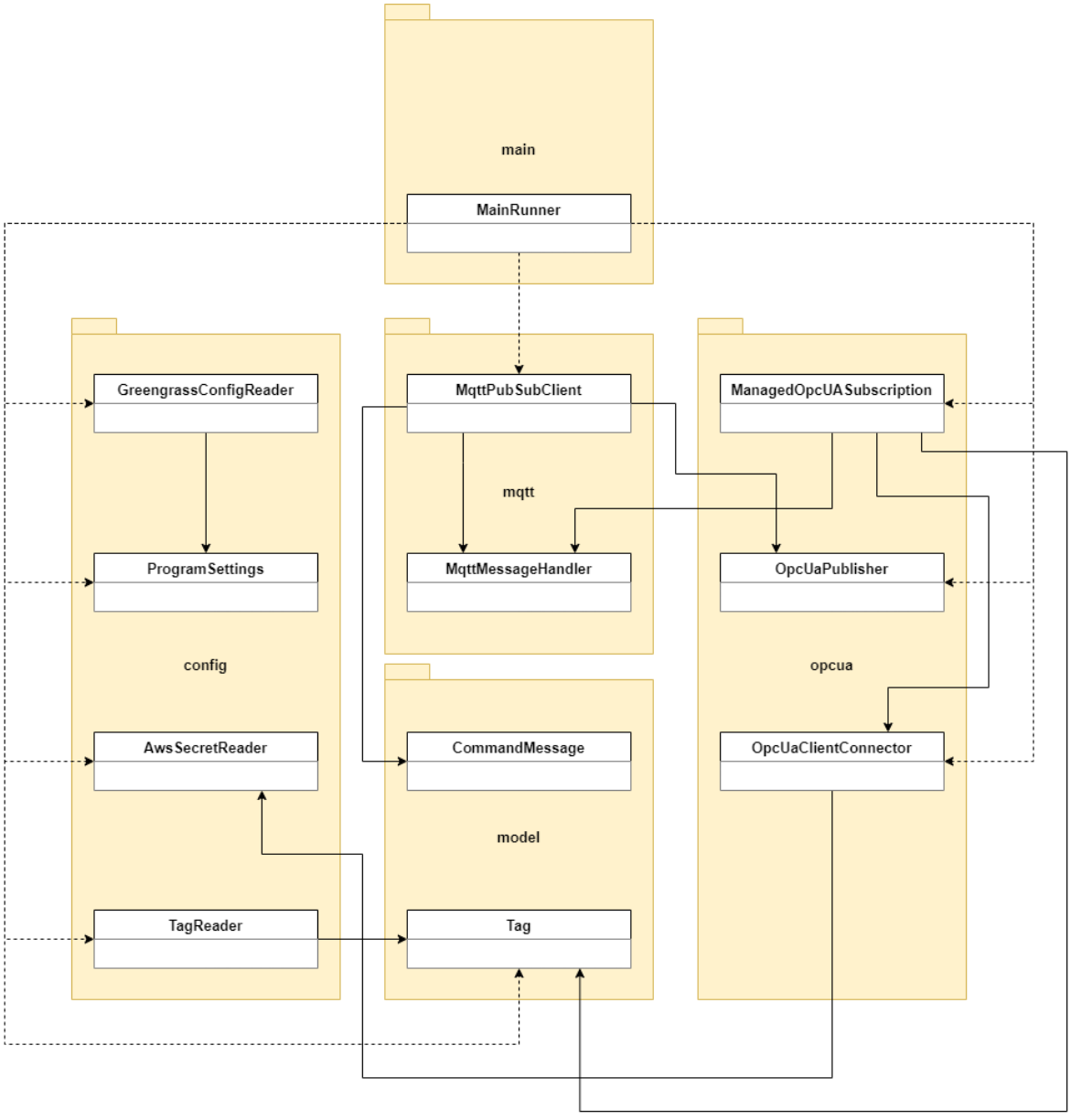
Järjestelmän komponentit ja rooli datavirran kulussa

Taulukko täydentää luvussa 4.3 esitettyä järjestelmäarkkitehtuuria. Siinä on kuviossa 6 numeroin (1–14) esitetyt komponentit sekä niiden roolit datavirran käsittelyssä reunasta pilveen**.**

Nro	Komponentti/palvelu	Rooli datavirrassa
1	PLC (S7-1500, OPC UA)	Toimii OPC UA -palvelimena ja välittää teollisuuden prosessidatan (mittaus- ja ohjaustiedot).
2	SIMATIC IOT2050	Reunalaitteena kerää ja esikäsittelee datan, suorittaa AWS Greengrass -ympäristöä sekä Java-ohjelma.
3	Java-ohjelma	Tilaa prosessidatan OPC UA -palvelimelta, esikäsittelee sen ja julkaisee arvot UNS-rakenteen mukaisina MQTT-viesteinä.
4	Mosquitto MQTT -välityspalvelin	Paikallinen välityspalvelin, toimii UNS-arkkitehtuurin mukaisena yhteisenä tietotilana.
5	MQTT Bridge (Greengrass)	Siirtää valitut aiheet (topics) Mosquittoon ja AWS IoT Coren välillä TLS-suojattuna X.509-varmenteilla.
6	AWS IoT Core	Vastaanottaa MQTT-viestit ja ohjaa ne eteenpäin määriteltyjen sääntöjen mukaisesti AWS-palveluihin.
7	AWS Lambda (1)	Käsittelee IoT Coresta saapuvan datan ja reitittää sen AWS IoT SiteWiseen sekä DynamoDB-välivarastoon.
8	AWS IoT SiteWise	Tallentaa mittausarvot, mahdollistaa suorituskykymittareiden laskennan ja hierarkkisen datamallin käytön.
9	Amazon Managed Grafana	Visualisoi SiteWisen ja Timestreamin dataa lähes reaaliajassa.
10	Amazon DynamoDB	Edullinen välivarasto lyhytaikaiselle datalle raportointia ja eräajoja varten.
11	AWS Lambda (2)	Siirtää dataa DynamoDB:stä eräajoina Timestreamiin, vähentäen jatkuvia kirjoituskustannuksia.
12	Amazon Timestream	Optimoitu aikasarjatietokanta, toimii datalähteenä Grafanalle ja QuickSightille.
13	Amazon QuickSight	Tarjoaa pitkän aikavälin analytiikan ja SQL-pohjaiset raportit Timestreamin datasta.
14	Amazon S3	Pysyvä tallennusalusta esim. raakadatalle, lokitiedostoille ja konfiguraatioille.

Reunasovelluksen arkkitehtuurikaavio

Kaaviossa on esitetty reunasovelluksen keskeiset luokat ja niiden väliset riippuvuudet. Pääluokka *MainRunner* vastaa sovelluksen käynnistämisestä sekä eri komponenttien luomisesta ja yhdistämisestä. Vasemmalla näkyvät konfiguraatioon liittyvät luokat *GreengrassConfigReader*, *ProgramSettings*, *AwsSecretReader*, *TagReader*, jotka huolehtivat ohjelman asetusten ja tagilistan lukemisesta sekä salaisuuksien hallinnasta. Keskellä sijaitsevat MQTT-kommunikaatiota toteuttavat luokat *MqttPubSubClient* ja *MqttMessageHandler* sekä tietomallit *Tag* ja *CommandMessage*. Oikealla puolestaan on OPC UA -yhteyksiin liittyvät luokat *ManagedOpcUASubscription*, *OpcUaPublisher* ja *OpcUaClientConnector*. *ProgramSettings* toimii keskeisenä luokkana, johon lähes kaikki muut komponentit viittaavat, sillä se välittää konfiguraatiotiedot eri osien käyttöön. Luokkien sisäisiä metodeja ja attribuutteja ei ole esitetty, sillä kaavion tarkoituksena on havainnollistaa sovelluksen arkkitehtuuria kokonaisuutena, ei yksityiskohtaista toteutusta.



Pseudokoodi reunasovelluksen pääluokasta

Alla esitetään opinnäytetyön reunasovelluksen pääluokan (MainRunner.java) toiminta pseudokoodina. Pseudokoodi havainnollistaa ohjelman päävaiheet ja logiikan ilman yksityiskohtaista Java-syntaksia. Tarkoituksena on havainnollistaa, miten sovellus alustaa asetukset, lukee konfiguraation, muodostaa MQTT- ja OPC UA -yhteydet, hallitsee monisäikeisyyttä sekä varmistaa sovelluksen hallitun alasajon.

```
MainRunner:

  Käynnistys:
    - Poista käytöstä AWS:n ja HTTP:n ylimääräiset lokit (turvallisuussyistä)
    - Lisää BouncyCastle-turvakirjasto (TLS/SSL)

  Valmistelu:
    - Luo odotuslaskuri (latch) synkronointia varten (odottaa MQTT-yhteyttä)
    - Luo muuttuja MQTT-asiakasinstanssille

  Konfiguraatio:
    - Lue asetukset AWS Greengrassista.
        - Jos epäonnistuu, lopeta ohjelma
    - Lue tagilista tiedostosta.
        - Jos epäonnistuu, lopeta ohjelma

  Komponenttien luonti:
    - Luo OPC UA -julkaisija (käyttää tagilistaa)
    - Luo MQTT-asiakas (käyttää asetuksia ja julkaisijaa)
    - Luo MQTT-viestinkäsittelijä (välittää dataa OPC UA ↔ MQTT)
    - Luo OPC UA -tilausten hallinta (subscription)
    - Luo OPC UA -asiakas (client connector), joka yhdistää tilaukset ja CMD-
      kirjoitukset

  MQTT-yhteyden käynnistys (omassa säikeessä):
    - Yritä yhdistää MQTT-välityspalvelimeen
        - Jos yhteys onnistuu, vapauta latch
        - Jos epäonnistuu kirjaa virhe

  Odota, kunnes MQTT-yhteys on valmis (latch)

  OPC UA -asiakkaan käynnistys:
    - Yhdistä OPC UA -palvelimeen
    - Aloita tilausten seuranta
    - Injektoi asiakas OPC UA -julkaisijaan

  Sammutusrutiini:
    - Katkaise MQTT- ja OPC UA -yhteydet hallitusti ohjelman päättyessä

  Sovelluksen käynnissäpito:
    - Pidä pääsäie aktiivisena loputtomasti (CountDownLatch)
```

Esimerkki JSON-konfiguraation tietueesta

Alla on esimerkki yksittäisen OPC UA -solmun konfiguraatiosta JSON-muodossa. Tällainen tietue määrittää, miten solmu on kuvattu ohjelmallisesti, mukaan lukien sen polku palvelimella, tietotyyppi ja mahdolliset kynnysarvoasetukset.

```
{
    "nodeID": "\"Energy_Meter_EM10_DB\".\"Status\".\"Active-
    Power_kW\"",
    "mappedName": "EnergyMeterEM10/Status/ActivePower_kW",
    "namespaceIndex": 3,
    "dataType": "double",
    "useThreshold": false,
    "threshold": 1.0,
    "thresholdType": "absolute",
    "useThresholdSeconds": true,
    "thresholdSeconds": 60
}
```

Alla oleva taulukko kuvaa JSON-tietueen attribuutit.

Attribuutti	**Kuvaus**	**Esimerkki**
nodeID	Palvelimella esiintyvä tagin nimi, jolla tehdään tilaus.	"\"Energy_Meter_EM10_DB\".\"Status\".\"ActivePower_kW\""
mappedName	Linkitetty nimi, jota käytetään sovelluksessa ja pilvipalveluissa palvelimella esiintyvän täydellisen nimen sijaan.	EnergyMeterEM10/Status/ActivePower_kW
namespaceIndex	OPC UA -nimiavaruuden indeksi.	3
dataType	Tagin datatyyppi, joka vaikutta ohjelmassa, kuinka tagin arvo tulee tulkita tai käsitellä.	double
useThreshold	True/False -valinta, käytetäänkö prosessiarvoa päätöksen teossa.	True
threshold	Prosessiarvon haluttu muutos, jota käytetään vertailussa edelliseen arvoon.	1.0
thresholdType	Absoluuttinen tai prosentuaalinen laskenta vertailussa.	Absolute
useThresholdSeconds	True/False -valinta, käytetäänkö ajallista muutosta päätöksenteossa.	True
thresholdSeconds	Ajallinen minimi muutos, edellisestä päivityksestä.	60

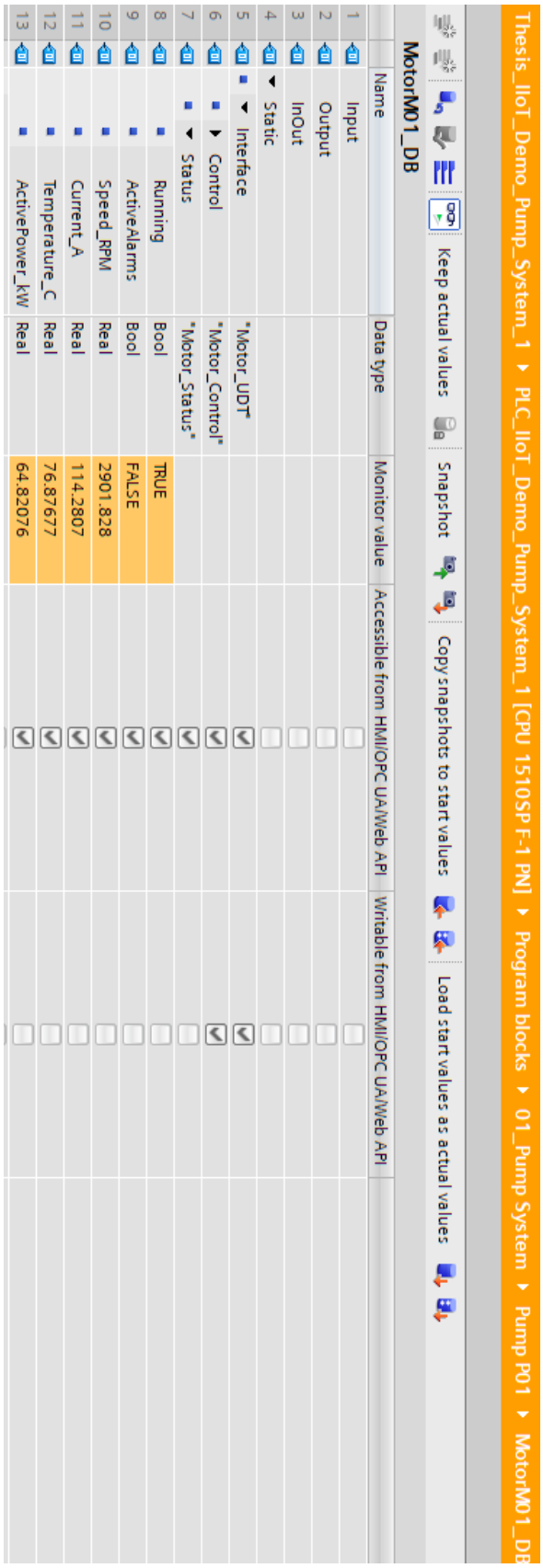
Esimerkki useamman MQTT-viestin hyötykuormasta

Viiden mittausarvon sisältämä MQTT-payload. Yhdyskäytävälaite on pakannut useita mittauksia yhteen julkaisuun kustannustehokkuuden ja viestinvälityksen optimoinnin vuoksi.

```
mqtt/Company1/Site1/Production/PumpSystem1/IOT2050/DDATA

May 25, 2025, 12:57:59 (UTC+0300)

{

  "Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/Tempera-
ture_C_ts_1748166991493": {

    "name": "Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/Tempera-
ture_C",

    "value": 72.99761962890625,

    "timestamp": 1748166991493

  },

  "Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/Vibra-
tion_ts_1748166991493": {

    "name": "Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/Vibration",

    "value": 3.6911497116088867,

    "timestamp": 1748166991493

  },

  "Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/Active-
Power_kW_ts_1748167003492": {

    "name": "Company1/Site1/Production/PumpSystem1/PumpP01/MotorM01/Status/Active-
Power_kW",

    "value": 60.76960754394531,

    "timestamp": 1748167003492,

  },

  "Company1/Site1/Production/PumpSystem1/PumpP02/PressureSensorPS02/Status/Pres-
sure_bar_ts_1748167003493": {

    "name": "Company1/Site1/Production/PumpSystem1/PumpP02/PressureSensorPS02/Sta-
tus/Pressure_bar",

    "value": 3.697439193725586,

    "timestamp": 1748167003493

  },

   "Company1/Site1/Production/PumpSystem1/EnergyMeterEM10/Status/EnergyConsump-
tion_kWh_ts_1748167003494": {

    "name": "Company1/Site1/Production/PumpSystem1/EnergyMeterEM10/Status/EnergyConsump-
tion_kWh",

    "value": 129638.8203125,

    "timestamp": 1748167003494

  }
```

PLC Data block OPC UA -palvelimen lähdedatana

Siemens S7-1500 -PLC:n data block (DB), joka toimii OPC UA -palvelimen lähdedatana. Nämä arvot välitetään yhdyskäytävälaitteelle prosessidatana ja julkaistaan edelleen MQTT-protokollan avulla AWS IoT Coreen pilvipalveluun.

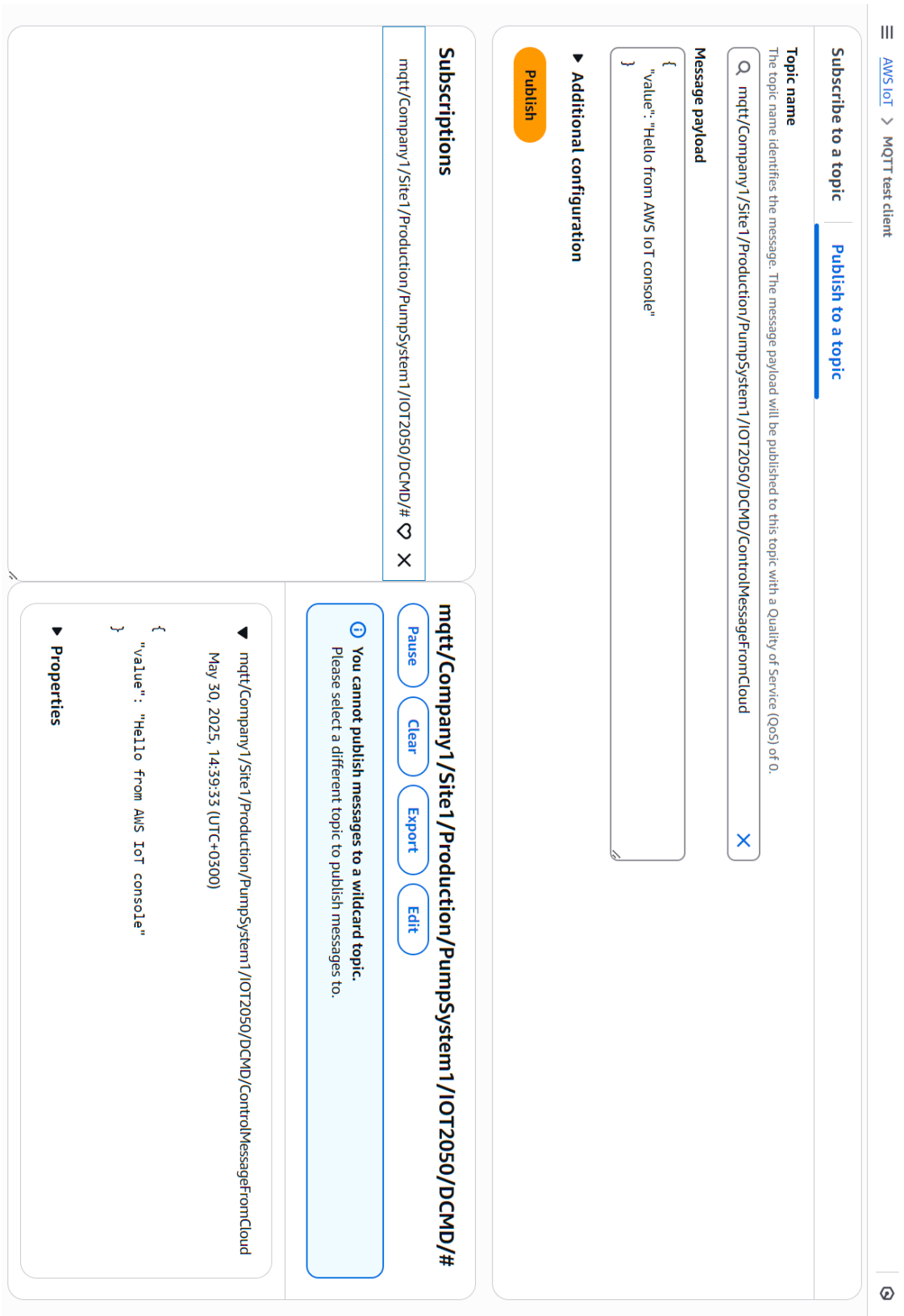


AWS IoT Core – vastaanotettu MQTT-viesti

SIMATIC IOT2050 -yhdyskäytävälaitteen julkaisema MQTT-viesti AWS IoT Core -palvelun MQTT test client -työkalussa. Viesti sisältää mittausarvon, joka perustuu liitteessä 6 1/2 esitettyyn PLC:n muuttujaan MotorM01/Speed_RPM.

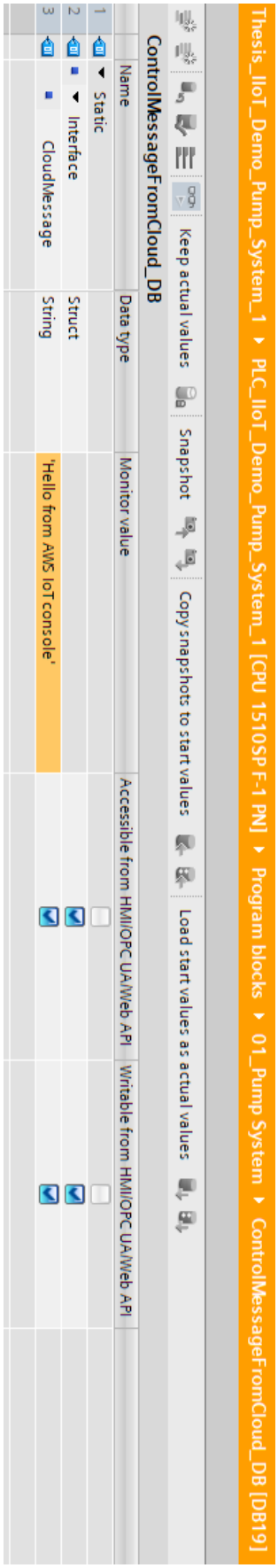
AWS IoT Core – testijulkaistu MQTT-viesti pilvipalvelusta

Esimerkki MQTT-viestin testijulkaisusta AWS IoT Coren MQTT test client -palvelulla. Yhdyskäytävälaite toimii sekä julkaisijana että tilaajana, ja tässä tapauksessa viesti välitetään edelleen OPC UA -palvelimelle.



AWS IoT Core – MQTT-viesti PLC:lle

AWS IoT Coresta lähetetty MQTT-viesti perustuu liitteessä 7 1/2 esitettyyn testijulkaisuun. Viesti välittyy yhdyskäytävälaitteelle ja julkaistaan edelleen PLC:lle OPC UA -protokollan kautta.



AWS Lambda – prosessidatan tallennus DynamoDB:hen

Alla on esimerkki AWS Lambda -funktiosta, joka toteutettiin osana järjestelmän datankeruuta. Funktio suoritetaan AWS IoT Coren sääntöjen perusteella. Se suodattaa MQTT-viestistä prosessimittaukset ja tallentaa ne DynamoDB-tauluun IoT_ProcessBuffer. Jokainen tallennettu kohde saa aikaleiman ja TTL-arvon (Time to Live), jolloin vanhentuneet tiedot poistuvat automaattisesti 24 tunnin kuluttua. Tämä mahdollistaa kustannustehokkaan puskuroinnin ja nopean väliaikaistallennuksen ennen datan jatkokäsittelyä. Koodi on toteutettu Node.js-ohjelmointikielellä (JavaScript) ja hyödyntää AWS SDK:ta DynamoDB-kirjoituksia varten.

```
const AWS = require('aws-sdk');
const dynamodb = new AWS.DynamoDB.DocumentClient();
const TABLE_NAME = 'IoT_ProcessBuffer';
const TTL_SECONDS = 86400; // 24h

/**
 * Lambda triggered by AWS IoT Core rule
 * - Filters valid process measurements (ignores heartbeat & metadata
     fields)
 * - Writes valid measurements to DynamoDB with TTL
 */
exports.handler = async (event) => {
  console.log("Received event:\n", JSON.stringify(event, null, 2));

  for (const [key, measurement] of Object.entries(event)) {
    // Process only keys that are actual measurement keys (e.g. in-
clude "_ts_")
    if (!key.includes("_ts_")) {
      console.debug(`Skipping non-measurement key: ${key}`);
      continue;
    }

    // Validate the structure of the measurement object
    if (
      typeof measurement !== 'object' ||
      typeof measurement.name !== 'string' ||
      typeof measurement.timestamp !== 'number' ||
      typeof measurement.value === 'undefined'
    )
```

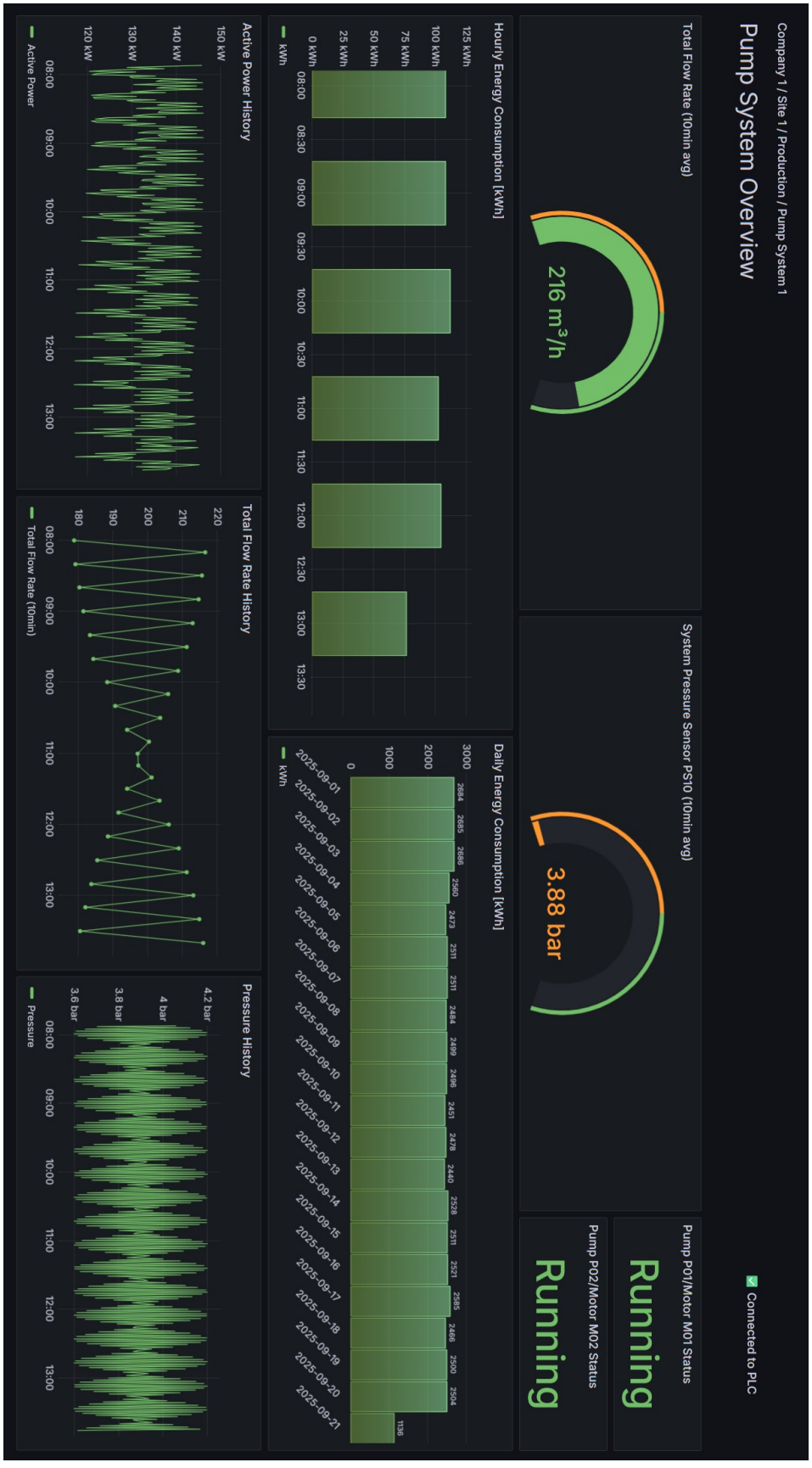
(jatkuu)

(jatkuu)

```
    {
      console.warn(`Skipping invalid or malformed measurement for
key:
      ${key}`);
      continue;
    }

    // Optional: skip heartbeats
    if (measurement.name.toLowerCase().includes("heartbeat")) {
      continue;
    }

    // TTL expiration time
    const expireAt = Math.floor(Date.now() / 1000) + TTL_SECONDS;

    const item = {
      id: `${measurement.name}_${measurement.timestamp}`,
      name: measurement.name,
      timestamp: measurement.timestamp,
      value: measurement.value,
      expireAt
    };

    const params = {
      TableName: TABLE_NAME,
      Item: item
    };

    try {
      await dynamodb.put(params).promise();
      console.log(`Stored record for ${item.name} at
${item.timestamp}`);
    } catch (err) {
      console.error(`Failed to store record for ${item.name}:`, err);
    }
  }
};
```

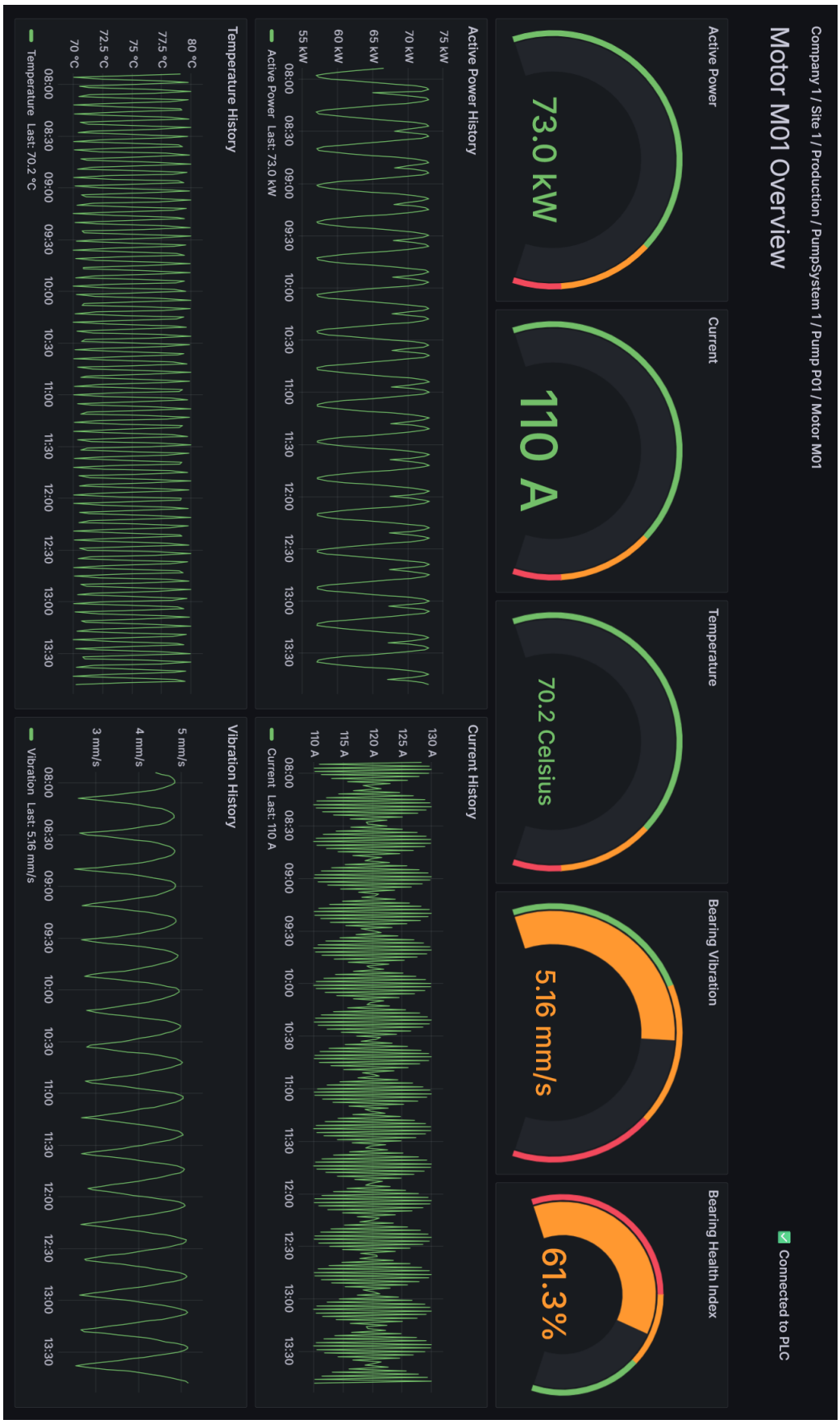
Amazon Managed Grafana – pumppausjärjestelmän yleisnäkymä

Alla oleva kuva esittää Amazon Managed Grafanalla toteutetun pumppausjärjestelmän yleisnäkymän. Prosessidata tulee reaaliaikaisesti AWS IoT SiteWise -palvelusta ja Amazon Timestream -aikasarjatietokannasta.



Amazon Managed Grafana – moottorin M01 yleisnäkymä

Alla oleva kuva esittää Amazon Managed Grafanalla toteutetun moottorin M01 yleisnäkymän. Visualisointi näyttää lähes reaaliaikaiset moottorin tilatiedot AWS IoT SiteWisesta.



SQL-kysely Grafanaa varten

Alla on esimerkki SQL-kyselystä, jolla laskettiin pumppausjärjestelmän energiankulutus päivätasoilla Amazon Timestream -tietokannasta. Kysely hyödyntää binning-toimintoa, jolla data ryhmitellään vuorokausittain, ja suodattaa epärealistisen suuret arvot pois tuloksista. Tämä versio palauttaa tulokset päiväkohtaisella tunnisteella (day_label), mikä soveltuu suoraan Grafanan pylväsdiagrammeihin.

```
WITH all_data AS (
  SELECT
    BIN(time, 1d) AS binned_day,
    measure_value::double AS value
  FROM "IOT2050DataDB"."IOT2050StructuredDataTable"
  WHERE measure_name = 'delta_kWh'
    AND measure_value::double IS NOT NULL
    AND time BETWEEN date_trunc('Month', now()) AND now()
),
bad_days AS (
  SELECT binned_day
  FROM all_data
  GROUP BY binned_day
  HAVING MAX(value) > 1000
),
filtered_data AS (
  SELECT *
  FROM all_data
  WHERE binned_day NOT IN (
    SELECT binned_day FROM bad_days
  )
)
SELECT
  format_datetime(binned_day, 'yyyy-MM-dd') AS day_label,
  SUM(value) AS total_kWh
FROM filtered_data
GROUP BY format_datetime(binned_day, 'yyyy-MM-dd')
ORDER BY day_label ASC
```

SQL-kysely QuickSightia varten

Alla on esimerkki SQL-kyselystä, jolla laskettiin pumppausjärjestelmän energiankulutus päivätasoilla Amazon Timestream -tietokannasta. Kysely hyödyntää binning-toimintoa ja suodattaa epärealistisen suuret arvot pois tuloksista. Tämä versio palauttaa tulokset muodossa, joka sopii Amazon QuickSightin analytiikkaan ja KPI-raportointiin.

```
WITH all_data AS (
  SELECT
    BIN(time, 1d) AS binned_day,
    measure_value::double AS value
  FROM "IOT2050DataDB"."IOT2050StructuredDataTable"
  WHERE measure_name = 'delta_kWh'
    AND measure_value::double IS NOT NULL
    AND time BETWEEN date_trunc('month', now() - interval '1' month)
AND now()
),
bad_days AS (
  SELECT binned_day
  FROM all_data
  GROUP BY binned_day
  HAVING MAX(value) > 1000
),
filtered_data AS (
  SELECT *
  FROM all_data
  WHERE binned_day NOT IN (
    SELECT binned_day FROM bad_days
  )
)
SELECT
  binned_day AS "Daily Consumption",
  SUM(value) AS total_kWh
FROM filtered_data
GROUP BY binned_day
ORDER BY binned_day ASC
```

AWS QuickSight – energiankulutuksen raportti

Alla oleva kuva esittää pumppausjärjestelmän energiankulutuksen tunti-, päivä- ja viikkokohtaisen raportin Amazon QuickSight -palvelussa.