VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Miika Rinta-Korkeamäki

TEHDASVISUALISOINTIEN TULEVAISUUS: ARM64-ARKKITEHTUURIIN PERUSTUVIEN LAITTEIDEN HYÖDYNTÄMINEN KONTITETTUJEN HYBRIDIPILVIYMPÄRISTÖJEN YHTEYDESSÄ

Tekniikka

2025

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Miika Rinta-Korkeamäki
Opinnäytetyön nimi	Tehdasvisualisointien tulevaisuus: ARM64-arkkitehtuuriin perustuvien laitteiden hyödyntäminen kontitettujen hybridipilviympäristöjen yhteydessä
Vuosi	2025
Kieli	suomi
Sivumäärä	55
Ohjaaja	Johan Dams

Tämän opinnäytetyön tavoitteena oli tutkia ARM64-arkkitehtuuriin perustuvien laitteiden soveltuvuutta teollisiin tehdasvisualisointeihin kontitetussa hybridipilviympäristössä. Työ toteutettiin Mirka Oy:n toimeksiannosta osana Smart Factory -alustan kehitystyötä, jonka tarkoituksena on digitalisoida tehtaan tuotantolaitteet käyttämään analytiikka- ja visualisoinitiratkaisuja. Tutkimusongelmana oli selvittää, onko ARM64-laitteilla mahdollista toteuttaa kustannustehokas ja kevyeen laskentaan soveltuva ratkaisu tehdasympäristöön olemassa olevan x86-arkkitehtuurin ratkaisun rinnalle.

Työssä hyödynnettiin Azure DevOps -palvelua ja sen CI/CD-putkistoja, Microsoft Azure -pilvipalvelualustaa sekä Docker-konttiteknologiaa. Kehitystyön yhteydessä olemassa olevaa automatisoitua asennusprosessia ja ohjelmistokontteja laajennettiin ARM64-arkkitehtuurille sopivaksi. Visualisoinnin toimivuutta testattiin Raspberry Pi -laitteella sekä resurssien käytön että käyttövarmuuden näkökulmasta.

Tulokset osoittivat, että ARM64-pohjaiset laitteet ovat kykeneviä suorittamaan kevyeen laskentaan soveltuvia tehtäviä kuten visualisointia lähes yhtä hyvin kuin x86-pohjaiset laitteet, mikäli komponentit ja kotelointi ovat riittävän laadukkaita. Kustannusvertailu arkkitehtuurien välillä osoitti huomattavan säästöpotentiaalin ARM64-pohjaisten laitteiden käytössä. ARM64-arkkitehtuuri tarjoaa varteenotettavan vaihtoehdon teollisiin visualisointitarpeisiin.

Avainsanat ARM64-arkkitehtuuri, CI/CD, hybridipilviympäristö, kontittaminen, pipeline, visualisointi

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

ABSTRACT

Author	Miika Rinta-Korkeamäki
Title	Future of Factory visualizations: Utilization of ARM64 architecture based devices in containerized hybrid cloud environment
Year	2025
Language	Finnish
Pages	55
Name of Supervisor	Johan Dams

The objective of this thesis was to investigate the suitability of ARM64-based devices for industrial factory visualizations in a containerized hybrid cloud environment. The work was carried out on behalf of Mirka Oy as part of the development of the Smart Factory platform, which aims to digitalize factory production equipment to utilize analytics and visualization solutions. The research problem was to determine whether ARM64 devices can provide a cost-effective and lightweight computing solution alongside the existing x86-based architecture in a factory setting.

The thesis utilized the Azure DevOps service and its CI/CD pipelines, the Microsoft Azure cloud platform, and Docker container technology. During the development work, the existing automated installation process and software containers were extended to support the ARM64 architecture. The functionality of the visualization was tested on a Raspberry Pi device from the perspectives of resource usage and operational reliability.

The results showed that ARM64-based devices can perform lightweight computing tasks such as visualization almost as well as x86-based devices, provided that the components and enclosures used are of sufficient quality. The cost comparison between the architectures revealed significant savings potential in favor of ARM64-based devices. ARM64 architecture thus offers a viable alternative for industrial visualization needs.

Keywords ARM64-architecture, CI/CD, containerization, hybrid cloud environment, pipeline, visualization

SISÄLLYS

KUVAT

LYHENTEET

CI/CD	Continuous Integration / Continuous Delivery / Deployment
DevOps	kehitys ja operointi (engl. development and operations)
ERP	toiminnanohjausjärjestelmä (engl. Enterprise Resource Planning)
IoT	Internet of Things
IoS	Internet of Services
IPC	teollisuus-PC (engl. Industrial PC)
ISA	International Society of Automation
MES	tuotannonseuranta (engl. Manufacturing Execution System)
MOM	tuotantotoimintojen hallinta (engl. Manufacturing Operations Management)
MQTT	Message Queuing Telemetry Transport
NIST	National Institute of Standards and Technology
OPC UA	Open Platform Communications Unified Architecture
OT	operatiivinen teknologia (engl. Operational Technology)
PLC	ohjelmoitava logiikka (engl. Programmable Logic Controller)
PR	Pull Request
SaaS	sovelluspalvelu (engl. Software as a Service)
YAML	merkintäkieli (engl. Yet Another Markup Language)

1 JOHDANTO

Tämä opinnäytetyö toteutettiin Mirka Oy:n toimeksiannosta. Opinnäytetyön aiheena oli tutkia ARM64-arkkitehtuuriin perustuvien laitteiden hyödyntämistä kontitettujen hybridipilviympäristöjen yhteydessä. Tutkimuksen ohessa tehdään kehitystyö, jossa kehitetään automatisoitu asennusprosessi ja ohjelmistokontit ARM64-arkkitehtuurille sopivaksi. Tämä kehitystyö tehdään olemassa olevan x86-arkkitehtuurille kehitetyn asennusprosessin rinnalle Mirkan Smart Factory -alustalle.

Työhön liittyvä hanke liittyy olennaisesti Mirkan Smart Factory -alustaan, joka yhdistää tehtaan laitteet hybridipilviympäristöön. Voidaan siis puhua ns. OT-alustasta, jossa ohjelmoitavat järjestelmät tai laitteet toimivat vuorovaikutuksessa fyysisen ympäristön kanssa (NIST, 2018, s. 101). Tällä alustalla pystytään esimerkiksi keräämään tehtaan laitteilta dataa PLC-kontrollereilta IPC-laitteiden avulla, ja tätä dataa voidaan Smart Factory -alustan avulla visualisoida erilaisissa visualisointisovelluksessa (esim. Grafana).

Kyseinen kehittämis- ja tutkimustyö tehdään, jotta olisi mahdollista saada selville, soveltuuko ARM64-arkkitehtuurin laitteet kevyeen laskentaan tehdasympäristössä. Lisäksi samalla voidaan tutkia, kuinka isoja kustannussäästöjä on mahdollista saada aikaan verrattuna olemassa olevaan Siemensin IPC-ratkaisuun. Vertailua voidaan myös tehdä suorituskyvyn kannalta x86- ja ARM64-arkkitehtuurien välillä.

2 MIRKA

Mirka on hioma- ja kiillotustarvikkeiden tuottamiseen erikoistunut yritys, jonka Onni Aulo perusti vuonna 1943 Helsingissä. Yrityksen alkuvaiheiden jälkeen nykyiseksi päätoimipaikaksi valikoitui Jepua vuonna 1962. Yrityksellä on useita tytäryhtiöitä esimerkiksi Belgiassa, Isossa-Britanniassa, Italiassa ja Yhdysvalloissa. Mirka työllistää lähes 1 600 työntekijää tällä hetkellä ympäri maailmaa (Mirka, n.d.).

Pintojen viimeistelyn ja tarkkuushionnan lisäksi yritys valmistaa laajasti erilaisia innovatiivisia kokonaisratkaisuja. Yksi tällainen on pölyttömät ratkaisut, jossa hionnasta ei synny ollenkaan pölyä. Yrityksen tuotevalikoimaan sisältyy esimerkiksi hiomatuotteita, kiillotusaineita ja sähköisiä hiomakoneita. Mirkan tuottamia ratkaisuja hyödynnetään laajasti eri teollisuuden aloilla kuten ajoneuvojen vauriokorjauksissa ja rakentamisessa sekä remontoinnissa.

Suomessa Mirkalla on viisi eri tuotantolaitosta Jepualla, Karjaalla, Oravaisissa, Pietarsaaressa sekä Nurmijärvellä. Suurin osa tuotannosta keskittyy Jepuan päätoimipisteeseen, jossa on myös tuotekehityksen osasto. Tämän lisäksi Mirkalla on Vaasassa toimisto, jossa tämä opinnäytetyö tehtiin.

3 SMART FACTORY -ALUSTA

3.1 Smart Factory yleisesti

Smart Factory (älykäs tehdas) on korkeasti digitalisoitu tehdasympäristö, jossa kytkettyjen laitteiden ja reaaliaikaisen datan avulla pyritään optimoimaan tuotantoprosesseja sekä parantamaan tehokkuutta (Fortoul-Diaz ja muut, 2023, s. 101728). Toisin sanoen älykkään tehtaan perinteisiä valmistusprosesseja pyritään digitalisoimaan niin, että tehtaan laitteilta saatavaa dataa voidaan käyttää valmistusprosessien analysointiin ja avustamaan niitä käyttäviä operaattoreja päivittäisessä työssä. Lisäksi digitalisaatio auttaa liiketoiminnan kehittämistä organisaation ylemmillä tasoilla sekä parantamaan tuotteiden laatua esimerkiksi tuotekehityksessä (SAP, n.d.).

Älykäs tehdas käsiteenä on osa teollisuuden neljättä vallankumousta (Industry 4.0), jossa IoT ja IoS ovat integroitu tiiviisti valmistusympäristöön (Gilchrist, 2016, luku 13). Tämä tarkoittaa sitä, että tehtaan laitteet on mahdollista kytkeä verkkoon ns. kyberfyysisinä järjestelminä, jolloin tuotantoprosessin aikana on esimerkiksi mahdollista saada tietoa, mitä tuotetta ja miten sitä prosessoidaan missäkin tuotantoprosessin vaiheessa.

3.2 ISA-95-standardi

Olennaisena osana älykästä tehdasta voidaan pitää ISA-95-standardia, joka on kansainvälinen standardi automaation kehittämiseen yritysso-vellusten sekä -järjestelmien ja teollisuuden ohjausjärjestelmien välillä. ISA-95-standardin avulla voidaan luoda perusta johdonmukaisille tietoja toimintamalleille. Se helpottaa myös tavarantoimittajien sekä valmistajien viestintää (Lachance, 2021).

Käytännössä ISA-95-standardilla voidaan määritellä esimerkiksi eri loogisten tasojen välisiä vuorovaikutuksia ERP- ja MES-järjestelmien välillä. Tämä mahdollistaa järjestelmien välisen tiedonsiirron selkeyttämisen, tehostaa tuotannon ja liiketoiminnan välistä viestintää sekä parantaa tuotannonohjauksen tehokkuutta ja joustavuutta.

ISA-95-standardi koostuu viidestä eri tasosta (Kuva 1), joista alimmat tasot ovat fyysiseen ympäristöön sijoittuvia tasoja, kun taas ylemmät tasot sijoittuvat enemmän pilviympäristöihin (Lachance, 2021).

Alimpana pyramidissa sijaitsee taso 0, joka on tuotannon fyysinen prosessi. Tällä tasolla saadaan tehtaan laitteista raakadataa isolla volyymillä, ja prosesseja ohjataan tällä tasolla esimerkiksi kuljettimen moottorilla. Saatava data on mitattavissa millisekunteina tai jopa mikrosekunteina.

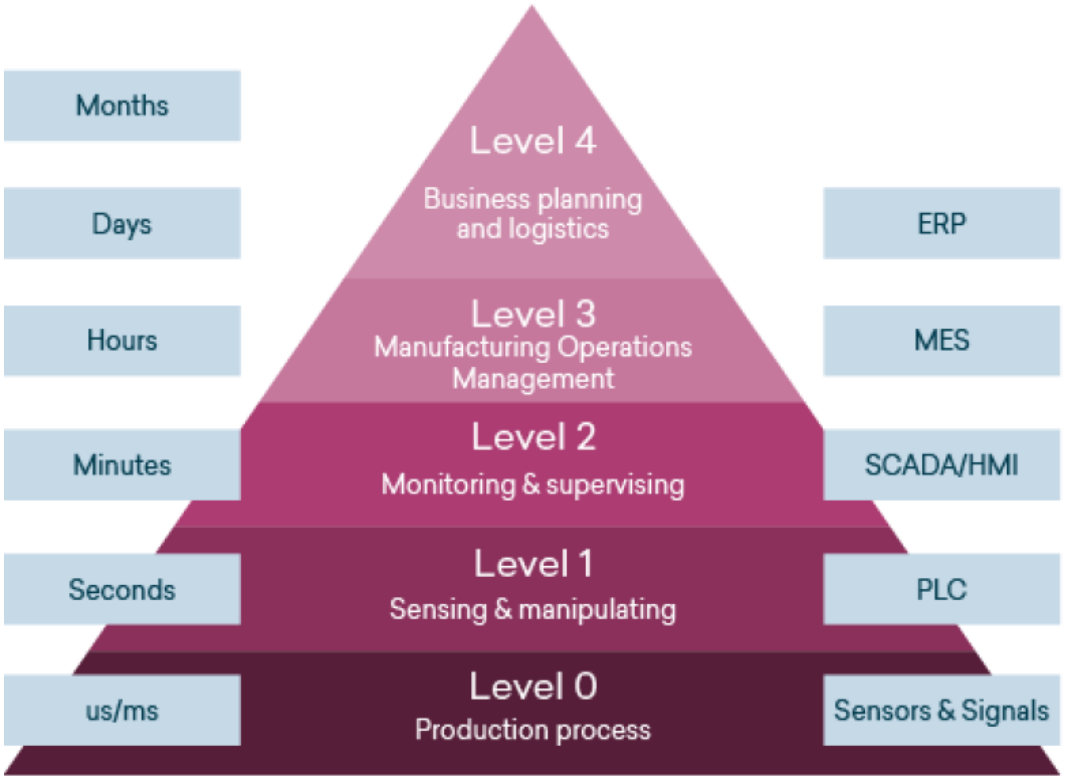
Tasolla 1 on tuotantolaitteen erilaiset sensorit, jotka keräävät reaaliaikaista dataa. Tähän tasoon kuuluu muun muassa PLC-kontrollerit, lämpötila-anturit ja paineanturit. Prosesseja voidaan ohjata tällä tasolla esimerkiksi moottorin ohjauksella. Tässä tasossa data on mitattavissa sekunteina.

Pyramidin tasolla 2 on laitteiden hallinta ja valvonta. Laitteiden tilaa voidaan valvoa esimerkiksi erilaisina visualisointeina. Hallinta toteutetaan automaatio-ohjauslaitteistolla, joka on yhdistetty sovellukseen (esim. SCADA ja HMI). Tällä tasolla puhutaan minuutin aikakäsityksestä.

Tasolla 3 on tuotantotoimintojen hallinta (MOM). Tällä tasolla tapahtuu tuotannon aikataulutus, optimointi ja työkuorman tasapainottaminen. MES-järjestelmät toimivat tällä tasolla. Aikakäsitys tällä tasolla voi olla tunteja.

Pyramidin viimeinen taso 4 on liiketoiminnan suunnittelu ja logistiikka. Tällä tasolla liiketoiminnan tavoitteet ja tuotantotoiminnot pyritään yh-

distämään toisiinsa. Tasolta 3 saatava data on kriittisessä roolissa tämän tason päätösten kanssa. Erilaiset ERP-järjestelmät toimivat tällä tasolla ja aikakäsitys voi olla jopa kuukausia.



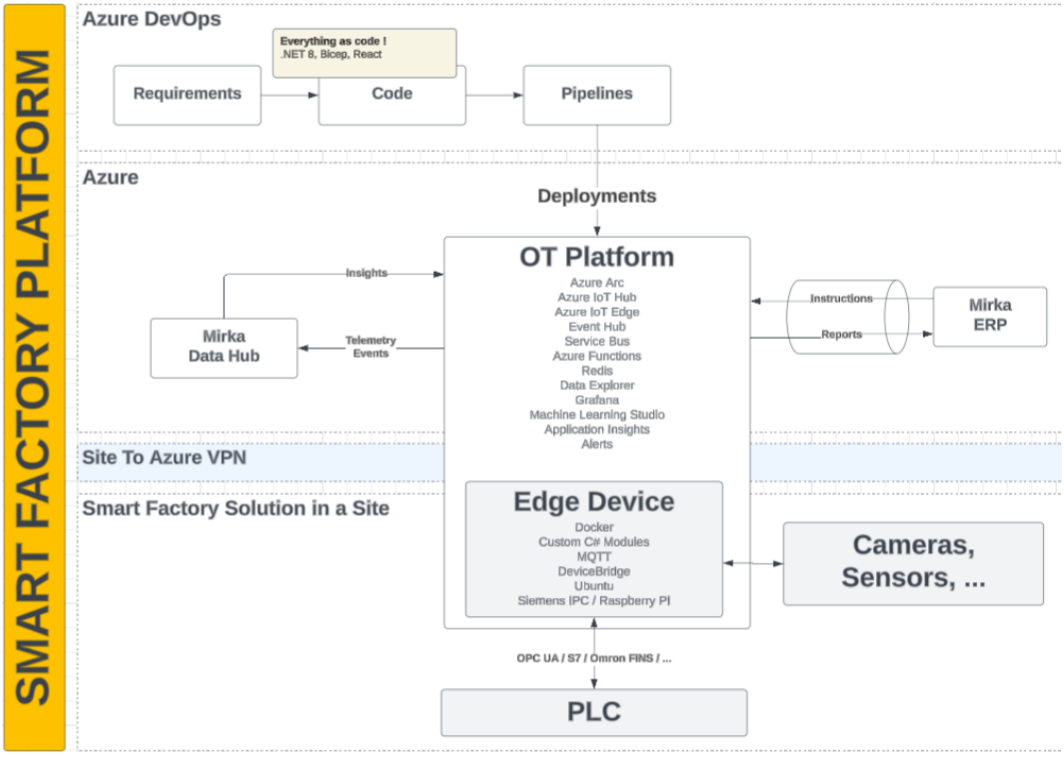
Kuva 1. ISA-95-standardi tasoittain (Brightly, 2021).

3.3 Mirkan Smart Factory -alustan yleiskuvaus ja arkkitehtuuri

Mirkan Smart Factory -alustan kehitys aloitettiin keväällä 2023, jolloin Operational Technology -tiimi perustettiin yritykseen tämän kyseisen alustan kehitystä varten. Alusta toimii linkkinä Mirkan eri tehtaiden valmistuslaitteiden ja hybridipilviympäristön välillä. Jokaisella valmistuslaitteella on yksi kappale x86-arkkitehtuurin pohjautuvia Siemensin Industrial PC -ratkaisuja eli ns. Edge-laitteita, joilla kerätään tehtailta erilaista dataa analysoitavaksi ja visualisoitavaksi.

Korkealta tasolta kuvattuna (Kuva 2), Mirkan oma Smart Factory -alusta koostuu kolmesta eri päätasosta:

1. Azure DevOps on työkalu, jolla hallitaan koko ohjelmistokehitysprosessi.
2. Azure puolestaan sisältää eri hybridipilviympäristöön liittyvät resurssit sekä kehitys- (dev) että tuotantoympäristöjä (prod) varten.
3. Kolmas taso on Smart Factory -ratkaisu tehtailla, joka sisältää itse Edge-laitteen sekä erilaiset sensorit ja PLC:t, joilta dataa voidaan kerätä.



Kuva 2. Mirkan Smart Factory -alusta korkealta tasolta kuvattuna (Mirka, 2023)

Tällä hetkellä, Smart Factory -alusta on käytössä muutamalla Mirkan tehtaalla. Ratkaisua käyttäviä Edge-laitteita on tällä hetkellä kokonaisuudessaan noin 30 laitetta, joista 22 on aktiivisessa tuotantokäytössä. Tuon määrän on tarkoitus kasvaa merkittävästi lähitulevaisuudessa, kun olemassa olevien ja uusien tuotantolaitteiden digitalisointi etenee.

4 RATKAISUN KESKEISET TEKNOLOGIAT

4.1 Azure DevOps

Smart Factory -alustan perustan yksi tärkeimmistä osista on Microsoftin tarjoama Azure DevOps SaaS-palvelu. SaaS käsitteenä tarkoittaa sovelluspalvelua, jossa pilvipalveluntarjoajat ylläpitävät sekä hallinnoivat sovelluksia. Sovellukset operoivat täysin pilvessä, jolloin mitään ei tarvitse asentaa käyttäjien koneille. Nämä sovellukset toimitetaan asiakkaalle eli tässä tapauksessa yritykselle käytettäväksi ja palvelun käytöstä maksetaan vuosittain tai kuukausittain sen mukaan, kuinka paljon käyttäjät käyttävät tarjottuja sovelluksia. Microsoft Azuren (n.d.) sivuilla todetaankin näin:

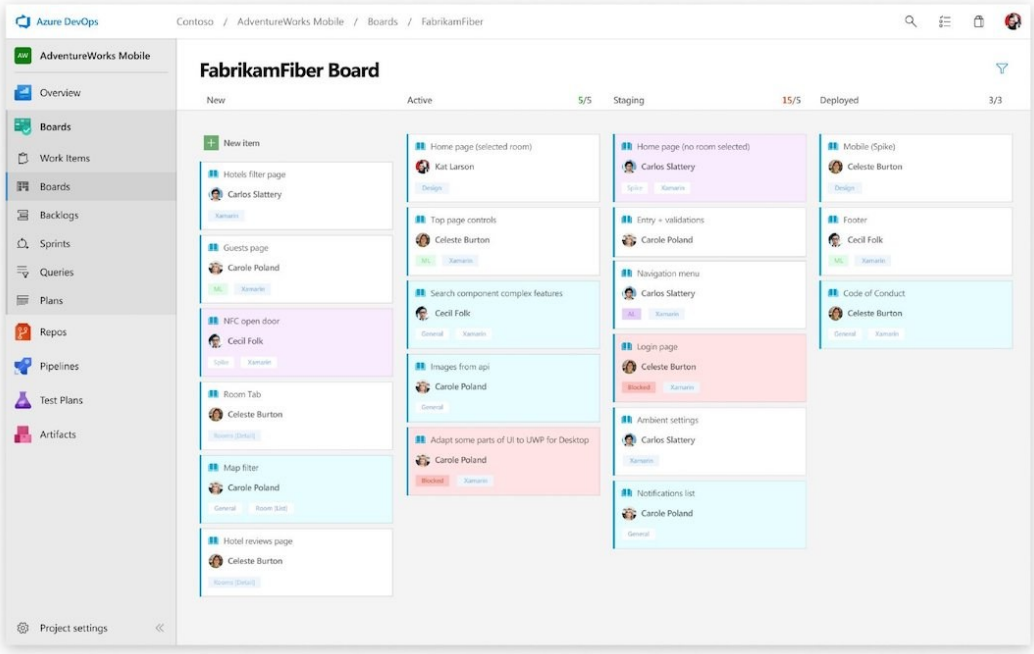
> *"SaaS is a method for delivering software applications over the Internet where cloud providers host and manage the software applications making it easier to have the same application on all of your devices at once by accessing it in the cloud."*

Azure DevOps toimii alustana kehittäjien, projektipäällikköjen, tuoteomistajien, testaajien, UX-suunnittelijoiden ja sidosryhmien välillä. Se sisältää projektinhallintatyökalut sekä laajan versionhallinnan että myös testausohjelmiston, joiden avulla organisaatio voi luoda ja kehittää ohjelmistoja nopeasti CI/CD-mallin mukaisella tavalla. Organisaation sisällä Azure DevOpsissa on myös mahdollista luoda jokaiselle projektille oma projekti-instanssi (Azure DevOps, 2024). Mirkalla on käytössä useita kymmeniä projekteja tällä hetkellä Azure DevOps -palvelussa.

Azuren DevOps-palveluun sisältyy erilaisia työkaluja kuten Azure Boards, Azure Repos ja Azure Pipelines. Näistä ominaisuuksista lisää seuraavissa luvuissa.

4.1.1 Azure Boards

Azure Boards toimii työkaluna agile-kehitysmallissa, jossa voidaan suunnitella ja seurata töitä työkohteiden (work item) avulla. Tässä työkalussa on mahdollista toteuttaa sprinttejä sekä toteuttaa Kanban- että Scrum-ohjelmistokehitysmalleja, jossa työtehtäviä voidaan siirrellä eli vaiheiden välillä (Kuva 3). Organisaation jäsenet voivat itse määritellä taulun eri vaiheet omaan ohjelmistokehitykseen sopivaksi.



Kuva 3. Mallikuva taulusta Azure Boards -työkalussa (Microsoft, 2021).

Palvelussa on erityylisiä työkohteita, joiden avulla projekteja on helppo hallinnoida. Scaled Agile Frameworkin (SAFe) mukaan (2025) työkohteet voidaan määritellä SAFe-viitekehyksen mukaisesti:

- Korkein työkohde tasoltaan hierarkiassa on Epic, joka on merkittävä ja laajavaikutteinen ratkaisun kehityshanke. Tällä tasolla sidosryhmien tavoitteena on määritellä odotetut tulokset sekä liiketoimintahyödyt.

- Seuraavalla tasolla hierarkiassa on Feature eli ominaisuus, jonka avulla Epic-tason työkohde voidaan jakaa pienempiin kokonaisuuksiin. Feature kuvaa tiettyä toiminnallisuutta, joka tuottaa arvoa loppukäyttäjälle. Yleisesti yhden Featuren sisältämä työmäärä on määritelty alle kahteen kuukauteen.
- Featuren alle voidaan jaotella User Storyjä eli käyttäjätarinoita. Se on käyttäjän näkökulmasta luotu työkohde, jonka avulla voidaan kuvata, minkä pienen toiminnallisuuden asiakas haluaa ohjelmaan. Jokainen käyttäjätarina keskittyy tiettyyn toiminnallisuuteen, jota voidaan kehittää vaiheittain ratkaisuun. Kun tarinat pidetään pieninä, jokaisesta iteraatiosta saadaan tuotettua arvoa.
- Käyttäjätarinoiden alle voidaan luoda tehtäviä eli Taskeja. Näiden avulla esimerkiksi laajempi käyttäjätarina voidaan jakaa useammalle kehittäjälle.

Erikseen edellä olevien työkohteiden lisäksi voidaan mainita Bug, jonka avulla voidaan seurata ja hallita ohjelmistossa ilmeneviä vikoja eli bugeja järjestelmällisesti.

Operational Technology -tiimissä käytetään ns. Scrumban-menetelmää, joka on Scrum- ja Kanban-menetelmien risteytys. Tässä tapauksessa Scrum esiintyy tiimin toiminnassa esimerkiksi tavalla, jossa joka aamu pidetään päivittäinen kokous. Kokouksessa käydään läpi, mitä kukin on tiimissä tehnyt. Kanban puolestaan tulee esiin juuri aiemmin mainitun taulun muodossa. Taulu on jaettu moneen eri sarakkeeseen työvaiheittain ja sen lisäksi työkohteisiin pyritään laittamaan värit sen mukaan, mihin projektiin työ esimerkiksi liittyy.

4.1.2 Azure Repos

Azure Repos tarjoaa tiimin projekteihin Git-versionhallinnan, johon kaikki koodi voidaan tallettaa turvallisesti. Yleisessä versionhallintakäytännössä kehittäjän valmis koodi viedään ns. pull request -pyyntöön (PR), jossa koodi katselmoidaan ennen kuin se voidaan julkaista ja yhdistää se versionhallinnassa olevaan toiseen haaraan. Lisäksi kyseinen

ominaisuus tarjoaa mahdollisuuden käyttää TFVC-versionhallintaa (Team Foundation Version Control), joka toimii ns. keskitettynä versionhallintana, missä tiimin jäsenillä on vain yksi versio joka tiedostosta heidän koneillaan (Microsoft DevOps, 2024).

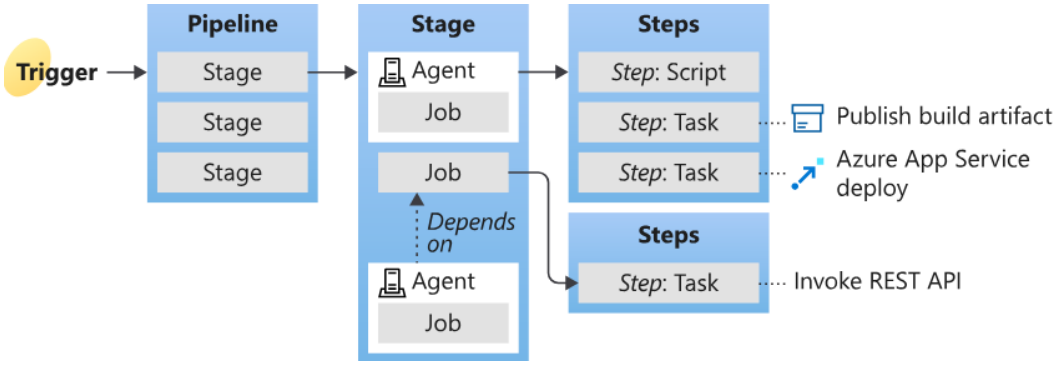
Tiimissäni käytetään juuri ensimmäisenä mainittua Git-versionhallintaa, mutta koodin kehittäminen ja sen katselmointi eroaa hieman aiemmin mainitusta tavasta. Käytämme ns. *trunk based development* -versionhallintamenetelmää, jossa kaikki kehittäjät työskentelevät käytännössä yhden päähaaran kanssa. Toisin sanoen tällaisessa kehityksessä kehittäjät tekevät pieniä muutoksia omissa kehityshaaroissaan, jotka sitten pull request -pyyntöjen kautta yhdistetään päähaaraan (yleensä main). Tämä tehostaa CI/CD-menetelmän toteuttamista tehokkaasti projekteissa.

4.1.3 Azure Pipelines

Olennainen osa CI/CD-ohjelmistokehitystä on ns. ohjelmistoputki eli pipeline. Pipeline on ketju erilaisia funktioita tai tehtäviä, jotka määritellään erilliseen määrittelytiedostoon. Näin pystytään automatisoimaan esimerkiksi koodin tarkistusta sekä testausta saumattomasti kehityksen ohessa. Voidaankin kuvailla pipelinen koostuvan seuraavista vaiheista; koodin rakentaminen (build), testaaminen (test) ja käyttöönottaminen tai julkaiseminen (deploy). Automaatio tuottaa tyypillisesti konsistentin lopputuloksen ja lisäksi automaatiokoodi toimii dokumentataationa, mitä eri vaiheissa tapahtuu.

Azure Pipelines on yksi monesta pipeline-työkaluista, joita on ohjelmistokehityksessä käytössä tänä päivänä. Sen päätoimintaperiaate (Kuva 4) perustuu muutamaan eri komponenttiin. Trigger on niin sanottu käynnistin, joka voi olla esimerkiksi koodiin tehty muutos, ja jonka jälkeen pipeline aloittaa tekemään prosessejaan. Pipelinessa voi olla useampia eri vaiheita (stage), joiden alle voidaan järjestää töitä (job). Työ puolestaan pyörii agentilla, joka on tässä tapauksessa pilvessä pyörivä

virtuaalikone. Lopulta töiden sisälle voidaan sisällyttää useampia vaiheita (step), jotka voivat suorittaa tiettyjä skriptejä (script) tai tehtäviä (task). Koko pipeline-prosessi luodaan erilliseen konfiguraatiotiedostoon esimerkiksi YAML-merkintäkielellä.



Kuva 4. Pipelinen toimintaperiaate (Microsoft, 2024).

Mirkan Smart Factory -tiimissä ohjelmistoputket pyritään tekemään jokaiselle moduulille ja projektille erikseen YAML-merkintäkielellä. Kehityksessä käytetään usein ns. PR-ohjelmistoputkea, joka käynnistyy aina, kun kehityshaaraan julkaistaan muutoksia. Näin voidaan varmistua, että ohjelmassa tai moduulissa ei ole virheitä ennen kuin koodi voidaan hyväksyä päähaaraan yhdistettäväksi. Tämän lisäksi käytössä on myös ns. julkaisuohjelmistoputket (publish pipeline) ja esijulkaisuohjelmistoputket (prerelease pipeline). Nämä ohjelmistoputket tekevät muutoin samat prosessit kuin PR-ohjelmistoputkessa, mutta myös rakentavat moduulit ohjelmistokonteiksi ja julkaisevat ne Azuren konttirekisteriin. Lisäksi jokaiselle Edge-laitteelle on oma käyttöönotto-ohjelmistoputki (deployment pipeline), joihin määritellään, mitkä moduulikontit otetaan käyttöön milläkin laitteella.

4.2 Microsoft Azure

Microsoft Azure on Microsoftin tarjoama pilvipalvelualusta, joka tarjoaa laajan valikoiman erilaisia pilvipalveluita liittyen esimerkiksi tallennusti-

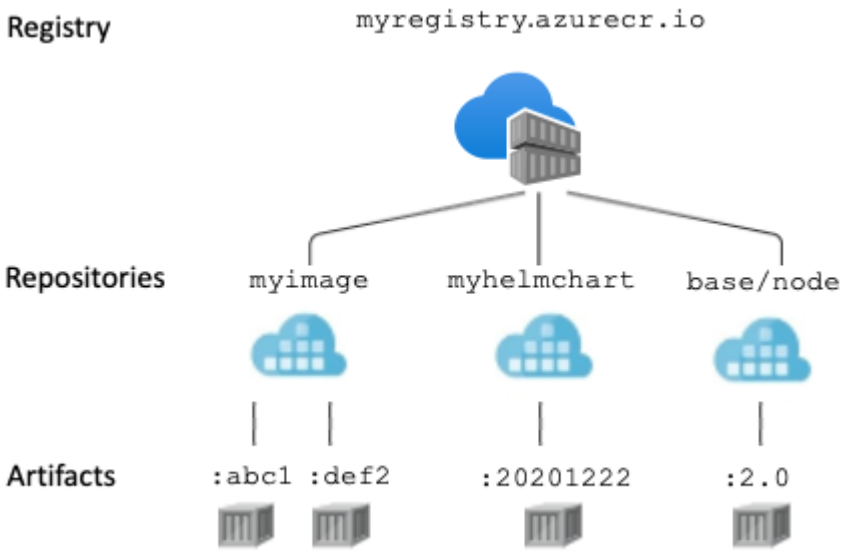
laan, verkkopalveluihin, analytiikkaan ja tekoälyyn. Azuressa asiakkaiden on mahdollista itse valita, mitä palveluita he haluavat käyttöönsä kehittääkseen uusia sovelluksia tai ajaakseen jo olemassa olevia sovelluksiaan. Näitä edellä mainittuja palveluita isännöidään Microsoftin datakeskuksissa ympäri maailmaa, mutta viime kädessä Microsoft huolehtii niiden hallinnoinnista ja tarjoamisesta (Microsoft Azure, n.d.). Mirkalla on käytössä laajasti eri Azuren tarjoamia palveluita.

Yksi tärkeä Edge-laitteiden hallinnointiin liittyvä palvelu on Azure Arc. Sen avulla Azuren palveluita on mahdollista käyttää myös tehtailla sinne sijoitetuissa laitteissa tai palvelimissa. Se toimii siltana, joka laajentaa Azure-alustan niin, että organisaatio voi rakentaa sovelluksia ja palveluita joustavasti Edge-laitteilla ja hybridi- ja monipilviympäristöissä (Microsoft Azure, n.d.).

Toinen keskeinen palvelu Edge-laitteiden hallintaan liittyen on Azure IoT Hub. Tämä palvelu toimii ns. keskuspaikkana (hub) IoT-laitteiden ja pilvipalvelun välillä. Se tukee kaksisuuntaista viestintää pilven ja laitteiden välillä, joiden avulla esimerkiksi datan lähettäminen laitteelta pilveen (device-to-cloud-message) ja komentojen vastaanottaminen pilvestä laitteella (cloud-to-device-messsage) on mahdollista. Lisäksi IoT Hub palveluna tuo laitteille lisää turvaa. Jokaiseen laitteeseen on lisätty autentikointi sekä salaus ja ne on yksilöity IoT Hubissa sijaitsevaan identiteettirekisteriin (Microsoft Learn, 2025).

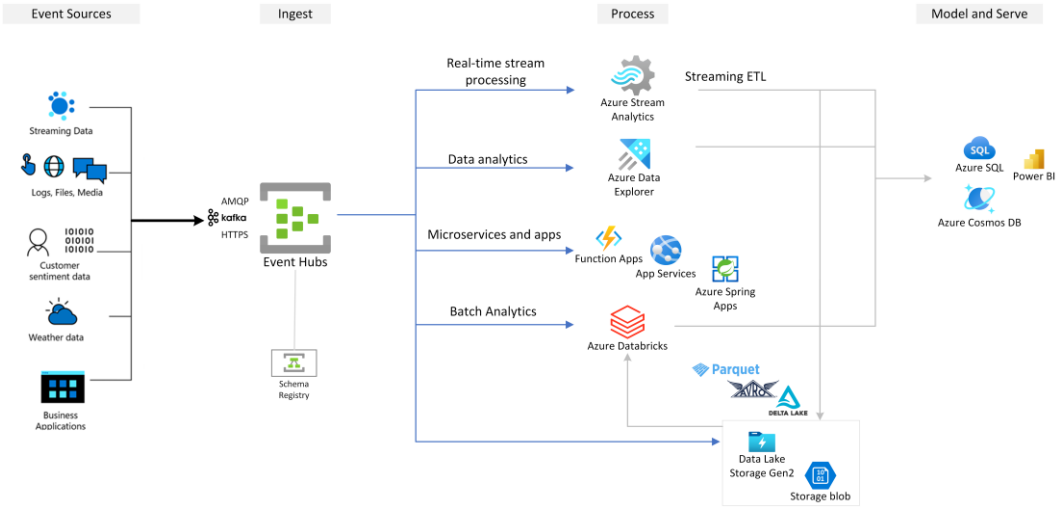
Ohjelmistokonttien hallintaa varten Mirkalla on käytössä Azuren tarjoama konttirekisteri, Azure Container Registry. Konttirekisteri itsessään on palvelu, johon voidaan tallentaa ohjelmistokonttikuvia (container image) ja jakaa niitä eteenpäin. Ohjelmistokonttikuvat tallennetaan repositorioon yksilöitynä esimerkiksi nimiavaruuden (namespace) avulla. Artifakti konttirekisterissä tarkoittaa puolestaan ohjelmistokonttikuvaa, jolla on yksi tai useampi tunniste (tag), yksi tai useampi kerros (layer) ja se on yksilöity manifestin avulla (Microsoft Learn, 2024) (Kuva 5).

Manifesti puolestaan määrittelee ohjelmistokontin kerrokset sekä artifaktit. Manifestejä voi olla useampi, jos ohjelmistokontti on rakennettu useammalle arkkitehtuurille.



Kuva 5. Azure Container Registry -konttirekisterin rakenne (Microsoft, 2024).

Telemetrian ja tapahtumien käsittelyyn Mirkalla on käytössä Azure Event Hub. Se on reaaliaikainen datastriimaukseen pohjautuva palvelu, joka voi vastaanottaa ja käsitellä miljoonia tapahtumia sekunnissa mistä tahansa lähteestä (Microsoft Learn, 2024). Kyseessä on siis eräänlainen puskuri tai keräyspiste, johon dataa voidaan kerätä laajasti monelta eri laitteelta. Event Hubiin vastaanotettuja tapahtumia voidaan lähettää eteenpäin käsiteltäväksi erilaisiin analytiikkatyökaluihin, kuten Azure Data Exploreriin tai Azure Databricks -analytiikkapalveluun (Kuva 6).



Kuva 6. Mallikuva tapahtumapohjaisesta data-arkkitehtuurista Azure-ympäristössä (Microsoft, 2024)

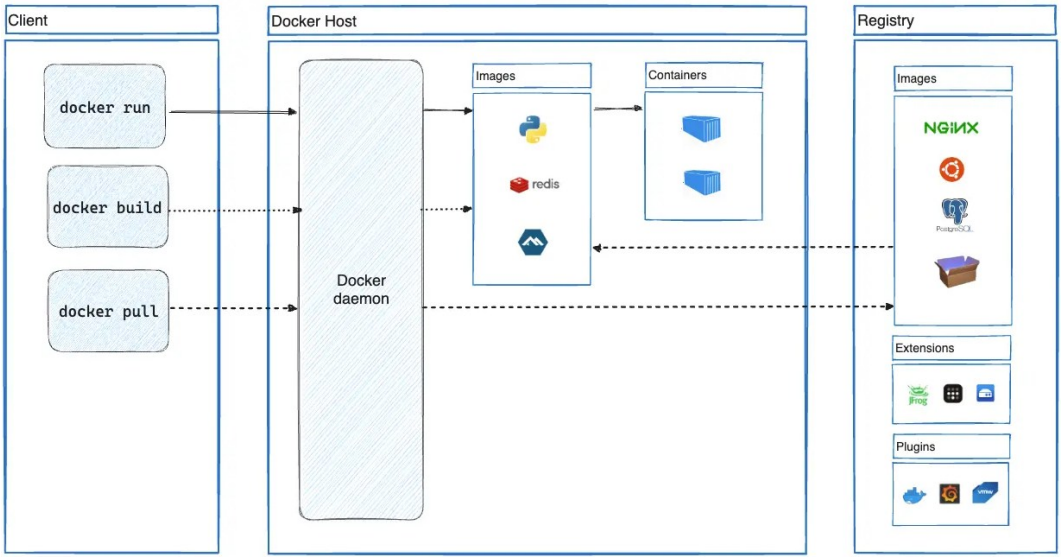
Event Hubilta data kulkee Azure Data Exploreriin Smart Factory -alustan ekosysteemissä. Kyseessä on pilvipalvelu, jonka avulla voidaan ottaa vastaan jäsenneltyä, puolijäsenneltyä ja jäsentämätöntä dataa eri lähteistä. Yksi hyvin yleinen käyttötarkoitus tällä palvelulla on esimerkiksi massiivisen telemetriadatavirran vastaanottaminen tai kyselyjen tekeminen siihen (Caspi, 2018). Kyseinen palvelu toimii tietynlaisena tietokantana, johon voidaan luoda erilaisia klustereita (cluster). Näihin klustereihin voidaan puolestaan sisällyttää tietokantoja, joihin voidaan tallentaa dataa. Yleisesti, tässä työkalussa käytetään Microsoftin omaa kyselykieltä, Kustoa, joka omaa osittain samoja piirteitä kuin SQL-kyselykieli.

Mirkan tapauksessa edellä mainitut Azuren palvelut ovat laajasti käytössä Smart Factory -alustalla. Azure Arcin avulla voidaan esimerkiksi hallinnoida Edge-laitteiden päivityksiä, monitoroida niiden tilaa reaaliaikaisesti sekä muodostaa SSH-etäyhteyksiä laitteisiin. IoT Hubin kautta voidaan puolestaan hallinnoida laitteille asetettuja moduuleita sekä kerätä jokaiselta laitteelta telemetriaa niiden monitoroimiseksi. Smart Factory -alustan jokaisen moduulin takaa löytyy ohjelmistokontti, joka on tallennettu Azuren konttirekisteriin. Kaikki tehtaan laitteilta saatava data puolestaan tallennetaan Azure Data Exploreriin.

4.3 Docker

Smart Factory -alustalle rakennetut ohjelmistokontit toteutetaan Docker-konttiteknologialla, joka on avoin alusta ohjelmistojen kehittämiseen, ajamiseen ja toimittamiseen. Sen pääperiaate on paketoida ja ajaa sovellusta eristetyssä ympäristössä, jota kutsutaan ohjelmistokontiksi (container). Ohjelmistokontit itsessään ovat kevyitä ja ne sisältävät kaiken, mitä sovellus tarvitsee toimiakseen (Docker, n.d.). Käytännössä ohjelmistokontin sisälle voi paketoida minkä tahansa sovelluksen, ja tällöin sitä ei tarvitse asentaa laitteelle erikseen, vaan se pyörii omassa eristetyssä ympäristössään kuormittamatta varsinaista käyttöjärjestelmää.

Docker-ohjelmistokontin arkkitehtuuri (Kuva 7) perustuu ns. asiakas-palvelin-arkkitehtuuriin (client-server architecture), jossa Docker-asiakas (Docker client) puhuu Docker daemonille. Ne kommunikoivat keskenään REST API:a, UNIX-liitäntää tai verkkoliitäntää vasten. Daemon puolestaan on vastuussa ohjelmistokontin ajamisesta ja rakentamisesta tai sen jakelusta. Kuvassa 7 voidaan huomata, että daemon on loppujen lopuksi se komponentti arkkitehtuurissa, joka rakentaa sovellukset ohjelmistokontteihin ja jakelee ne eteenpäin konttirekisteriin.



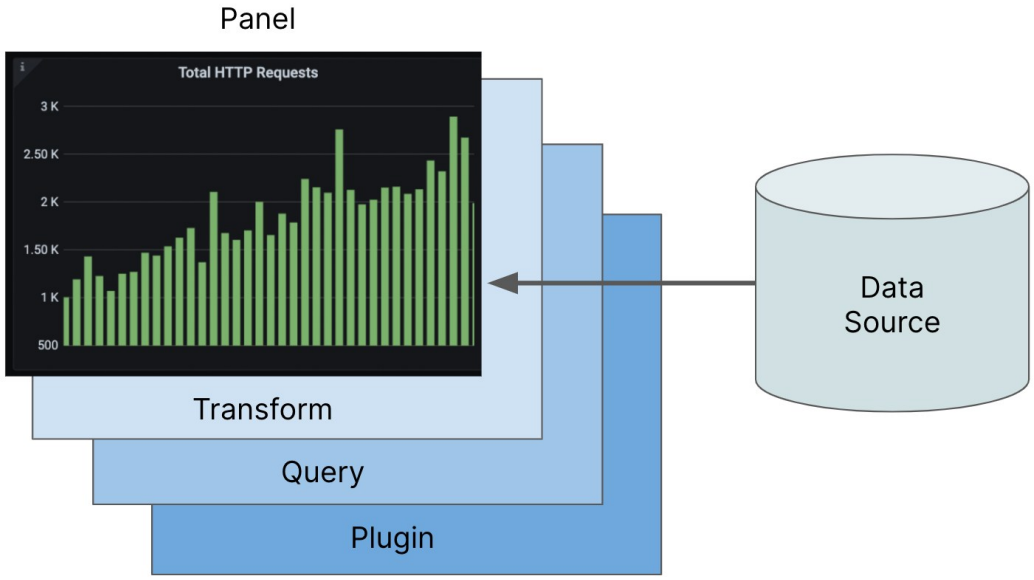
Kuva 7. Dockerin arkkitehtuurikuva (Docker, n.d.)

Kuten aikaisemmassa luvussa todettiin, käytännössä kaikki Mirkan Smart Factory -alustalla olevat moduulit ovat ohjelmistokontteja, jotka on tallennettu Azure Container Registry -konttirekisteriin. Näiden ohjelmistokonttien rakentaminen ja julkaiseminen konttirekisteriin on sidottu vahvasti moduulien pipeline-putkistoihin. Jokaiselle moduulille on luotu ns. Dockerfile-tiedosto, joka määrittelee yksinkertaisella syntaksilla, mitä vaiheita sovelluksen ajamiseen tarvitaan. Näihin voidaan määritellä esimerkiksi, että jotain sovellusta varten avataan tietty portti tai johonkin sovellukseen määritellään erillinen käyttäjä tietyllä nimellä. Tämä tiedosto ajetaan sitten pipelinessa, joka onnistuneen ajon jälkeen julkaistaan Azureen konttirekisteriin käytettäväksi Edge-laitteille.

4.4 Grafana

Datan visualisointiin Smart Factory -alustassa käytetään Azure Managed Grafana -palvelua. Grafana on avoimen lähdekoodin visualisointityökalu, jolla voidaan visualisoida, valvoa ja analysoida metriikoita, lokeja sekä jälkiä tietokannoista saatavasta datasta. Tätä tietokannoista saatavaa dataa voidaan muuntaa erilaisiksi visualisoinneiksi (Grafana, n.d.).

Grafanan pääominaisuus on ns. kojelauta (dashboard), johon käyttäjä voi luoda erilaisia kaavioita visualisoidakseen dataa helposti omaksuttavassa muodossa käyttäjälle. Kojelaudan toimintaperiaate perustuu datalähteeseen, josta saatava data muutetaan kyselyn ja laajennuksen kautta visualisoinniksi paneeliin (Kuva 8). Datalähteenä tällaiseen kojelautaan voi toimia esimerkiksi SQL-tietokanta, JSON-pohjainen API tai jopa CSV-tiedosto. Sen mukaan kysely voidaan toteuttaa sopivalla kielellä, kuten esimerkiksi SQL-kyselykielellä. Datalähteen mukaan laajennus täytyy olla oikeanlainen, koska se ottaa vastaan käyttäjän kyselyn ja hakee sen mukaan datan datalähteestä.



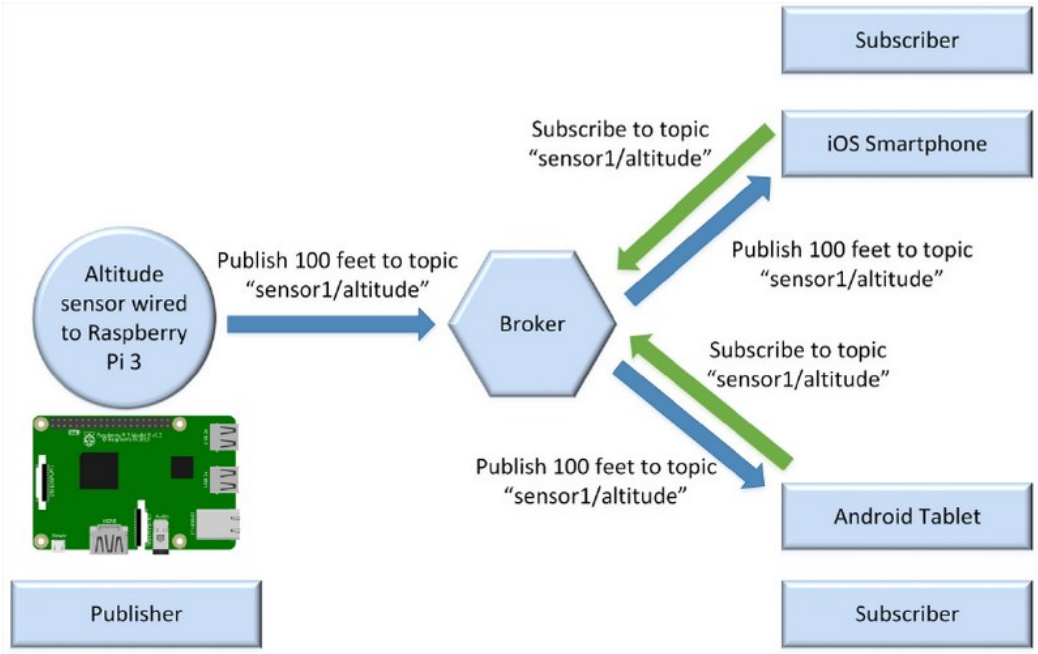
Kuva 8. Grafanan kojelaudan arkkitehtuuri (Grafana, n.d.)

Grafana on aktiivisesti käytössä Mirkalla tehtaiden eri laitteiden datan visualisoinnissa. Datalähteenä tälle saatavalle datalle toimii Azure Data Explorer, johon kaikki data tallennetaan klusteriin ympäristön mukaan. Kysely toteutetaan Microsoftin omalla Kusto-kyselykielellä, jolla saadaan haettua oikeanlainen data visualisoitavaksi Grafanan kojelautaan. Visualisointiin valitaan sopivat kaaviot ja kuviot, jotka sovitaan laitteiden operaattoreiden kanssa yhteistyössä.

4.5 MQTT

MQTT on kevyt viestintäprotokolla, joka toimii välittäjäpohjaisella (broker-based) julkaisija-tilaaja -mallilla (publish-subscribe), joka toimii TCP/IP-protokollan (Transmission Control Protocol/Internet Protocol) päällä. Tämän protokollan avulla voidaan lähettää ja vastaanottaa dataa reaaliajassa, jonka ansiosta tämä protokolla on suosittu varsinkin IoT- ja M2M-ympäristöissä (Machine-To-Machine) sekä sulautettuihin järjestelmiin perustuvissa ympäristöissä (Hillar, 2017, s. 9–10).

Aiemmin mainittu julkaisija-tilaaja-malli MQTT:lla toimii niin, että julkaisija eli tässä tapauksessa laite, lähettää viestin ns. aiheeseen (topic). Tässä vaiheessa viestillä ei ole siis vastaanottajaa, vaan tilaaja voi tilata viestin haluamastaan aiheesta, jolloin viesti tulee perille. Välittäjä (broker) toimii tässä kuviossa viestien keskusasemana, joka välittää aiheita eteenpäin niitä tilaaville tilaajille (Kuva 9). Aiheiden ansiosta tämä malli mahdollistaa joustavan ja tehokkaan tiedonsiirron eri laitteiden välillä. Tilaaja voi siis tilata ainoastaan sille olennaiset ja tarvittavat aiheet, jolloin resursseja säästyy sekä dataa lähettävältä että vastaanottavalta laitteelta.



Kuva 9. Mallikaavio julkaisija-tilaaja-mallin toiminnasta (Hillar, 2017, s. 12).

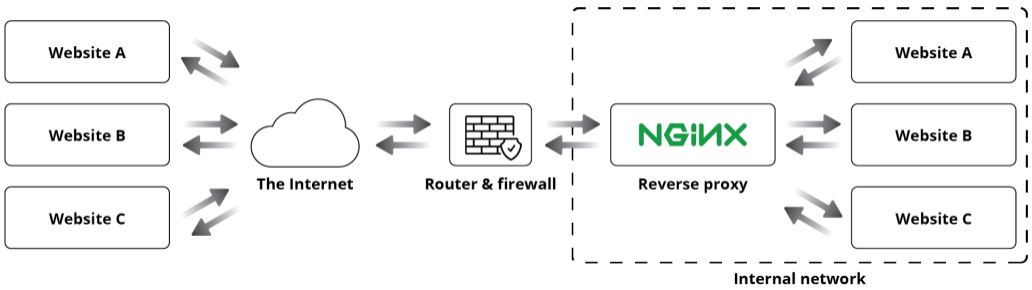
Tällä hetkellä MQTT on käytössä Mirkalla kaikilla Smart Factory -alustaan kytketyillä Edge-laitteella, joilla kerätään valmistuslaitteilta aktiivisesti dataa. Kerättävä data saadaan pääosin PLC-kontrollereilta ja Ruuvi-sensoreilta reaaliaikaisena, ja jokaiselle anturille luodaan oma aihe kuvaavalla tavalla. Aiheen rakenne voidaan määritellä esimerkiksi organisaation nimen, tehtaan sijainnin, laitteen sijainnin, laitteen nimen ja anturin nimen perusteella (esim. Mirka/JE/Hall1/Machine/Temperature_1[C]). Tilatut aiheet julkaistaan lopuksi Azure Data Exploreriin tietokantaan kustomoidulla moduulilla. Tietokantaan tallennettuja arvoja voidaan lopulta hyödyntää esimerkiksi visualisoinnissa.

4.6 NGINX

Yksi visualisointiin liittyvistä tärkeimmistä teknologioista Mirkan Smart Factory -alustalla on NGINX-verkkopalvelin. NGINX toimii HTTP-verkkopalvelimena, mutta sitä voidaan käyttää myös ns. käänteisenä välityspalvelimena (reverse proxy) ja kuormantasaimena (load balancer) eri-

laisiin käyttötarkoituksiin (Solo.io, n.d.). NGINX:ää voidaan myös käyttää tavallisena välityspalvelimena (forward proxy), jossa se välittää asiakaspyyntöä alkuperäiselle palvelimelle. Tämä mahdollistaa asiakkaiden IP-osoitteiden piilottamisen ja kyseistä palvelinta voidaan käyttää esimerkiksi eristetyissä sisäisissä verkoissa, kun vain tiettyihin internet-resursseihin tarvitsee päästä käsiksi (Baeldung, 2025).

Mirkan ympäristössä NGINX:ää käytetään käänteisenä välityspalvelimena ohjelmistokontissa esimerkiksi Grafanan visualisoinnin välittämisessä televisioihin tai muihin näyttöihin. Käänteinen välityspalvelin (Kuva 10) itsessään on palvelin, joka vastaanottaa ja ohjaa pyyntöjä oikealle taustapalvelimelle (backend server) (Herlitz, n.d.). Kun taustapalvelin on onnistuneesti vastaanottanut pyynnön käänteisen välityspalvelimen kautta, palauttaa se vastauksen takaisin sitä pyytäneelle asiakkaalle välityspalvelimen kautta.



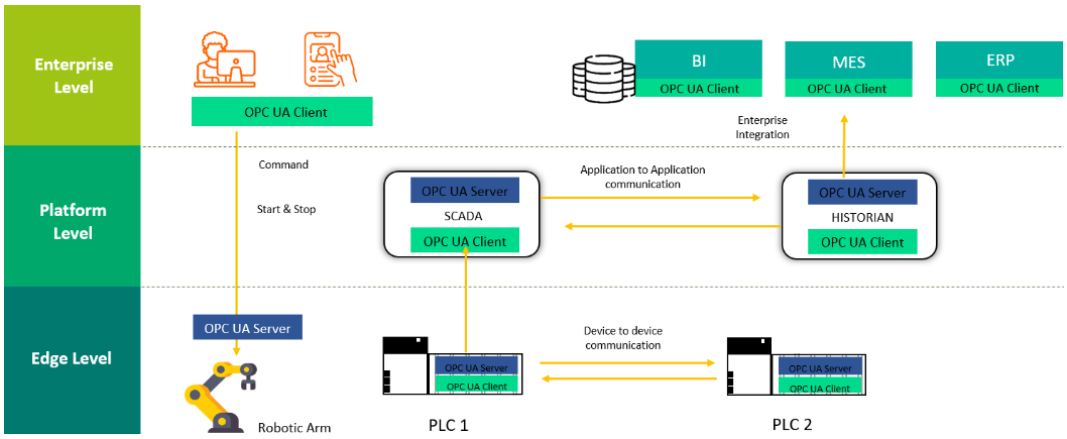
Kuva 10. Mallikuva NGINX:n käänteisestä välityspalvelimesta (Herlitz, n.d.)

Mirkan käyttötapauksessa Azure DevOpsiin on määritelty Edge-laitteen muuttujiin HTTP-osoite, joka on määritelty esimerkiksi valmistuslaitteen nimeksi. NGINX-palvelin ottaa pyynnön vastaan Edge-laitteelta ja välittää pyynnön eteenpäin oikealle Grafana-taustapalvelimelle oikeaan kojelautaan lisäten samalla oikean API-avaimen pyyntöön mukaan. Näin varsinaista osoitetta ei paljasteta ulospäin missään vaiheessa ja API-avaimia ei paljastu systeemin ulkopuolelle.

4.7 Kommunikointiprotokollastandardit OPC-UA ja S7

Monet valmistuslaitteet sisältävät PLC-kontrollereita, joilta on mahdollista kerätä dataa reaaliaikaisesti. Jotta datan keruu olisi mahdollista, tarvitaan siihen oikeanlaiset kommunikointiprotokollat, jotta dataa saadaan PLC-kontrollereilta prosessoitua eteenpäin. Mirkalla on käytössä OPC-UA- ja S7-kommunikointiprotokollastandardit.

OPC-UA on OPC Foundationin kehittämä teollisuuden viestintästandardi, jolla dataa ja informaatiota voidaan siirtää tuotantotasolta yritystasolle. Kyseinen standardi on monialustainen, joka helpottaa IT-integraatiota ja mahdollistaa kompleksisien tietojen mallintamisen. Se on kehitetty korvaamaan vanhat fieldbus- ja IP-pohjaiset protokollat. Informaatiota voidaan kuljettaa jokaisella, aiemmin mainitun ISA-95-mallin pyramidin tasoilla. Tämä tekee pyramidista tarpeettoman, koska tietoa ja dataa voidaan kuljettaa suoraan pyramidin ylemmille tasoille (Ladegourdie ja Kua, 2022, s. 508–511). Tämän voi huomata alla olevasta mallikuvasta (Kuva 11).



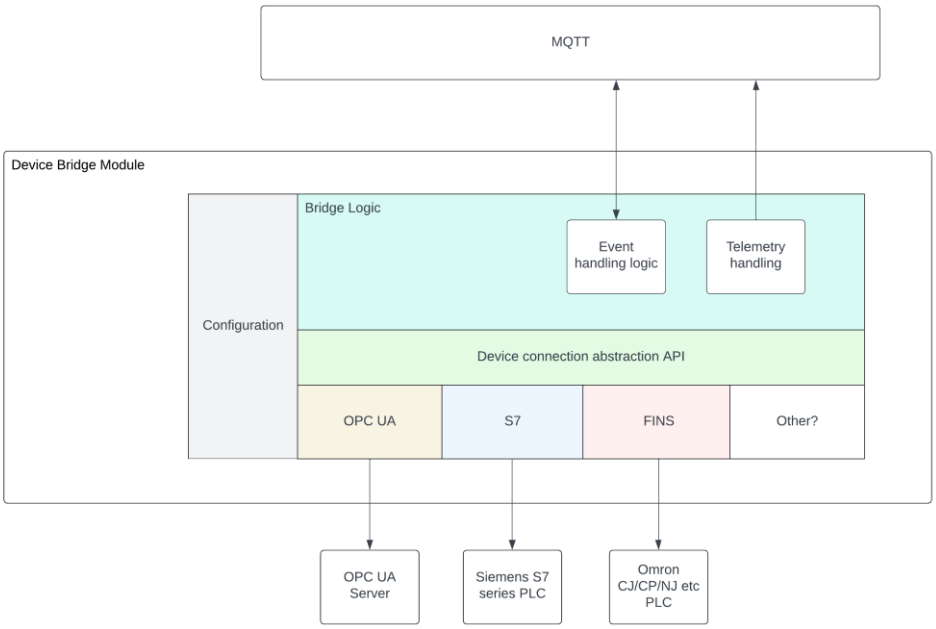
Kuva 11. Mallikuva digitaalisesta transformaatiosta OPC-UA:ta hyödyntäen (Ladegourdie & Kua, 2022, s. 511).

Mirkan Smart Factory -alustalla OPC UA -protokollaa käytetään pääosin PLC-kontrollereilta saatavan datan keräämiseen. Kerätty data on ylei-

sesti erilaista telemetriaa, mutta lisäksi myös tapahtumapohjaista dataa, jota voidaan mahdollisuuksien mukaan kuljettaa esimerkiksi Mirkan omaan ERP-järjestelmään käytettäväksi.

Toinen käytössä oleva kommunikointiprotokolla on S7-protokolla, joka on Siemensin omistama protokolla. Kyseistä protokollaa käytetään PLC-ohjelmointiin ja datan siirtoon PLC-kontrollereiden välillä (Heisenware, n.d.). Samaan tapaan kuin OPC UA -protokollalla, tällä protokollalla voidaan lukea telemetriadataa tai tapahtumia valmistuslaitteilta.

Kumpikin edellä mainituista kommunikointiprotokollista on sisällytetty Device Bridge -nimiseen moduuliin (Kuva 12), joka toimii ns. siltana näiden protokollien ja MQTT:n välillä. Tämä moduuli ottaa vastaan telemetriadataa tai tapahtumia, jotka käsitellään moduulin sisällä niin, että ne voidaan lähettää eteenpäin MQTT:lle oikeassa muodossa. Lopulta, MQTT:lta data voidaan kuljettaa Azureen Data Exploreriin tai Mirkan ERP-iin käytettäväksi.



Kuva 12. Device Bridge -moduulin arkkitehtuurikuva (Mirka, 2024).

4.8 .NET 8.0

Ohjelmistokehys .NET on Microsoftin kehittämä avoimen lähdekoodin kehitysalusta, joka aiemmin tunnettiin nimellä .NET Core. Se tukee monialustaista ohjelmistokehitystä usealle eri käyttöjärjestelmälle (esim. Linux, macOS ja Windows) ja suoritinarkkitehtuureille. Kehitysalustan ekosysteemiin lukeutuu myös .NET Framework, jolla voidaan kehittää työpöytäsovelluksia, selaimella käytettäviä sovelluksia ja palveluita Windowsille. C# on pääasiallinen .NET-ympäristön ohjelmointikieli (Microsoft, 2024).

Alusta koostuu useasta eri komponentista. .NET-ympäristöön sisältyy oma kääntäjä, jonka avulla lähdekoodia voidaan kääntää ajettavaan muotoon. Ajettava koodi puolestaan suoritetaan erillisessä ajoympäristössä (runtime), jota kutsutaan nimellä CLR (Common Language Runtime). Ohjelmien rakentamiseksi .NET-ympäristössä käytetään laajasti työkaluja kuten komentorivityökaluja ja laajan määrän kehittämiseen

käytettäviä erilaisia kirjastoja. Tätä työkalukokoelmaa kutsutaan .NET SDK:ksi (Software Development Kit) (Microsoft, 2024).

Lähes kaikki Smart Factory -alustalle kehitetyt moduulit ovat kehitetty .NET 8.0-ohjelmistokehyksellä. Kyseinen versio tarjoaa tuen ARM64-arkkitehtuurille, joten siitä on merkittävästi apua tässä kehitystyössä. Moduuleja ajetaan Edge-laitteilla kontitetuissa Docker-ohjelmistokonteissa.

5 TYÖN TOTEUTUS

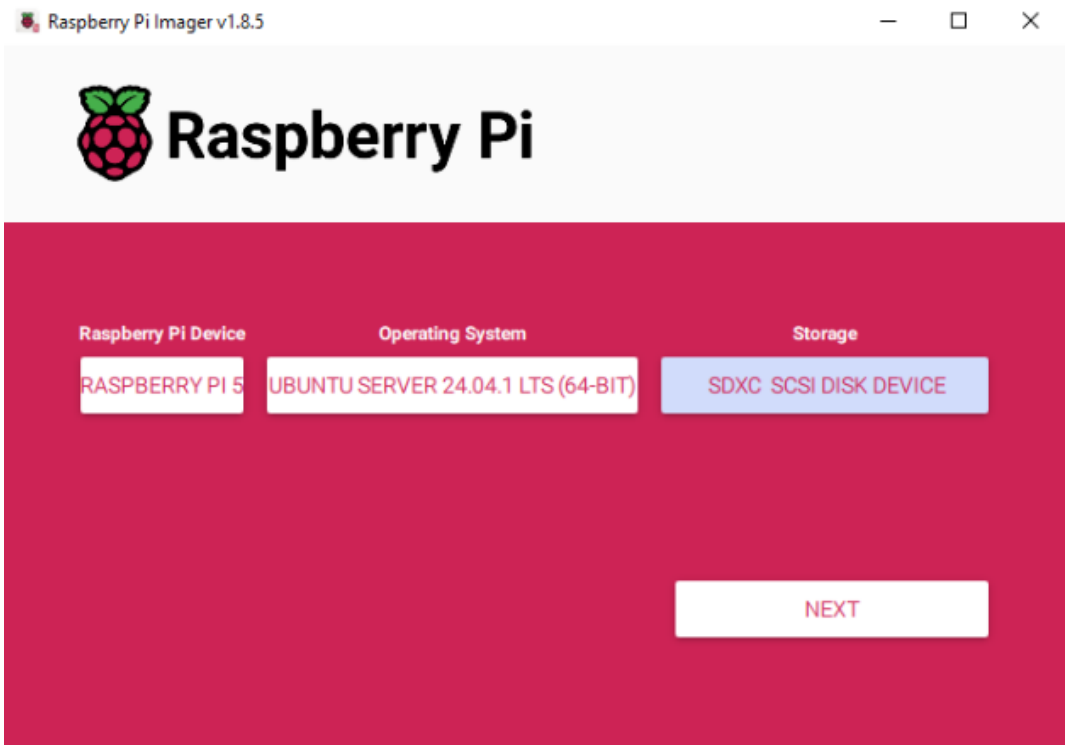
Smart Factory -alustan nykyinen ratkaisu on kehitetty tukemaan x86-arkkitehtuurin laitteita, jota pyritään tässä työssä laajentamaan tukemaan ARM64-arkkitehtuuria. Työn toteutuksessa käytetään ARM64-arkkitehtuuriin pohjautuvaa viidennen sukupolven Raspberry Pi 5 -tietokonetta. Kyseessä on kevyeen laskentaan soveltuva, piirilevyn kokoinen tietokone, jota voidaan käyttää kevyissä käyttötarkoituksissa kuten web-palvelimena tai kotiautomaation ohjauksen apuna. Toteutuksessa tullaan selvittämään, kykeneekö Raspberry Pi toimimaan visualisointiin soveltuvana laitteena tehdasympäristössä.

5.1 Automatisoidun asennusprosessin laajentaminen ARM64-arkkitehtuurille

Smart Factory -alustalla on käytössä automatisoitu asennusprosessi uusia Edge-laitteita varten. Nykyinen asennusprosessi on rakennettu x86-arkkitehtuurilla toimivia laitteita varten. Tätä asennusprosessia on tarkoitus laajentaa tukemaan olemassa olevan arkkitehtuurin lisäksi myös ARM64-arkkitehtuuria.

Nykyisessä asennusprosessissa Azure DevOps:ssa ajetaan *Edge Installer* -niminen pipeline, joka luo uudelle laitteelle Azureen resurssit (IoT Hub, Key Vault, Azure Arc) käyttäjän valitsemaan toimintaympäristöön. Kyseinen pipeline luo myös kustomoidut konfiguraatiotiedostot (*user-data* ja *meta-data*) uuden laitteen asennusta varten, jotka kirjoitetaan ISO-levykuvaksi muutettuun USB-massamuistiin. Lisäksi pipeline alustaa laitteelle konfiguraation, jossa määritellään muun muassa sillä käytettävät moduulit. Asennusta varten tarvitaan myös toinen USB-massamuisti, jossa on itse käyttöjärjestelmä, Ubuntu Server.

Asennusprosessin laajennustyö aloitettiin kartoittamalla mahdolliset eroavaisuudet arkkitehtuurien välillä. Kävi ilmi, että Raspberry Pi ei tue ISO-levykuvaan perustuvaa asennusta, sillä se hyödyntää eri käynnistysjärjestelmää ja levyrakennetta. Pi lataa käynnistystiedostot suoraan FAT32-muotoiselta SD-kortilta, eikä tue x86:lle tyypillistä käynnistystä ISO-levykuvasta. Tämän vuoksi asennus täytyi toteuttaa *Raspberry Pi Imager* -työkalulla, jolla voidaan luoda yhteensopiva käyttöjärjestelmäkuva suoraan SD-kortille (Kuva 13).



Kuva 13. Raspberry Pi Imager

Lisäksi havaittiin, että x86-arkkitehtuurilla käytetty *autoinstall*-menetelmä ei toimi ARM64-ympäristössä. Raspberry Pi käyttää sen sijaan *cloud-init*-pohjaista konfiguraatiota, joka suoritetaan laitteen ensimmäisellä käynnistyskerralla. *Cloud-init* on yleisesti käytetty standardi pilvipalveluissa, ja se mahdollistaa konfiguraation automatisoinnin suoraan käyttöjärjestelmän mukana (Canonical, n.d.). Tämän vuoksi myös user-data-tiedoston rakenne on poikkeava. Alkuperäisen konfiguraatiotiedos-

ton autoinstall-, late-commands- ja interactive-sections-osuudet voidaan jättää pois Raspberry Pi:llä, jolloin tiedostosta tulee yksinkertaisempi ja se saadaan ajettua onnistuneesti uudelle laitteelle.

Koska ISO-artifakti ei toimi Raspberry Pi:llä sellaisenaan, päätettiin konfiguraatiotiedostot muuttaa (refactor) arkkitehtuurikohtaisiksi. Molemmille arkkitehtuureille luotiin omat *user-data*-tiedostonsa, mutta yhteiset työvaiheet keskitettiin yhteen erilliseen *setup.sh*-asennusskriptiin. Näin on mahdollista välttää toisteisuutta koodissa sekä ylläpito helpottuu mahdollisia korjauksia tai uusia arkkitehtuureja varten. Asennuksen aikana PC-ympäristössä skripti kopioidaan väliaikaiseen hakemistoon ja ajetaan sieltä, kun taas Raspberry Pi:llä skripti suoritetaan

/boot/firmware-hakemistosta symbolisen linkin (symlink) kautta (Kuva 14).

```
#cloud-config
hostname: <hostname>
keyboard:
  layout: fi
users:
- name: <username>
  groups: sudo
  lock_passwd: false
  passwd: '<password_hash>'
  ssh_authorized_keys:
  - <ssh_pubkey>

ssh_pwauth: false

package_update: true
package_upgrade: true

timezone: "Etc/UTC"

runcmd:

  - |
    # Create symlink to CIDATA directory
    ln -s /boot/firmware /media/cidata

  - |
    # Ensure setup.sh is executable
    chmod +x /media/cidata/setup.sh

  - |
    # Run installation script
    /media/cidata/setup.sh

  - |
    # Remove CIDATA symlink
    rm /media/cidata
```

Kuva 14. ARM64-arkkitehtuurille rakennettu *user-data* -konfiguraatiotiedosto.

Keskeisimmät erot PC- ja Raspberry Pi -ympäristöjen välillä liittyvät siis asennustavan rakenteeseen (ISO-levykuva vs. SD-kortti), konfiguraatiotiedostojen syntaksiin (*autoinstall* vs. *cloud-init*) sekä skriptien suorittamistapaan. Näitä mukauttamalla on mahdollista tuottaa skaalautuva ja yhtenäinen asennusprosessi molemmille arkkitehtuureille.

5.2 Smart Factory -alustan konttien rakentaminen ARM64-arkkitehtuurille

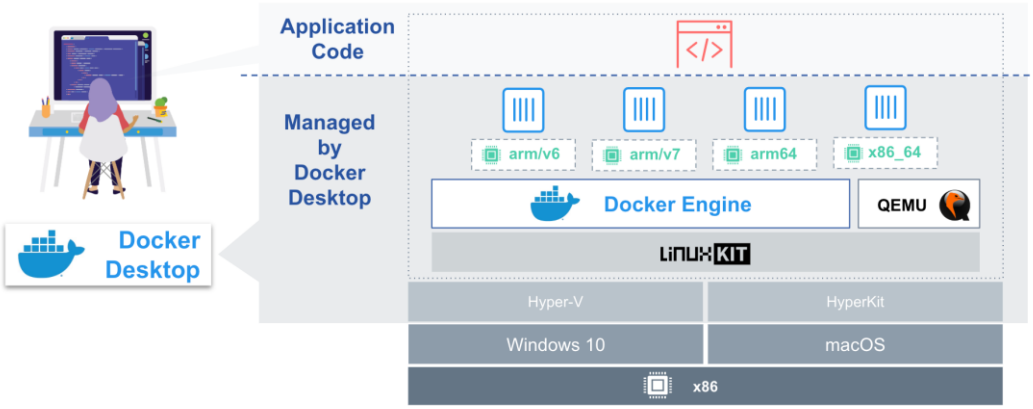
Seuraavassa vaiheessa aloitettiin tutkimaan ja kehittämään ratkaisua, jolla Smart Factory -alustan ohjelmistokontit voitaisiin rakentaa tukemaan x86-arkkitehtuurin lisäksi myös ARM64-arkkitehtuuria. Tällä hetkellä Smart Factory -alustan suurin osa kolmannen osapuolen tarjoamista ohjelmistokonteista on tarjolla useammalle arkkitehtuurille (multi-architecture), mutta tiimin luomat moduulit tukevat pääosin vain x86-arkkitehtuuria.

Kontit ovat toteutettu .NET 8.0 -sovelluskehyksellä. Koska .NET on monialustainen sovelluskehys, ARM64-arkkitehtuuria varten ei ollut tarve tehdä mitään muutoksia moduulien lähdekoodeihin vaan ainoastaan siihen, miten kontti rakennetaan.

Aiemmin moduulien rakentaminen on perustunut pipelinessa käytettävään *Docker@2*-taskiin eli tehtävään, jonka avulla moduulien rakentaminen ohjelmistokonteiksi on onnistunut x86-arkkitehtuurilla. Kyseinen tehtävä myös julkaisee rakennetun ohjelmistokontin Azure Container Registryyn, josta se voidaan poimia käytettäväksi Edge-laitteilla.

Edellä mainitun tehtävän avulla ei voida kuitenkaan rakentaa ohjelmistokontteja usealle arkkitehtuurille, koska monialustainen rakentaminen ei ole tuettu kyseisellä Docker-ajurilla. Tätä varten prosessia täytyy muuttaa käyttämään Dockerin tarjoamaa *buildx*-komentoa. Parcon (2019) mukaan *buildx*-komennolla voidaan rakentaa moniarkkitehtuurisia ohjelmistokuvia (multi-arch image), jotka linkitetään toisiinsa yhdessä manifestitiedostossa ja julkaistaan käyttäjän haluamaan ohjel-

mistokonttirekisteriin. *Buildx*-komento käyttää QEMU-emulaattoria moniarkkitehtuuristen ohjelmistokuvien rakentamiseen, joka sisältyy Docker Desktop -työkaluun (Kuva 15).



Kuva 15. Docker Desktop -ohjelmiston arkkitehtuurirakenne (Parco, 2019)

Ohjelmistokonttien rakentamisen muuttaminen moniarkkitehtuuriseksi tehtiin pipelineen sisällytettävien skriptien (inline script) avulla. *Docker@2*-tehtävä korvattiin *buildx*-komennolla niin, että ensin alustettiin ns. rakentajainstanssi (builder instance), jolle määriteltiin tuettavat alustat tai arkkitehtuurit. Tämän jälkeen luotiin itse rakennusskripti, joka rakentaa moduulista ohjelmistokuvan määritetyille arkkitehtuureille ja julkaisee sen Mirkan ohjelmistokonttirekisteriin, Azure Container Registryyn (Kuva 16). Moniarkkitehtuuristen ohjelmistokonttien toimivuutta testattiin sekä lokaalisti, mutta myös ajamalla moduulien pipelineja kehityshaarasta Azure DevOps:ssa ja ajamalla moduuleja Raspberry Pi:llä.

```
- script: |
    docker buildx create --platform linux/amd64,linux/arm64 --use
  displayName: 'Set up Docker Buildx for multi-platform builds'

- script: |
    docker buildx build \
      --platform linux/amd64,linux/arm64 \
      --tag '${{ parameters.containerRegistry }}/mirka/otplatform/${{ lower(parameters.name) }}:$(Build.BuildId)' \
      --push $(Build.Repository.LocalPath)/Modules/${{ parameters.name }}
  displayName: 'Build and push ${{ parameters.name }} Docker image to ACR'
```

Kuva 16. Mallikuva Smart Factory -alustan ohjelmistokontin rakentamisesta x86- ja ARM64-arkkitehtuureille

5.3 ARM64-laitteen asentaminen ja visualisoinnin alustaminen

Raspberry Pi -laitteen integroimista Smart Factory -alustaan suoritettiin kehitys- ja laajennustyön ohessa. Jokaisen työvaiheen aikana testattiin tiiviisti, kuinka ARM64-laitteen käyttöönotto ja integroiminen Smart Factory -alustaan onnistuu.

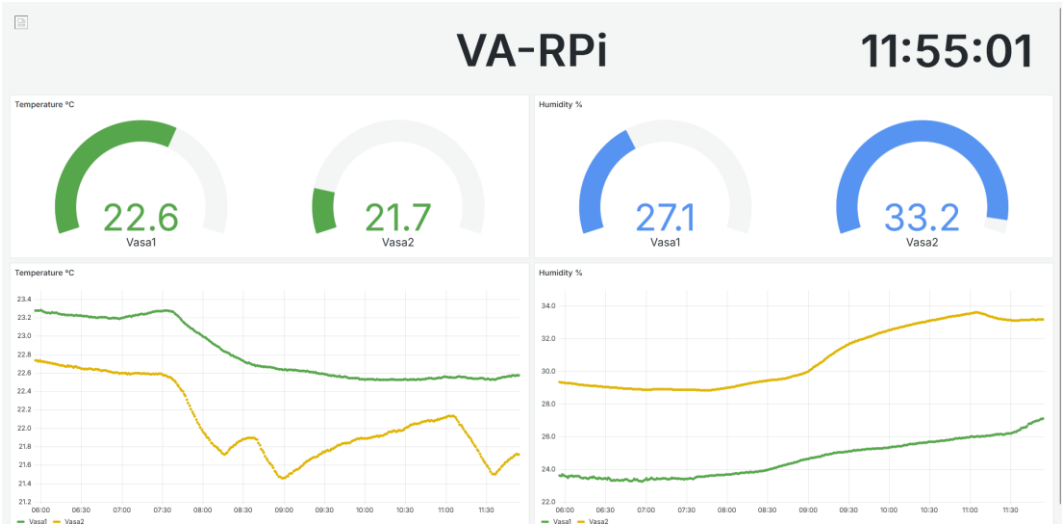
Laajennettua asennnusprosessia jouduttiin testaamaan useaan otteeseen sen kehityksen aikana, mutta lopulta asennus sekä laitteen integroiminen Azuren pilviympäristöön tapahtui saumattomasti. Onnistuneen asennuksen jälkeen Raspberry Pi:lle oli mahdollista saada SSH-yhteys, jonka avulla laitetta on helppo monitoroida sekä hallita tarvittaessa.

Seuraavassa vaiheessa tarkoituksena oli ottaa käyttöön moduuleja, jotka oli rakennettu tukemaan sekä x86- että ARM64-arkkitehtuuria. Aluksi turvallisinta oli ottaa käyttöön kolmannen osapuolen julkaisemia ohjelmistokontteja ja testata niiden toimivuus. Testattavaksi kolmannen osapuolen ohjelmistokontiksi valikoitui Microsoftin *Azure IoT Edge Metrics Collector* -moduuli, joka kerää metriikoita laitteella toimivilta muilta moduuleilta. Nämä metriikat lähetetään eteenpäin Azure Log Analytics -palveluun analysoitavaksi ja sieltä eteenpäin Grafanaan visualisoitavaksi. Kyseisen moduulin käyttöönotto oli onnistunut Raspberry Pi:llä.

Työn tavoitteena oli alustaa visualisointi Raspberry Pi -laitteelle. Tätä varten laitteella täytyi ottaa käyttöön *XOrgChromium*- ja *GrafanaNginx*-

nimiset moduulit. *XOrgChromium* on käytännössä ohjelmistokonttiin kääritty Chromium-selain, joka tarjoaa web-sivujen rendereröintipinon X11-palvelimen ja i3-ikkunamanagerin avulla. Tähän voidaan yhdistää *GrafanaNginx*-moduuli, joka sisältää käyttäjän määrittelemän Grafana-kojelaudan, mikä pyörii Nginx:ssä käänteisenä välityspalvelimena.

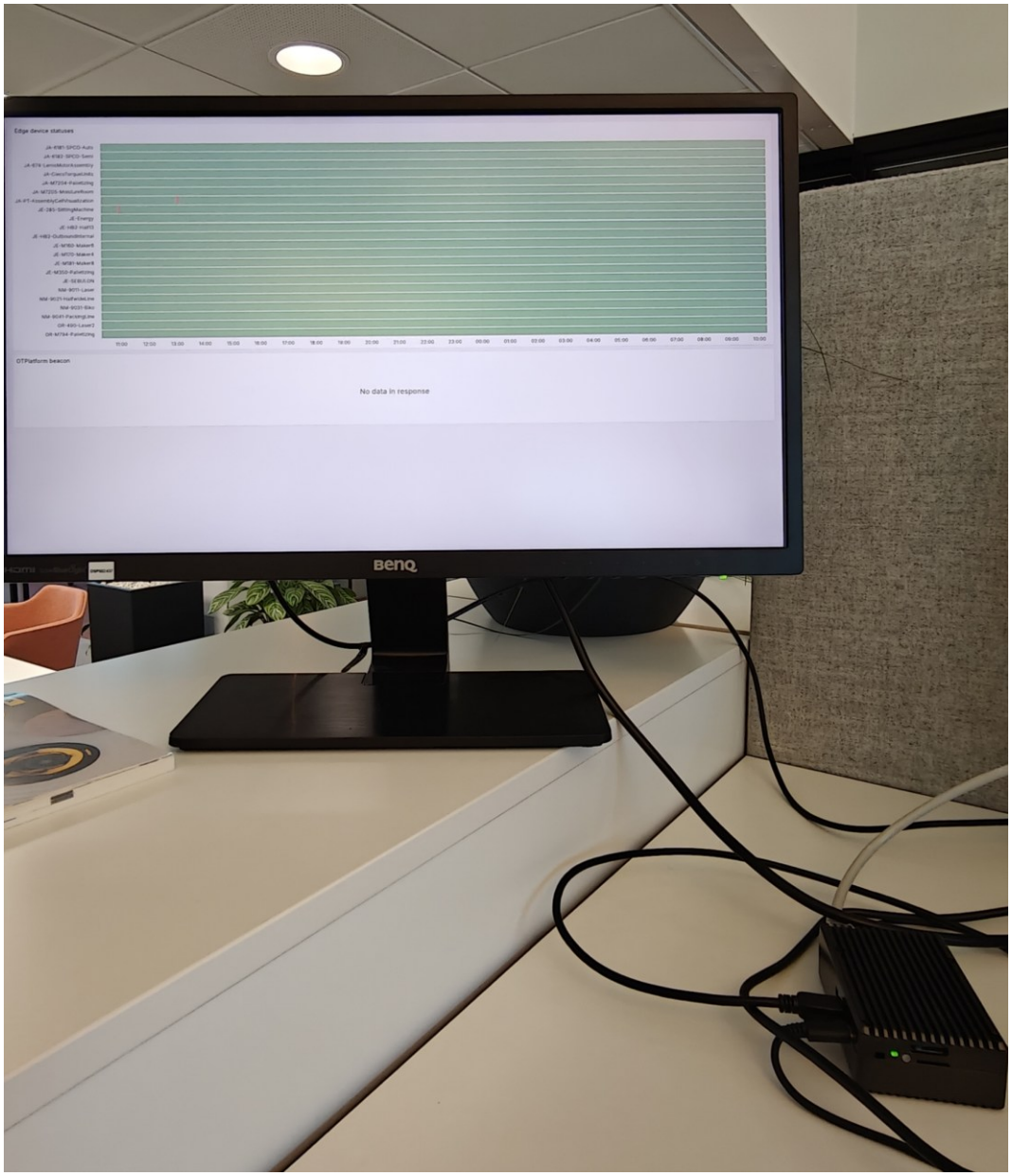
Ohjelmistokonttien rakennusprosessin muutosten jälkeen, *XOrgChromium*- ja *GrafanaNginx*-moduulit oli helppo ottaa käyttöön Raspberry Pi:llä. Alkuun Raspberry Pi:llä oli tarkoitus visualisoida Mirkan Vaasan toimistoon sijoitettujen Ruuvi-sensoreiden avulla lämpötilaa eri huoneissa tai tiloissa (Kuva 17). Tätä varten laitteella täytyi ottaa käyttöön edellä mainittujen moduulien lisäksi myös tiimin kehittämät *RuuviToEdgePublisher*- ja *EdgeToCloudPublisher*-moduulit. *RuuviToEdgePublisher*:in avulla Ruuvi-sensorien data voitiin Ruuvi Gateway -reitittimen avulla siirtää ensin Edge-laitteelle ja tämän jälkeen data siirrettiin *EdgeToCloudPublisher*in avulla Azure Data Exploreriin. Azuresta data puolestaan siirrettiin edelleen visualisoitavaksi Grafanaan.



Kuva 17. *VA-Office*-kojelauta visualisoituna

Jatkuvaan visualisointiin valikoitui lopuksi *Edge Device Statuses* -kojelauta, joka näyttää jokaisen Smart Factory -alustaan integroidun Edge-laitteen tilan viimeisen 24 tunnin ajalta. Visualisointi pyörii *GrafanaN-*

ginx-moduulin avulla *XOrgChromium*-selainmoduulissa ilman työpöytäympäristöä pelkästään näyttäen selaimen koko näytöllä. Tätä kutsutaan ns. kiosk mode -asetukseksi (Kuva 18). Tähän kerättävä data saadaan myös Azure Data Explorerin tietokannasta.



Kuva 18. *Edge Device Statuses* -kojelauta visualisoituna Raspberry Pi -laitteen avulla.

5.4 Toteutuksessa tehdyt havainnot

Laajennus- ja kehitystyön aikana tuli ilmi erilaisia havaintoja, jotka vaikuttivat työn läpivientiin. Pääosin nämä liittyivät moduulien käyttöönottoon mutta myös laitteen päivityskonfiguraatioon.

Alussa *VA-Office* -kojelaudan datan visualisoimisessa tuli ongelmia, kun itse kojelauta näkyi näytöllä, mutta minkäänlaista dataa ei tullut näkyviin. Tämä johtui ongelmasta *EdgeToCloudPublisher*-moduulissa, jossa *himds*-nimiselle (Hybrid Identity Metadata Service) ryhmälle oli määritelty väärä ryhmätunniste (Group ID) verrattuna laitteen järjestelmän ryhmätunnisteeseen.

Kun Edge-laite oli kytketty Azure Arciin, sille syntyy identiteetti, jolle voidaan luvittaa pääsyjä Azuren palveluihin. Tässä tapauksessa *EdgeToCloudPublisher*-moduuli käyttää tätä identiteettiä lähettääkseen dataa Azure Event Hubiin. Väärä *himds*-ryhmätunniste aiheutti sen, että moduuli ei saanut identiteettiä käyttöönsä. Tästä syystä moduuliin tullutta dataa ei saatu liikutettua eteenpäin pilviympäristöön.

Syyksi löydettiin Raspberry Pi:llä käytetty Ubuntu Server 24.04 -käyttöjärjestelmäversio, jossa ryhmätunnisteet oli määritelty uudelleen verrattuna muihin Edge-laitteisiin, joissa oli vanhempi versio 22.04. Ongelma ratkaistiin lisäämällä moduulin konfiguraatioon dynaamisesti määriteltävä ryhmätunniste. Tämä tunniste haetaan Azuresta laitteen instanssista, kun uutta laitetta asennetaan ja moduulin pipeline ajetaan. Sama korjaus täytyi tehdä myös parille muulle moduulille.

Toinen havainto liittyi päivitysongelmiin uudessa Raspberry Pi -laitteessa. Ubuntu Serverin 24.04 -käyttöjärjestelmäversio osoittautui tässäkin tapauksessa haasteelliseksi, sillä Azure Update Manager -päivitystyökalun kautta ajettavat päivitykset epäonnistuivat kerta toisensa jälkeen tällä laitteella. Tämä sama ongelma ilmeni myös muilla Edge-laitteilla, joissa oli samainen käyttöjärjestelmäversio käytössä. Kyseessä on

bugi, joka Microsoftin pitäisi korjata. Kirjoittamishetkellä kyseistä ongelmaa ei ole vielä korjattu.

6 TULOSTEN ANALYSOINTI

6.1 ARM64- ja x86-arkkitehtuurin vertailu

Tässä luvussa vertaillaan ARM64-arkkitehtuurin pohjautuvan Raspberry Pi -laitteen ja Siemensin x86-pohjaisen Simatic Industrial PC:n (IPC) suorituskykyä ja kustannustehokkuutta Smart Factory -ympäristön näkökulmasta. Vertailu keskittyy erityisesti niihin käyttötapauksiin, joissa laitteita hyödynnetään tehdasvisualisointien suorittamiseen sekä kevyiden kontitettujen sovellusten ajamiseen. Lisäksi, molempien tapauksien osalta voidaan arvioida ja vertailla laitteiden luotettavuutta ja komponentteja.

6.1.1 Suorituskyky, luotettavuus ja komponentit

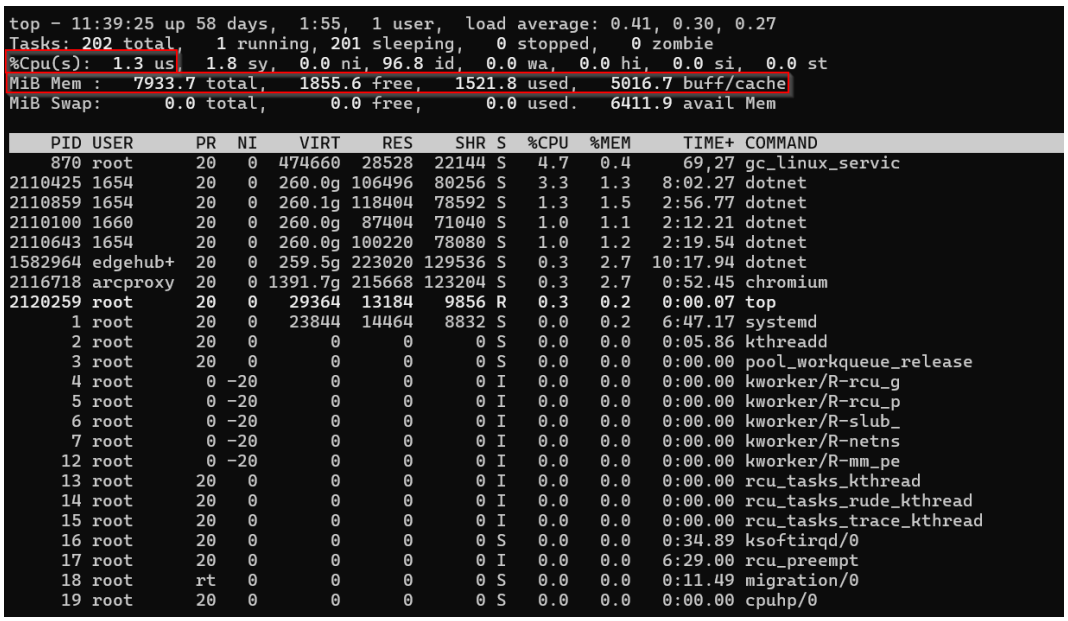
Ulkoisesti katsottuna puhutaan kahdesta hyvin erilaisesta tietokoneesta, mutta laitteiden tekniset spesifikaatiot ovat hyvin lähellä toisiaan. Siemensin Simatic IPC227E sisältää neliytimisen x86-pohjaisen Intelin 1,8 gigahertsisen prosessorin, 8 gigantavun verran DDR3-muotoista RAM-muistia sekä nopean 240 gigatavun SSD-muistin. Lisäksi se sisältää laajan jäähdytyssiilin (Kuva 19). Raspberry Pi puolestaan perustuu kevyempään neliytimiseen 2,4 gigahertsiseen ARM-pohjaiseen Broadcomin prosessoriin, 8 gigatavun DDR4-muotoiseen RAM-muistiin ja 128 gigatavun SD-korttiin, joka toimii laitteen tallennustilana. Laite on koteloitu erilliseen alumiiniseen passiivijäähdytyskoteloon (Kuva 20). Liitäntöjen puolesta molemmat laitteet ovat myös hyvin lähellä toisiaan.

Kuva 19. Siemens Simatic IPC227E tehdas-PC.

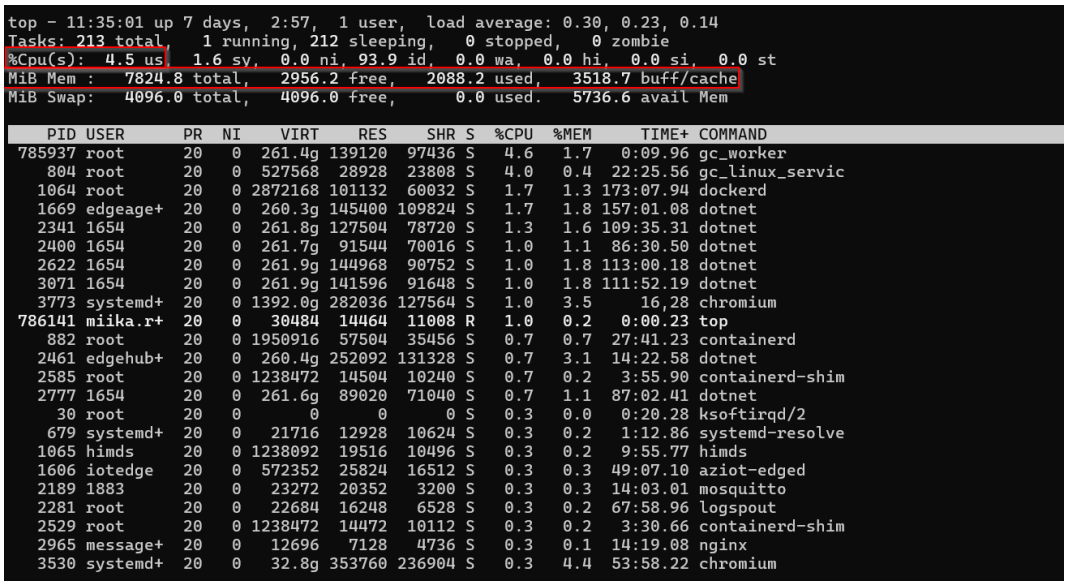
Kuva 20. Raspberry Pi 5 alumiinisessa passiivijäähdytyskotelossa.

Suorituskykyä voidaan mitata näiden laitteiden välillä visualisoinnin avulla. Kun molempiin laitteisiin alustetaan samankaltainen visualisointi, voidaan suorituskykyä vertailla samoista lähtökohdista. Tutkitaan seuraavaksi molempien laitteiden suorituskykyä Linuxin *top*-järjestelmänvalvontatyökalulla, jolla voidaan tutkia prosessorin, keskusmuistin ja prosessien käyttöä. Laitteiden suorituskykyä tutkitaan visualisoinnin ollessa käynnissä molemmilla laitteilla.

Mittauksen tulokset osoittivat, että Raspberry Pi:n (Kuva 21) ja Siemens Simaticin IPC:n (Kuva 22) kuormituksessa ei havaittu merkittäviä eroavaisuuksia. Keskimääräinen prosessorin käyttö oli molemmilla laitteilla hyvin samaa luokkaa, ja keskusmuistin kulutuksessa näiden laitteiden välillä oli hyvin pienet erot. Tulos on yllättävä ottaen huomioon laitteiden tekniset ominaisuudet sekä kohderyhmät, sillä Raspberry Pi on kevyempi ja edullisempi ratkaisu verrattuna Siemensin IPC-laitteeseen.



Kuva 21. Raspberry Pi:n resurssien käyttö top-työkalussa



Kuva 22. Siemens Simatic IPC:n resurssien käyttö top-työkalussa

Lisäksi laitteilla on käytössä eri määrä moduuleja, jotka voidaan ottaa huomioon kuormituksessa. Moduulit ovat kontitettuja sovelluksia, joten niiden sisällä oleva Docker-kerros aiheuttaa pienen lisäkuormituksen järjestelmän eri osa-alueille. Lisäksi täytyy ottaa huomioon, että havainnointihetkellä ajossa olevissa konteissa ei tapahdu intensiivistä prosessoria tai keskusmuistia kuormittavaa laskentaa, mikä voi selittää osaltaan tasaisen kuormituksen laitteilla.

Näiden havaintojen perusteella voidaan todeta, että visualisointi ei aiheuta merkittävää kuormitusta kummallekaan laitteelle. Tämän perusteella Raspberry Pi soveltuu ainakin suorituskyvyn puolesta täysin käyttökelpoiseksi visualisointikäyttöä varten.

Suorituskyvyn lisäksi molempien laitteiden osalta voidaan arvioida niiden luotettavuutta sekä komponenttien laadukkuutta ja riittävyyttä pitkäaikaiseen tehdaskäyttöön. Raspberry Pi käyttää tallennustilanaan SD-korttia, jota käytetään sekä käyttöjärjestelmän että sovellusten tallentamiseen. Vanhemmat SD-kortit ovat varsinkin hyvin haavoittuvaisia ja ne voivat esimerkiksi korruptoitua helposti. Lisäksi SD-kortin suorituskyky ei välttämättä ole riittävä, josta voi aiheutua luotettavuusongelmia

toimintahäiriöiden tai järjestelmän kaatumisten muodossa. Tässä mielessä Siemensin IPC:ssä käytettävä SSD-muisti on selkeästi luotettavampi ratkaisu tallennustilana verrattuna Raspberry Pi:hin.

Jos tarkastellaan ja vertaillaan näiden laitteiden komponentteja, voidaan huomata selkeitä eroja. Raspberry Pi on ilman ostettuja lisävarusteita kuten kotelointia pelkkä piirilevy, jolloin laite voi helposti rikkoutua tai vahingoittua vaativissa tehdasolosuhteissa. Siemens Simatic IPC on puolestaan suunniteltu varta vasten vaativiin tehdasympäristöihin laadukkaiden komponenttiensa ansiosta. Se sisältää vahvan koteloinnin ja jäähdytyksen joka edesauttaa laitteen toimimista tehdaskäytössä.

6.1.2 Kustannukset

Laitteiden kustannukset eroavat paljon toisistaan. Mirkalla käytössä oleva Siemens Simatic IPC227E -tehdas-PC maksaa noin 1535 euroa yksittäisenä kappaleena. Tällä hetkellä Mirkalla on käytössä 30 kappaletta kyseistä laitetta, jolloin saadaan seuraavanlainen laskutulos:

$$1\,535€ * 30 = 46\,050€$$

Vastaavasti tässä työssä käytetty Raspberry Pi 5 koteloineen, SD-kortteineen ja virtalähteineen maksaa yhteensä noin 190 euroa. Lasketaan, kuinka paljon vastaava määrä Raspberry Pi -laitteita kustantaisi:

$$190€ * 30 = 5\,700€$$

Tuloksista voidaan huomata, että puhutaan todella isosta säästöstä, jos kaikki 30 laitetta olisivat Raspberry Pi 5 -laitteita. Kuten aiemmassa luvussa mainittiin, oikeanlaisilla komponenteilla Raspberry Pi olisi huomattavasti kustannustehokkaampi laite tuotantokäytössä pyöritettävää visualisointia varten.

Vihreä siirtymä on keskiössä Mirkan toiminnassa, ja energiatehokkuus on tärkeä osa kestävämpää tuotantoa. Siksi laitteiden sähkönkulutuksella on merkitystä, joten molempien laitteiden osalta voidaan arvioida

ja vertailla myös niiden sähkönkäyttöä ja siitä aiheutuvia kustannuksia. Tyypillinen sähkönkulutus Raspberry Pi:llä noin 5 wattia, kun taas Siemens Simatic IPC:llä se on noin 15 wattia. Lasketaan ensin molempien laitteiden keskimääräinen sähkönkulutus vuodessa.

Raspberry Pi 5:n keskimääräinen sähkönkulutus vuodessa:

$$5\ W * 24\ h * 365\ vrk = 43{,}8\ kWh/vuosi$$

Siemens Simatic IPC227E:n keskimääräinen sähkönkulutus vuodessa:

$$15\ W * 24\ h * 365\ vrk = 131{,}4\ kWh/vuosi$$

Vaihtelevan sähkömarkkinatilanteen ja teollisuudessa käytettävien sähkösopimusten vuoksi tarkkaa sähkön hintaa on vaikea arvioida, mutta voimme käyttää kustannusten laskemiseen kuvitteellista hintaa. Käytetään sähkönkulutuksen vuosikustannusten laskemiseen hintaa 0,20 euroa per kilowattitunti. Lisäksi, laskutoimituksiin voidaan lisätä aiemmin mainittu 30 Edge-laitteen määrä oikeanlaisten kustannuksien laskemiseen.

Raspberry Pi 5:n keskimääräiset vuosittaiset sähkökustannukset 30 laitteella yhteensä:

$$43{,}8\ kWh * 0{,}20\ € * 30 = 262{,}80\ €/vuosi$$

Siemens Simatic IPC227E:n keskimääräiset vuosittaiset sähkökustannukset 30 laitteella yhteensä:

$$131{,}4\ kWh * 0{,}20\ € * 30 = 788{,}40\ €/vuosi$$

Yllä olevien laskutulosten perusteella voidaan todeta, että merkittäviä säästöjä sähkön kulutuksesta ei synny ison teollisuusorganisaation näkökulmasta. Huomattavasti pidemmällä aikavälillä energiasäästöillä on huomattavasti enemmän merkitystä siirryttäessä enemmän kohti vihreää siirtymää.

6.2 ARM64-laitteiden soveltuvuus tehdasympäristöihin

Edellisen luvun perusteella voidaan arvioida ARM64-laitteen soveltuvuutta tehdasympäristöihin. Mitattujen tulosten avulla voidaan todeta, että Raspberry Pi soveltuu varsin hyvin tehdasympäristöön visualisointikäyttöön. Oikeanlaisella koteloinnilla ja komponenttivalinnoilla ARM64-laitteet voivat operoida vaativissakin tehdasolosuhteissa sujuvasti.

ARM64-laitteiden soveltuvuutta tehdasympäristöön voidaan myös arvioida rakennettujen ohjelmistokonttien näkökulmasta. NET-ohjelmistokehityksen mahdollistaessa moniarkkitehtuuristen sovellusten rakentamisen, on sovelluksia vaivatonta rakentaa myös ARM64-arkkitehtuurille sopivaksi. Lisäksi Docker-teknologian tarjoama tuki moniarkkitehtuuristen ohjelmistokonttien rakentamiseen on merkittävä tekijä, jonka avulla ARM64-laitteita voidaan parhaimmillaan soveltaa tehdaskäyttöön digitaalisaation avuksi.

7 JOHTOPÄÄTÖKSET JA YHTEENVETO

Tämän opinnäytetyön tavoitteena oli selvittää ARM64-arkkitehtuuriin pohjautuvien laitteiden soveltuvuutta kevyen laskennan tehtäviin kuten visualisointiin tehdasympäristössä. Työn aikana Smart Factory -alustan uusien Edge-laitteiden asennusprosessia sekä ohjelmistokonttien rakentamista laajennettiin ARM64-arkkitehtuurille. Lopuksi arvioitiin ja vertailtiin ARM64-laitteiden suorituskykyä, luotettavuutta ja kustannustehokkuutta verrattuna olemassa olevaan x86-arkkitehtuuripohjaiseen ratkaisuun.

Saadut tulokset osoittivat, että Raspberry Pi -laite suoriutui hyvin kevyen laskennan tehtävistä ja voitiin myös huomata, että sen suorituskyky oli yllättävän lähellä x86-pohjaisen Siemens Simatic IPC:n tasoa. Lisäksi kustannusvertailussa ARM64-pohjainen ratkaisu osoittautui huomattavasti edullisemmaksi, mikä voisi mahdollistaa kyseisen arkkitehtuurin ratkaisun laajemman käyttöönoton tuotantoympäristöissä. Voidaan siis todeta, että alkuperäisiin tutkimuskysymyksiin saatiin selkeitä vastauksia.

Työ oli sopivan haastava ja mielenkiintoinen. Kehitys- ja laajennustyön aikana vastaan tuli muutamia haasteita ARM64-laitteiden integroimisessa hybridipilviympäristöön sekä ohjelmistokonttien sovittamisessa ARM64-arkkitehtuuriin. Työ vaati paljon muutoksia sekä testaamista jatkuvasti, jotta saatiin aikaan vakaa ja toimiva ratkaisu. Kehitystyön aikana otettiin myös huomioon eettinen näkökulma, sillä työ kehitettiin Mirkan tietoturvamääräysten mukaisesti paljastamatta arkaluontoista tietoa ulospäin missään vaiheessa.

Tämä opinnäytetyö tarjosi minulle loistavan mahdollisuuden oppia uutta ja lisää älykkään tehtaan periaatteista sekä siihen liittyvästä digitalisaatiosta. Lisäksi kehitystyön aikana opin paljon uutta Smart Factory -alustalla käytetyistä teknologioista ja niiden hyödyntämisestä tehdasympäristössä Edge-laitteilla. Vaikka Mirkalla on aikaisemminkin ollut käytössä

Raspberry Pi -laitteita, oli tämä ensimmäinen tapaus Mirkan Smart Factory -alustalla, jossa sen käyttömahdollisuuksia voitiin tutkia. Tämän työn perusteella voidaan todeta, että ARM64-pohjaisia laitteita voidaan ottaa laajemmin käyttöön eri tuotantolaitoksissa tehtyihin arviointeihin perustuen.

LÄHTEET

Baeldung. (2025). *Using Nginx as a Forward Proxy*. Noudettu 11.4.2025 osoitteesta https://www.baeldung.com/nginx-forward-proxy

Canonical Group Ltd. (n.d.). *Cloud-init documentation*. Noudettu 15.4.2025 osoitteesta https://cloudinit.readthedocs.io/en/stable/

Caspi, Z. (2018). *Azure Data Explorer Technology 101. Microsoft Azure.* Noudettu 8.4.2025 osoitteesta https://azure.microsoft.com/en-us/blog/azure-data-explorer-technology-101/

Docker. (n.d.). *What is Docker.* Noudettu 30.3.2025 osoitteesta https://docs.docker.com/get-started/docker-overview/

Fortoul-Diaz, J. A., Carrillo-Martinez, L. A., Centeno-Tellez, A., Cortes-Santacruz, F., Olmos-Pineda, I., & Flores-Quintero, R. R. (2023). *A smart factory architecture based on industry 4.0 technologies: open-source software implementation*. *IEEE Access*, 11, 101728. https://doi.org/10.1109/ACCESS.2023.3316116

Gilchrist, A. (2016). *Industry 4.0*. Apress. http://doi.org/10.1007/978-1-4842-2047-4

Grafana. (n.d.). *Introduction | Grafana documentation.* Noudettu 30.3.2025 osoitteesta https://grafana.com/docs/grafana/latest/fundamentals/

Heisenware. (n.d.). *Siemens S7.* Noudettu 11.4.2025 osoitteesta https://docs.heisenware.com/build-and-deploy-apps/integrations/protocol-connectors/siemens-s7

Herlitz, E. (n.d.). *Running Optimizely CMS behind NGINX Reverse Proxy*. Noudettu 9.4.2025 osoitteesta https://www.herlitz.io/2022/03/running-optimizely-cms-behind-nginx-reverse-proxy/

Hillar, G. C. (2017). *MQTT essentials: A lightweight IoT protocol*. Packt Publishing.

Lachance, P. (2021). *An Introduction to IoT, CMMS and the ISA-95 Framework*. *Brightly*. Noudettu 31.3.2025 osoitteesta https://www.brightlysoftware.com/blog/introduction-iot-cmms-and-isa-95-framework

Ladegourdie, M., & Kua, J. (2022). *Performance Analysis of OPC UA for Industrial Interoperability towards Industry 4.0*. *IoT, 3*(4), 508–511. https://doi.org/10.3390/iot3040027

Microsoft Azure. (n.d.). *Azure Arc – Hybrid and Multi-Cloud Management and Solution*. Noudettu 8.4.2025 osoitteesta https://azure.microsoft.com/en-in/products/azure-arc

Microsoft Azure. (n.d.). *Azure Boards | Microsoft Azure*. Noudettu 5.4.2025 osoitteesta https://azure.microsoft.com/en-gb/products/devops/boards

Microsoft Azure. (n.d.). *Types of Cloud Computing – Definition*. Noudettu 2.4.2025 osoitteesta https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/types-of-cloud-computing

Microsoft Azure. (n.d.). *What is Azure.* Noudettu 8.4.2025 osoitteesta https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure

Microsoft Learn. (2024). *Azure Event Hubs: A real-time data streaming platform with native Apache Kafka support*. Noudettu 11.4.2025 osoitteesta https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about

Microsoft Learn. (2024). *Key concepts for new Azure Pipeline users.* Noudettu 7.4.2025 osoitteesta https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/key-pipelines-concepts?view=azure-devops

Microsoft Learn. (2024). *What is Azure Repos.* Noudettu 7.4.2025 osoitteesta https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops

Microsoft Learn. (2024). *Introduction to .NET*. Noudettu 10.4.2025 osoitteesta https://learn.microsoft.com/en-us/dotnet/core/introduction

Microsoft Learn. (2024). *About registries, repositories, and artifacts.* Noudettu 8.4.2025 osoitteesta https://learn.microsoft.com/en-us/azure/container-registry/container-registry-concepts

Microsoft Learn. (2024). *What is Azure DevOps.* Noudettu 30.3.2025 osoitteesta https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops

Microsoft Learn. (2025). *What is Azure IoT Hub.* Noudettu 8.4.2025 osoitteesta https://learn.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub

Mirka. (n.d.). *Historia*. Noudettu 31.1.2025 osoitteesta https://www.mirka.com/fi-fi/yritys/tietoa-meista/historia/

National Institute of Standards and Technology (NIST). (2018). *Risk management framework for information systems and organizations: A system life cycle approach for security and privacy*. U.S. Department of Commerce. 101. https://doi.org/10.6028/NIST.SP.800-37r2

Parco, A. (2019). *Building Multi-Arch Images for Arm and x86 with Docker Desktop*. Docker. Noudettu 16.4.2025 osoitteesta https://www.docker.com/blog/multi-arch-images/

RedHat. (2023). *What is CI/CD.* Noudettu 20.3.2025 osoitteesta https://www.redhat.com/en/topics/devops/what-is-ci-cd

SAP. (n.d.). *What is a smart factory.* Noudettu 11.4.2025 osoitteesta https://www.sap.com/products/scm/what-is-a-smart-factory.html

Scaled Agile Framework (SAFe). (2025). *Story*. Noudettu 11.4.2025 osoitteesta https://framework.scaledagile.com/story

Scaled Agile Framework (SAFe). (2025). *Features and Capabilities*. Noudettu 11.4.2025 osoitteesta https://framework.scaledagile.com/features-and-capabilities

Scaled Agile Framework (SAFe). (2025). *Epic.* Noudettu 11.4.2025 osoitteesta https://framework.scaledagile.com/epic

Solo.io. (n.d.). *Understanding NGINX: Architecture, Configuration & Alternatives*. Noudettu 9.4.2025 osoitteesta https://www.solo.io/topics/nginx