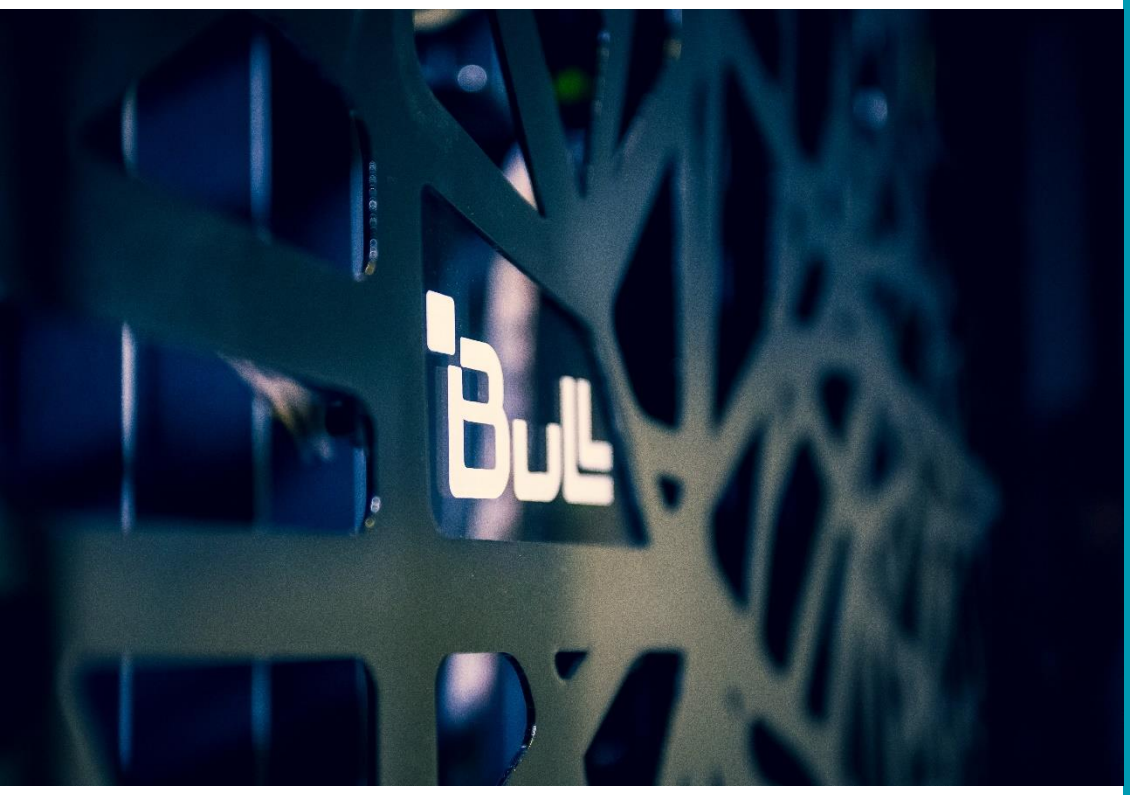
Henri Lehtimäki

Bull-supertietokoneen ylläpidollisen datan kerääminen ja visualisointi



Insinööri

Tieto- ja viestintä tekniikka

Kevät 2020

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Lehtimäki Henri

Työn nimi: Bull-supertietokoneen ylläpidollisen datan kerääminen ja visualisointi

Tutkintonimike: Insinööri tieto- ja viestintätekniikka

Asiasanat: Visualisointi, Tietokanta, Tiedon esittäminen, Graafinen esitys, Aikasarjadata, Modbus RTU/TCP, Supertietokone, Laskentaklusteri, Automaatio, IoT / esineiden internet, Rakennusautomaatio.

Opinnäytetyössä käydään läpi Kajaanin ammattikorkeakoulun Bull-supertietokoneen ylläpidollisen automaation datan keräämistä sekä visualisoimista. Työ on osa pidempään kestävien projektien kokonaisuutta, jossa Bull-supertietokone otetaan käyttöön osaksi KAMK koulutusympäristöjä.

Tavoitteena työssä oli kerätä automaatio- ja ylläpitolaitteista dataa, tallentaa kerätty data aikasarjatietokantaan ja visualisoida data helpommin ymmärrettävään muotoon. Toteutukselta vaadittiin luotettavaa toimintaa ympärivuorokautisessa toiminnassa. Sen tuli palautua mahdollisesta ongelmatilanteesta automatisoidusti sekä toimia luotettavasti. Merkittävä vaatimus sovelluksen toiminnalle asetettiin liittyen automaatiojärjestelmän kanssa kommunikointiin; sovelluksella oli lupa vain lukea tietoja, ei kirjoittaa niitä automaatioon päin. Automaatiojärjestelmän suuntaan kirjoittaminen saattaisi aiheuttaa jäähdytysympäristön parametrien muuttumisen ja pahimmillaan jäähdytysjärjestelmän vikaantumisen.

Vaatimuksiin perustuen opinnäytetyössä suunniteltiin ja toteutettiin sovellus, joka lukee dataa supertietokoneen automaatiojärjestelmästä sekä verkkoanalysaattorilta ja tallentaa datan tietokantaan. Tehtävää varten kehitettiin Python-ohjelmisto, joihin Modbus-protokolla on implementoitu PyModbus-kirjaston avulla. Noudettu data käsitellään ja siirretään Prometheus-aikasarjatietokantaan käyttäen Prometheuksen omaa Pushgateway ratkaisua. Datan visualisointi toteutetaan noutamalla aikasarjadataa Prometheus-tietokannasta Grafanan visualisointiympäristöön.

Tämän järjestelmän tietokanta- ja visualisointipalvelimien ylläpito toteutetaan virtualisoidussa ympäristössä, mutta Modbus-kommunikointiohjelmat ajetaan fyysisellä laitteistolla. Kommunikointiohjelmat suoritetaan Docker-säiliöissä, jotta ohjelmien virhetiloista palautuminen voidaan varmentaa ja eristää muusta järjestelmästä. Suoritusympäristöksi valittiin passiivisella jäähdytyksellä varustettu pienikokoinen tietokone. Siirtymällä passiiviseen jäähdytykseen perustuvaan tietokoneeseen pystyttiin pienentämään mekaanisesti kuluvien osien määrää ja siten parantamaan laitteiston luotettavuutta.

Työssä kehitetty sovellus Bull-supertietokoneen ylläpidollisen datan keräämiseen ja visualisointiin täyttää sille projektin alussa asetetut vaatimukset. Projektin jatkokehityksen ja pitkäaikaisemman tarkkailun osalta projekti jäi vielä kesken, johtuen opinnäytetyön toteutuksen aikataulusta. Näihin jatkokehityssuunnitelmiin kuuluivat visualisointialgoritmien viimeistely sekä mahdollisien hälytyksen aiheuttavien raja-arvojen määritys. Pitkäaikaisemman toiminnan tarkkailun osalta tehtävä siirtyi jatkokehitystiimille.

KAMK • University of Applied Sciences

Abstract

Author(s): Lehtimäki Henri

Title of the Publication: Bull Supercomputers Maintenance Data Collection and Visualization.

Degree Title: Bachelor of Engineering, information and communication technology

Keywords: Prometheus database, Grafana, Python, Docker, Time-series data, Data visualization, Modbus RTU/TCP, Supercomputer, Computation cluster, IoT, Automation data, Building automation

The topic of this Bachelor's thesis was to collect, store, and visualize data from the automation and upkeep systems around the Bull-supercomputer. The thesis project was a part of a larger group of projects which aim to improve and integrate the Bull high-performance computation cluster into part of the educational environment.

The thesis project aims to collect data from the automation system and network-analyzer using the Modbus-protocol. The collected data is stored in the Prometheus time-series database, from which the data is queried into Grafana for visualization purposes. The requirements for the end-product included; stable round-the-clock operation, self-correction from error states, and safety of the automation system requires that the Modbus-communications are only allowed to read and not write.

The cooling automation and maintenance system data is accessed through Modbus protocol, for which purpose a pair of custom Python software was programmed. The software uses PyModbus to implement the Modbus-protocol and Docker to fulfill the error-state correction and stable round-the-clock operation requirements.

While the data-collection software instances are run in Docker containers on a physical system, the Prometheus database and Grafana visualization service are hosted inside KajakDC virtualization environment, this helps with the overall system reliability. The data-collection system runs on a small and energy-efficient with passive cooling. The use of passive cooling increases to the reliability of long-term operation due to the reduced amount of mechanically wearing components.

The end-product of this thesis project is a system capable of collecting data from the automation system and network-analyzer, storing it safely inside a time-series database, and then visualizing it in a graphical form.

Further development plans for this system include; finalizing the visualization algorithms, long-term reliability and stability monitoring, and defining some alarm thresholds. The further development and system monitoring were carried out by the continuing team of developers.

Sisällys

Termiluettelo

Aikasarjadata – Joukko perättäisiltä ajankohdilta olevia datapisteitä, käytetään usein visualisoinnin yhteydessä.

Algoritmi – Sarja ennalta määritettyjä toimenpiteitä, joilla tehtävä tai muokkaus toteutetaan.

Automaatio – Erilaisia useasti toistuvia tehtäviä toteuttava järjestelmä, joka työskentelee ilman ihmisten puuttumista asiaan.

Blade – Korttipalvelinkehikko, joihin voidaan integroida palvelimen lisäksi laskenta, tallennus- ja tietoliikennelaitteistoa.

Chassis – Datacenterympäristössä käytettävä Blade-muotoisille tietokoneille kehitetty kehikko. Sisältävät monesti ominaisuuksia, jotka mahdollistavat Blade-tietokoneen nopean vaihtamisen ongelmatilanteen tapahtuessa.

Dashboard – Grafana- verkkosivu, jolla voidaan esittää useita erilaisia graafeja nähtäväksi yhdestä sijainnista.

Datapiste – Tietokannassa sijaitseva arvo, aikasarjadatan osalta sisältää myös kirjaamisen yhteydessä merkityn aikaleiman.

Docker – Tuoteperhe, jolla voidaan helposti ylläpitää säiliöityjen ohjelmien ja järjestelmien toimintaa.

Grafana – Datan visualisointityökalu.

HPC – (High Performance Computing) termillä viitataan korkeateholaskentaan, tällaisessa laskennassa käytetään usein supertietokoneita tai laskentaklustereita.

Infrastruktuuri –Toimintojen ja järjestelmien muodostama kokonaisuus, jonka olemassaolo on välttämätön loppukäyttäjän osalta.

IoT – Esineiden internet.

Jäähdytyslohko – Jäähdytysjärjestelmän osa, jonka avulla siirretään lämpöä pois sitä tuottavilta laitteiston osilta.

Laskentaklusteri – Monen erillisen tietokoneen yhdessä muodostama järjestelmä, jolla voidaan toteuttaa monimutkaisia laskentoja.

Laskentakortti – Tietokoneeseen liitettävä lisäkortti, jolla voidaan suorittaa erikoistuvaa laskentaa. Näihin lukeutuvat näytönohjaimet.

Modbus – Sarjaliikenneprotokolla, jolla kommunikoidaan isäntä- ja orjalaitteiden välillä.

Monoblock – Yksittäisestä kappaleesta koostuva jäähdytyslohko, joka kattaa tietokoneen emolevyn ja prosessorien jäähdytyksen.

Parametri – Muuttujan arvo, jonka avulla ohjelmia tai järjestelmiä voidaan konfiguroida.

Prometheus – Aikasarjadatan hallintaan erityisesti suunniteltu tietokantatyökalu

PyModbus – Python-kirjasto, joka mahdollistaa Modbus-protokollan integroinnin ohjelmaan.

Python – Skriptipohjainen ohjelmointikieli.

Rack – Tukirakenne, johon voidaan tukevasti ja suojatusti kiinnittää elektronisia laitteita, kuten palvelimia, virtasuojajärjestelmiä ja reitittimiä.

RS485, RS422, RS232 – Modbus-protokollan käyttämiä sarjamuotoisen tiedonsiirron standardeja.

Supertietokone –Suurteholaskennassa käytettävä tietokone, supertietokoneet koostuvat useasti monesta Blade-tietokoneesta yhdistetyssä rakenteessa.

TCP – Transmission Control Protocol, yhteydellinen tiedonsiirtoprotokolla.

UDP – User Datagram Protocol, yhteydetön tiedonsiirtoprotokolla.

Tietokanta – Kokoelma tietoja, joilla on yhteys toisiinsa.

Visualisointi –Datan esittäminen graafisessa muodossa.

1 Johdanto

Kajaanin ammattikorkeakoulu, KAMK, on Kajaanissa sijaitseva ammattikorkeakoulu, joka tarjoaa mahdollisuuksia kouluttautua monien eri ammattialojen huippuosaajaksi. KAMK mahdollistaa opiskelun muiden muassa datacenter ympäristöjen, ohjelmistokehittämisen ja älykkäiden järjestelmien aloilla. Nämä osaamisen alueet ovat hyvin käytännönläheisiä ja niiden opiskelua KAMK tukee työelämän ja projektit yhdistävällä lähestymistavalla.

Opiskelijoilla on KAMK:n ympäristössä mahdollista tutustua ja hyödyntää erikoisempia laitteistoja kuten Bull-supertietokone. Monia laitteistoja ja ratkaisuja käytetään ja toteutetaan yhdessä yhteistyökumppaneiden kuten CSC kanssa, jonka kautta tässä opinnäytetyössä esiintyvä Bull-supertietokone on saatu.

CSC eli Tieteen tietotekniikan keskus Oy on Suomessa sijaitseva korkeateholaskennan palveluita tarjoava yritys. CSC on perustettu jo vuonna 1971 ja se on toiminut yhteistyössä valtion, terveysalojen, tieteen ja mittaustekniikan yrityksien kanssa. CSC myös ylläpitää ja hallinnoi FUNET-verkkoinfrastruktuuria, joka yhdistää monia Suomen koulutus- ja tutkimuslaitoksia sekä valtion toimipisteitä toisiinsa.

Bull-supertietokone on alun perin CSC:n hankkima todella energiatehokas korkeatehoinen laskentaklusteri, joka vuoden 2018 aikana siirrettiin ja otettiin käyttöön Kajaanin ammattikorkeakoulun tiloissa. KAMK:ssa toimiva Kajak DC on vastannut laitteiston ja ohjelmistojen ylläpidosta vuodesta 2019 alkaen, ylläpidon ohella on toteutettu jatkokehitysprojekteja, joihin myös tämä opinnäytetyö kuuluu. [1-5.]

Opinnäytetyö kuuluu Bull-supertietokonetta koskevaan projektikokonaisuuteen ja toimeksiantajana toimi KAMK. Toimeksiantajan puolelta yhteyshenkilönä opinnäytetyössä toimi järjestelmäasiantuntija Jukka Jurvansuu.

2 Työn tausta

Tämä luku esittelee työn taustatietoja, jotka helpottavat opinnäytetyön myöhempien osuuksien kontekstin ymmärtämistä. Tulevissa luvuissa käsitellään pohjatietoja opinnäytetyön toteutusympäristöstä. Luku aloittaa esittelemällä isomman kokonaisuuden, jonka ympärillä olevaan infrastruktuuriin opinnäytetyö sijoittuu.

2.1 HPC-ympäristöt

Terminä HPC (High Performance Computing) tarkoittaa korkeateholaskentaa, jossa suuria määriä dataa ja monimutkaisia laskukaavoja ratkotaan tehokkaiden tietokone ratkaisuiden avulla kuten supertietokoneilla. Käytännössä HPC-ympäristöjä hyödynnetään isojen matemaattisten ongelmien laskemiseen, esimerkiksi fysiikan teorioiden, sairauksien, luonnonilmiöiden, ilmaston ja sään tutkimiseen. HPC-ympäristöt hyödyntävät useasti teknologiaratkaisuja, joita ei ole kuluttajilla käytettävissä. Näistä tyypillisiä esimerkkejä ovat klusteripohjainen ohjelmointi sekä monet erilaiset laitteistoratkaisut. Järjestelmien yhteyteen kehitetään yleensä myös niiden tuottamaa hukkalämpöä hyödyntäviä ratkaisuja, kuten esimerkiksi toimistotilojen lämmittäminen.

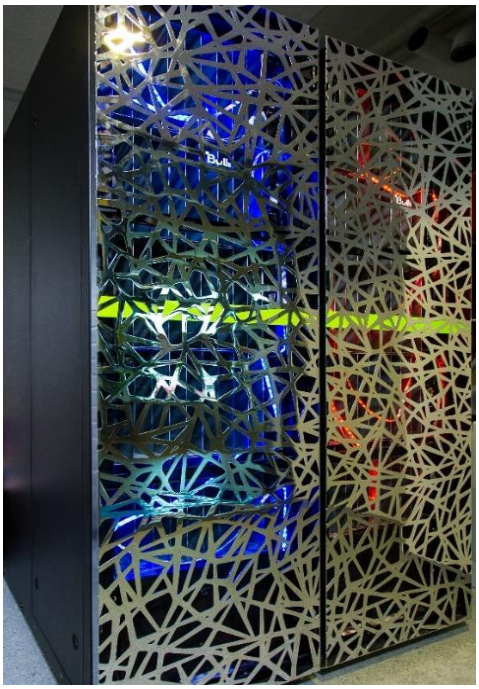
2.2 Bull-laskentaklusteri

Bull-laskentaklusteri on Kajaanin ammattikorkeakoulun tiloissa sijaitseva korkeateholaskentaklusteri. Klusteri on saanut nimensä laitteiston valmistajan Groupe Bullin mukaan, joka siirtyi vuonna 2014 Atos Oy:n omistukseen [5]. Korkeateholaskentaklustereista käytetään myös yleisnimitystä supertietokone. Bull-laskentaklusteri sijaitsi ensin CSC:n tiloissa Kajaanissa, josta klusteri siirtyi KAMKin tiloihin opiskelijoiden ja yhteistyökumppaneiden käytettäväksi vuonna 2018 [4]. Bull-laskentaklusteria on käytetty ennen Kajaanin ammattikorkeakoululle siirtämistä fysiikkaan liittyvien ongelmien numeerisessa ratkaisussa, suurien datamäärien seulomisessa sekä erilaisten tekoälyalgoritmien kehittämisessä. Kajaanin ammattikorkeakoulun tiloissa laskentaklusterin tehoja on hyödynnetty alkulukujen etsintään ja tarkistamiseen, fotogrammetriaan, ohjattuun koneoppimiseen, 3D-renderöintiin, simulointiin ja mallinnukseen sekä muihin korkeateholaskentaan soveltuviin projekteihin. Tulevaisuudessa laskentaklusteria tullaan hyödyntämään myös opetuksen yhteydessä. [4.]

Työn osalta Bull-supertietokoneesta on olennaisinta ymmärtää sen laskentatehon vaatiman sähköenergian määrä ja laatu. Kuten PC-tietokoneidenkin kohdalla, suoritinteho muuttuu lämpöenergiaksi supertietokoneen suorittimilla sekä laskentakorteilla. Supertietokoneessa lämpöenergiaksi muuttuva suoritinteho on täysin eri suuruusluokassa PC-tietokoneisiin verrattuna. KAMK Desk[4]-dokumentaatioiden mukaan Bull-supertietokone tuottaisi noin kahdeksan saunan verran lämpöä. Bull-supertietokoneelle energiaa syöttävään sähköverkkoon kytketyn verkkoanalysaattorin mittauslukemien mukaan sähköenergiaa kulutus voi, riippuen aktiivisten Blade-tietokoneiden määrästä ja rasituksesta, nousta 50 kilowattiin tunnissa.

Supertietokoneiden käyttämän sähköenergian vaihtojännitteen laadun tulee olla hyvä. Jännitteiden vaihtelevuuden täytyy olla hallittua, mahdollisimman vähäistä, puhdasta häiriötransienteista sekä sähköverkon energiansyötössä ei saa esiintyä katkoksia. Näistä syistä johtuen Bull-supertietokoneen infrastruktuurissa hyödynnetään sähköverkon analysointityökaluja. Nämä mittaustyökalut mahdollistavat supertietokoneelle energiaa syöttävän sähköverkon tarkkailun. [6.]

Bull-supertietokone koostuu kahdesta Rack-kaapista, jotka sisältävät yhteensä kymmenen chassis-tyyppistä infrastruktuuria, joihin Bull-supertietokoneen Blade-tietokoneita sijoitetaan. Nämä Blade-muotoiset tietokoneet sijaitsevat kuvassa 1 nähtävän Bull-supertietokoneen etusäleikön takana.



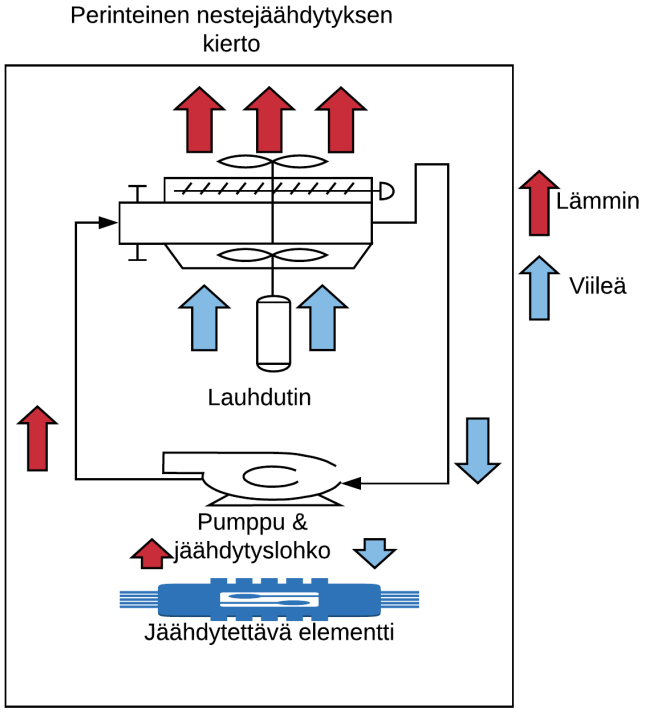
Kuva 1. Bull-supertietokone edestäpäin kuvattuna

2.3 Bull-laskentaklusterin ylläpidollinen automaatio

Bullin ylläpidollinen automaatio koostuu useammasta pääalueesta, joihin lukeutuvat jäähdytysjärjestelmä, sähkönsyöttö ja hallintapalvelin. Tämä opinnäytetyö keskittyy Bullin sähköverkon verkkoanalysaattorin ja jäähdytystä ohjaavan automaatiolaitteiston datan lukemiseen. Nämä laitteistot ovat HPC-ympäristöille toiminnaltaan tärkeitä ja alueet kuuluvat rakennusautomaatiolaitteistoihin. Yleensä näitä laitteita käytetään sähkön ohjaukseen ja jäähdytykseen erikoistuvissa rakenteissa. Tämän työn osalta automaatio on osa rakennetta, joka ohjaa nestekiertoon pohjautuvaa jäähdytysjärjestelmää. Jäähdytyksen ohjaus määrittyy laskentaklusterin ja laitetilan lämpötiloista otettujen mittauksien perusteella, automaatio ohjaa jäähdytyksen nesteen kierron nopeutta ja lauhduttimien tehoja. [4, 7.]

2.4 Nestekiertoinen jäähdytysjärjestelmä

Yleensä tietokoneiden yhteydessä nestekiertoisella jäähdytysjärjestelmällä viitataan prosessorin jäähdytykseen, joka koostuu pumpusta, lauhduttimesta, jäähdytyslohkosta ja yleensä nestesäiliöstä. Tyypillinen kuluttajalaitteistojen nestejäähdytysratkaisu on esitetty kuvassa 2.



Kuva 2. Tyypillisen nestejäähdytysratkaisun toimintakaavio

2.5 Verkkoanalysaattori

Verkkoanalysaattori on sähkön mittauslaitteisto, jolla analysoidaan sähköverkon käyttäytymistä, kuten jännitteiden vaihtelua, eri vaiheiden virran kulutuksia sekä virran kokonaiskulutusta. Näiden tarkkaileminen ja seuraaminen mahdollistaa pitkäaikaisten ongelmien ennakointia sekä ongelmatilanteiden syiden jäljittämistä. Energiankulutuksen seuraaminen mahdollistaa projektien kustannuksien paremman seuraamisen. Tässä kontekstissa verkkoanalysaattorilta saatava informaatio on tärkeää Bullin ylläpidon kannalta, johtuen klusterin suuresta energiankulutuksesta. [4.]

2.6 Teollinen internet

Teollisuusympäristöissä käytetään paljon laitteistoja, joista kerätään dataa ja joilla ohjataan järjestelmiä. Nämä laitteistot yleensä lukeutuvat teollisen internetin alaisuuteen (Industrial Internet of Things, IIOT). IoT ja IIOT-laitteiden tuottamaa dataa käytetään reaaliaikaiseen järjestelmien ja laitteiden säätelyyn, esimerkiksi ilmastointi, automatisoitu teollisuus ja rakennusautomaatio. Rakennusautomaation mittauslaitteilta kerättyä dataa myös tallennetaan monissa ympäristöissä; joissakin tilanteissa sitä myös visualisoidaan. Hyvinä esimerkkeinä toimivat sisäilman lämpötilan esittävät säätimet ja teollisuuden tarkkailussa käytettävät visualisoinnit.

2.7 Automaatioväylät

Automaatioväylien tarkoituksena on välittää automaatiokeskuksen antamia komentoja itse laitteistoille ja järjestelmille. Automaatioväyliin kuuluu monia erilaisia tiedonsiirtoprotokollia, kuten esimerkiksi RTU (RS232, RS422, RS485), Ethernet (TCP, UDP). Lisäksi automaatioväylissä voidaan käyttää erilaisia viestintäprotokollia, esimerkkinä tässä opinnäytetyössä käytettävä Modbus-protokolla.

Automaatiossa käytetään erilaisia protokollia, johtuen eri laitteistojen standardeista ja tyypeistä, esimerkiksi tietokoneella sähkömoottorin etänä ohjaaminen. Näihin protokolliin kuuluvat myös analogiseen jännitteeseen perustuvat ohjaukset sekä on/off-tyyppiset digitaalisignaaleihin perustuvat ohjauskomennot. [8.]

2.8 Aikasarjadata

Aikasarjadatalla viitataan tietokannassa sijaitsevaan tietoa sisältävään pisteeseen, jossa piste koostuu palasta kerättyä tai tuotettua dataa sekä aikaleimasta. Visualisoinnissa näitä aikasarjadatan pisteitä järjestellään aikaleiman perusteella ja pisteiden välille piirretään viiva helpottamaan visualisoidun datan lukemista. Tämän tyyppistä dataa voidaan helposti hyödyntää visualisoidessa ja analysoidessa laitteiston toimintaa.

Aikasarjadatan visualisointi on yksinkertaista, johtuen arvoihin kiinnitetyistä aikaleimoista, joiden avulla on helppo määrittää datan aikaan verrattava järjestys. Tämän järjestyksen pohjalta voidaan piirtää graafi, josta voidaan helpommin erottaa datassa tapahtuneita muutoksia, katkoksia ja vaihteluita. Näitä graafeja voidaan hyödyntää aikasarjadatan esittämisessä esimerkiksi dokumenteissa ja aiheesta selittämiseen.

Aikasarjadatan kanssa yhteensopivia tietokantoja on useampia, mutta ne kaikki toimivat pääosin yhdellä kahdesta tavasta tiedon sisällyttämisen osalta. Useimpiin tietokantoihin täytyy työntää dataa, esim. InfluxDB, mutta osa tietokannoista on mahdollista asettaa noutamaan dataa kohteilta, esimerkkinä dataa noutaviin tietokantoihin toimii Prometheus-tietokanta [9].

3 Laitteistot ja ohjelmistot

Tämän luvun tehtävänä on esitellä laitteita ja ohjelmia, joita opinnäytetyön toteutuksessa tullaan hyödyntämään. Lisäksi luvussa kerrotaan ja selitetään ominaisuuksista, joiden takia kyseinen laite tai ohjelma on valittu käytettäväksi.

Suurimmalta osalta laitteiston valinnat pohjautuivat siihen, mitä oli jo asennettu ja käytössä, sekä myös yhteensopivuuden pohjalta, jotta lopputuloksesta tulisi vakaa ja luotettava ympäristö. Tällä tavalla pystyttiin välttämään suurempia laitepohjaisia ongelmia ohjelmistokehityksen aikana.

Ohjelmistoissa pyrittiin pysymään käyttäjäystävällisissä vaihtoehdoissa sekä suosimaan avoimen lähdekoodin ratkaisuja, jotta ohjelmiston jatkokehitys kolmannen osapuolen toimesta olisi helpompaa. Valintoihin vaikuttivat myös aiemmat käyttökokemukset valituista ohjelmistoista kuten Python-ohjelmoinnin osaaminen.

3.1 Verkkoanalysaattori MPR-63

Sähköverkkoanalysaattori on laite, jolla voidaan mitata sähköverkon rasitusta, eri vaiheiden jännitevaihteluita sekä virran kokonaiskulutusta. Projektissa käytettävä versio MPR-63 on Modbus-protokollan kanssa yhteensopiva ja sisältää RS485-liitännän.

Verkkoanalysaattori on tarpeellinen energiankulutuksen sekä syöttöjännitteiden seurannan osalta. Automaatiojärjestelmissä verkkoanalysaattoreita voidaan käyttää mittaamaan energiankulutusta sekä tarkkailemaan syöttöjännitteen vaihtelua, josta voisi aiheutua ongelmia herkissä laitteistoissa. Näiden mittaustulosten avulla voidaan tarkkailla ja optimoida laitteiden ja järjestelmien energiatehokkuuksia.

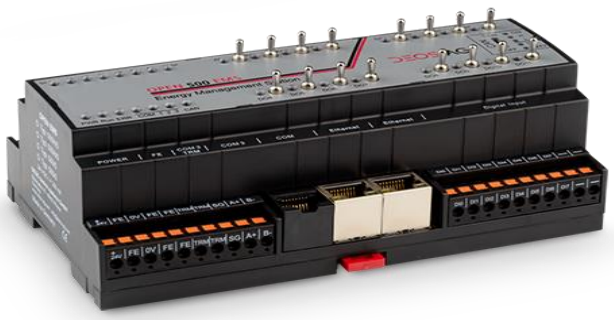
MPR-63-verkkoanalysaattorin valinta käytettäväksi projektissa pohjautui laitteen olemassaoloon työn tilaajalla sekä aiempiin käyttökokemuksiin laitteen luotettavuudesta. Muitakin kaupallisia verkkoanalysaattorivaihtoehtoja sähköverkon arvojen mittaamiseen olisi markkinoilla ollut tarjolla. Laitteen toteuttamista ominaisuuksista huolimatta se ei ole kooltaan kovinkaan suuri, kuten kuvassa 3 on nähtävissä.

Kuva 3. MPR-63-verkkoanalysaattori edestä sekä takaa.

Kuvassa 3 nähtävät ruskeat johdot ovat tässä projektissa käytettävän Modbus-protokollan kanssa käytettävä RS-485-kytkennän kaapelit. [8, 10.]

3.2 Automaatio-ohjain DEOS Open 500 EMS

Asentajan ohjelmoitavissa oleva automaatio-ohjain, josta voidaan myös kerätä dataa. Projektissa käytössä oleva laite on DEOS Open 500 EMS, jossa on käytettävissä Ethernet- ja RS-485-liitännät Modbus-protokollalle, projektissa oleva laite on yhdistetty myös informaationäyttöön Bullin laitteistotilassa. DEOS Open 500 EMS oli jo käytössä Bull-laskentaklusterin jäähdytyksen automaatiossa ennen projektin aloittamista, ja projektin aikana tämän laitteen ajamaa automaatio-ohjelmistoa muokattiin johtuen muutoksista jäähdytysjärjestelmässä. Kuva 4 esittää, miltä DEOS Open 500 EMS näyttää, sekä työssä käytettävät Ethernet-portit. [4,7,8,11.]



Kuva 4. Automaatiota ohjaava DEOS Open 500 EMS [11]

3.3 Python

Python on skriptipohjainen ohjelmointikieli, joka toimii luotettavasti johtuen osittain automatisoidusta rakenteestaan, mutta pitää kiinni monipuolisista ja käyttöjärjestelmän läheisistä ohjelmointiominaisuuksistaan.

Luotettavuuden, selkeyden, monipuolisuuden ja suuren yhteensopivuuden takia Python on tähän projektiin hyvin soveltuva. Projektin lopputulos on yhteyksissä laitteistoon, johon ei saa ilmestyä ongelmia. Lisäksi ohjelmien täytyy olla toimintavarmuudeltaan luotettavia, jotta datan keräämiseen ei tule katkoksia, jotka voisivat korruptoida lopullista visualisoitua tietoa. [12.]

3.4 PyModbus

Python-ohjelmointikielen kanssa toimiva Modbus-protokollan kirjasto, joka mahdollistaa Python-pohjaisten Modbus-protokollapohjaisten kommunikaatio-ohjelmien toteuttamisen. PyModbus-kirjaston dokumentaatio on hyvin selkeää ja yksinkertaista ymmärtää, mikä helpottaa ohjelmointivaiheessa ongelmien ratkomista sekä itse ohjelmointityötä.

Ominaisuuksiltaan PyModbus täyttää kaikki ohjelmien toteuttamisen vaatimukset. Näihin kuuluvat tarkat määrityksen sen osalta, ettei automaatioon päin saa kirjoittaa, koska se voisi aiheuttaa ongelmia järjestelmän toimimisen osalta. Lisäksi vaatimuksiin kuuluvat helposti ymmärrettävä koodi sekä jatkokehitysmahdollisuudet. [13.]

3.5 Prometheus DB

Prometheus on avoimen lähdekoodin käytäntöä noudattava dataa noutava aikasarjatietokanta. Tästä tietokantaohjelmistosta löytyy ominaisuuksia, jotka puuttuvat monista muista tietokannoista, esimerkiksi varoitusympäristöt ja datan noudon kohteiden toimimisen valvonta.

Näiden hakukohteiden state eli tila vaihtelee Up- ja Down-muotojen välillä riippuen datan noutokohteen toiminnan tilasta. Mikäli tila on Down-muodossa, on noutokohteessa päädytty virhetilaan. Näitä tilamuuttujia voidaan myös käyttää varoitusviestien eteenpäin lähettämiseen.

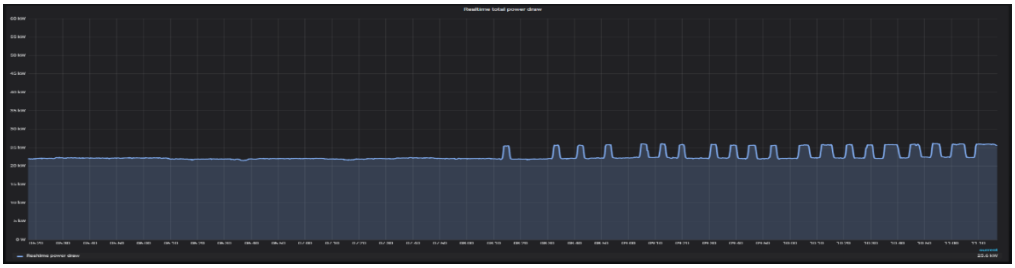
Visualisointityökalun olemassaolosta huolimatta Prometheuksen datanvisualisointityökalussa on suuria puutteita. Nämä puutteet johtavat tarpeeseen käyttää Prometheuksen kanssa erillistä visualisointiohjelmistoa, esim Grafana visualisointityökalua.

Prometheuksen tulevaisuuden käytön valintaan kannustaa myös aiempi kokemus Prometheus-tietokannan käytöstä, yhteensopivuus useiden järjestelmämonitorointiohjelmien kanssa sekä arkistointiominaisuuden yhteensopivuus muiden tietokantojen kanssa. [9.]

3.6 Grafana

Grafana on tehokas ja helppokäyttöinen avoimen lähdekoodin datanvisualisointityökalu, joka on monen tietokantarakenteen kanssa yhteensopiva. Tätä ohjelmistoa käytetään ympäri maailmaa datan esittämiseen helpommin ymmärrettävässä muodossa. Grafanan ominaisuuksiin kuuluvat monipuoliset yhteensopivuudet erimuotoisten tietokantojen ja datatyyppien kanssa, ja johtuen avoimesta lähdekoodista käyttäjät ovat kehittäneet ohjelmistoa eteenpäin ja luoneet ominaisuuksia, joita löytyy harvasta muusta visualisointityökalusta.

Kuvassa 5 esiintyy yksittäisen arvon aikasarja Grafanan visualisointityökaluilla. Näitä visualisointityökaluja löytyy useita, ja niitä on mahdollista muokata ja luoda sopimaan paremmin omaan käyttökohteeseen. [14.]



Kuva 5. Grafanan yhden arvon graafi

3.7 Prometheus pushgateway

Pushgateway on Prometheuksen kehittämä datansiirtoratkaisu, josta löytyy versioita useille ohjelmointikielille, jakaen saman avoimen lähdekoodin periaatteen. Pushgateway voidaan integroida suoraan osaksi ohjelmaa tai ajaa erillisenä ohjelmana, johon muut ohjelmat voivat syöttää dataa Prometheus-tietokannan noudettavaksi. Tästä datansiirtoratkaisusta löytyy myös Docker-ohjelmapaketti, jossa ohjelma pyörii eristetyssä ja hallinnoidussa säiliössä. [9, 15.]

3.8 Docker

Docker on automatisoitu ohjelmien ja järjestelmien säiliöintiympäristö, joka käynnistää eristetyn säiliön jokaista uutta ohjelmaa tai järjestelmää varten. Näiden säiliöiden ominaisuuksia voi muokata ohjauskomentojen parametrejä säätämällä, eikä Docker-säiliöissä ole suurempia rajoituksia siitä, mitä niissä voidaan ajaa.

Tämän työn vaatimuksien täyttämiseksi tarvitaan ohjelmisto, jolla voidaan turvallisesti hallinnoida ohjelmia, uudelleen käynnistää sekä varmistaa datan turvallisuus korruptoitumista kohtaan. Docker-säiliöintiohjelman ominaisuuksien avulla voidaan ohjelma pakottaa uudelleen käynnistymään virhetilanteen tapahtuessa. Tällä tapaa ohjelman luotettavuus paranee sekä pystytään välttämään suurempien katkoksien tapahtumista ohjelmistojen toiminnassa.

Kuvassa 6 nähdään esimerkki säiliön komentorivikonfiguroinnista, jota työn toteutuksessa käytetään. Kuvassa 7 nähdään Dockerin säiliöiden luettelointikomennon toiminta, jossa Docker container-ohjausyksikkö lukee aktiivisena olevat säiliöt ja esittää niiden tiedot käyttäjälle. [16, 17.]

```
docker run -d --network=host --restart unless-stopped prom/pushgateway
docker run -d --network=host --restart unless-stopped theolddog/pymodbus_sync_rtu
docker run -d --network=host --restart unless-stopped theolddog/pymodbus_sync_tcp
```

Kuva 6. Docker-säiliöiden käynnistyskomentoja

```
CONTAINER ID        IMAGE                          COMMAND             NAMES
a569889fe972        theolddog/pymodbus_sync_tcp    "python3 app.py"    ecstatic_germain
b3c9cd4df762        theolddog/pymodbus_sync_rtu    "python3 app.py"    vigorous_greider
873ec8bb93c3        prom/pushgateway               "/bin/pushgateway"  blissful_rubin
```

Kuva 7. Docker-säiliöiden tarkastelukomennon ulostulo

3.9 Kehitystyökalut

Tiedonluvun Python-ohjelmien toteuttamiseen käytettiin Pythonin asennuksen mukana tulevaa editoria sekä Geany-tekstinkäsittely ohjelmaa, joka on saatavilla Linux-käyttöjärjestelmille.

Modbus-ohjelman toiminnan testaamiseen käytettiin Modbus PLC Simulator-ohjelmaa. Tämän ohjelman avulla on mahdollista simuloida Modbus-protokollaa käyttäviä laitteita ja datarekistereitä. Ohjelma myös näyttää, minkä muotoinen viesti kommunikaatiossa liikkuu.

Testausohjelman käyttövaatimus perustuu riskiin kirjoittaa automaatiojärjestelmään päin ja pahimmillaan aiheuttaa toimintahäiriöitä jäähdytyksessä. Kehityksen ohella käytettiin Citius Group Oy:n Desk-dokumentaatiotyökalua. Tällä tapaa varmistetaan mahdollisuus toteuttaa työ uudelleen tai korjata tilanteen vaatiessa. [4.]

4 Työn toteutus

Tässä luvussa käydään läpi opinnäytetyön toteuttamiseen tehty esiselvitys ja projektin vaatimuksien ja rajoituksien määrittely. Myöhemmin luvussa kerrotaan projektin suunnitteluvaiheessa tehdyistä päätöksistä, selitetään järjestelmän rakenteesta sekä erillisten osuuksien toteutuksista.

4.1 Esiselvitys

Ennen toteutuksen aloittamista täytyi projektia varten toteuttaa esiselvitys. Tämän esiselvityksen täytyi sisältää vaihtoehtoisia toteutusmahdollisuuksia, laitteiden ja ohjelmistojen ominaisuudet, luotettavuus sekä rajoitteet ja myös selvittää mahdollisuudet virtualisoida työn lopputulos tai ainakin osa toteutuksesta. Esiselvityksen perusteella tehtiin suurin osa päätöksistä, mitä laitteita ja ohjelmistoja projektin toteuttamiseen käytettäisiin. Koska ratkaisun tärkeimpiä ominaisuuksia tulisi olemaan luotettavuus, jatkokäytettävyys sekä avoin lähdekoodi, osa valittiin johtuen niiden jo käytössä olemisesta.

Esiselvityksessä tehtiin vertailua mahdollisista käytettävistä tietokantaratkaisuista. Vertailussa verrattiin maksullisella, ilmaisella sekä avoimen lähdekoodin lisenssillä olevia vaihtoehtoja. Vertailun perusteella lopullisiksi vaihtoehdoiksi karsiutui seitsemän toimivaa ja aktiivisesti ylläpidettyä tietokantaa. Vertailua eniten karsiviin tekijöihin kuului avoimen lähdekoodin vaatimus, aikasarjadatan yhteensopivuus, kaupallinen käytettävyys, ylläpito/jatkuva kehityspolku sekä aiempi kokemus.

Käytettävyyspohjainen vertailu tehtiin myös visualisointityökaluille. Niitä oli kuitenkin jo heti vähemmän, johtuen käytettävissä olevista tietokannoista. Lisäksi visualisoinnissa yhteensopivuuden lisäksi arvioitiin helppokäyttöisyyttä ja toimivuuden varmuutta.

Projektin toteutuksen vaatimukset pohjautuivat mahdollisien riskitekijöiden välttämiseen sekä kaiken mahdollisen datan laitteistosta lukemiseen, talteen ottamiseen sekä visualisointiin. [3, 4, 7, 10, 12-17.]

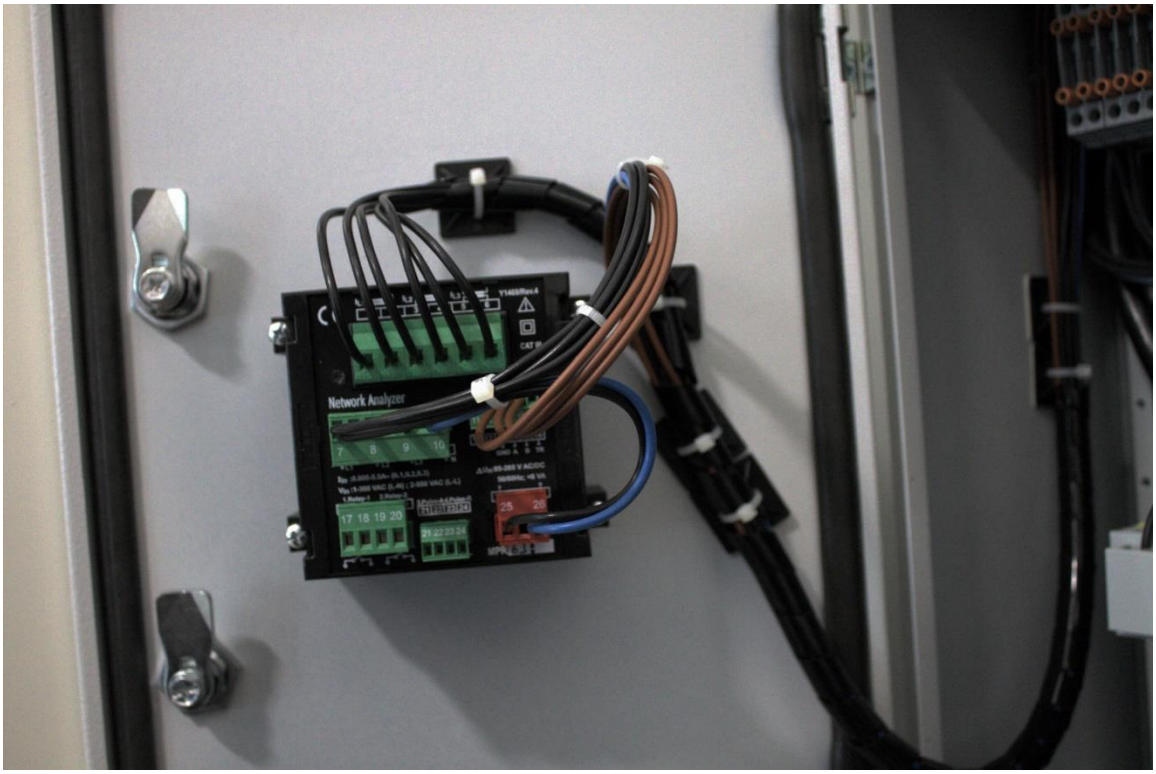
4.2 Suunnittelu

Toteutuksen suunnittelu aloitettiin lukitsemalla valinnat koskien ohjelman isompia osia, kuten tietokantaratkaisu, visualisointiohjelmisto, laitteet sekä datansiirto- ja muotokäytännöt. Näiden päätösten pohjalta aloitettiin toteuttamaan vuokaaviomuotoista suunnitelmaa, jossa kuvattiin, mistä, minne ja millä protokollalla ja liitännällä, sekä minkä laitteiden kautta dataa siirretään alkuperästä aikasarjatietokantaan, josta dataa myöhemmin käytetään visualisoinnissa. Suunnittelun aikana päätettiin myös, että alustava koodin ja ohjelmien testaaminen toteutetaan hallitussa ympäristössä, jotta ei sotkettaisi kalliita ja pahimmillaan hyvin herkkiä automaatiojärjestelmiä. Visualisoinnin osalta suunnittelu koostui pääosin päätöksistä koskien, mitä kaikkea dataa tultaisiin visualisoimaan sekä mahdollisia tyylejä ja tekniikoita, joilla data voitaisiin visualisoita.

Suunnittelussa otettiin myös huomioon, että laskentaklusteria sekä tämän projektin lukemaa dataa voidaan tulevaisuudessa käyttää kaupallisesti. Suunnittelun aikana huomioitiin myös luotettavuuden vaatimus, ja mahdollisessa virhetilanteessa ohjelman täytyisi käynnistyä uudelleen ilman käyttäjän apua. Uudelleen käynnistymisen osalta vaihtoehtoja oli Linux Daemonin luominen tai Docker-säiliöinnin hyödyntäminen, joka vaati ohjelmiston pakkaamisen Docker-säiliömuotoon. Tästä ei ollut projektin osallisilla aiempaa kokemusta; tästä johtuen suurin osa ongelmista kohdistui Docker-säiliöihin. [16,17.]

4.3 Verkkoanalysaattorin seurantajärjestelmä

Tämän järjestelmän tarkoitus on lukea dataa MPR-63-verkkoanalysaattorilta, käsitellä ja skaalata data oikean muotoiseksi, jotta dataa on myöhemmin helpompaa ymmärtää ja lukea. Ohjelman, jolla verkkoanalysaattoria luetaan, täytyy kommunikoida Modbus RTU-protokollalla käyttäen RS-485-standardia. RS-485-standardin käyttö johtuu MPR-63:n Ethernet-portin puutteesta. Kuvasta 8 nähdään MPR-63-laitteen liitäntöjä, joiden kautta laite toteuttaa mittauksena ja joiden kautta laitteen mittauksia luetaan.



Kuva 8. MPR-63-verkkoanalysaattorin kytkentäpaneeli

Verkkoanalysaattorin lukemiseen käytetään RS485 USB-adapteria, jonka ohjaamiseen käytetään Python-ohjelmaa, joka hyödyntää PyModbus-kirjastoa Modbus-kommunikoinnin yksinkertaistamiseen. USB-adapteriin kiinnittyvät johdot on kiinteästi kytketty sähkökaapin rakenteeseen. Nämä johdot ovat ruskean värisiä ja merkitty kaapin sisällä, ne ovat nähtävissä kuvan 8 ruskeina johtoina.

Ohjelma alustaa tarvitsemansa muuttujat ja Modbus-kommunikaation yhteyden parametrit käynnistyksen yhteydessä, jonka jälkeen ohjelma siirtyy toiminnalliseen silmukkaan. Toiminnallisessa silmukassa ohjelma varmistaa Modbus-laitteeseen tehdyn linkin, jonka jälkeen siirtyy lähettämään Modbus-komennot laitteelle datan noutamista varten. Tämän jälkeen datalle tehdään yhteensopivuusskaalaukset ja siirretään pushgateway-ohjelmaan, josta Prometheus-tietokanta noutaa datan. Ohjelman toimintaa selkeyttävä vuokaavio on nähtävissä Modbus-ohjelmaa koskevassa luvussa 4.4.2 kuvassa 10.[8,10.]

4.4 Automaation seurantajärjestelmä

Automaation seurantajärjestelmä lukee dataa DEOS Open 500 EMS-laitteelta, joka on vastuussa jäähdytyskierron antureiden lukemisesta sekä jäähdytyksen hallinnoinnista. Tämän lukemista varten toteutettava ohjelma ei saa kirjoittaa jäähdytykseen päin, johtuen riskistä sekoittaa jäähdytysjärjestelmää ohjaava automaatio.

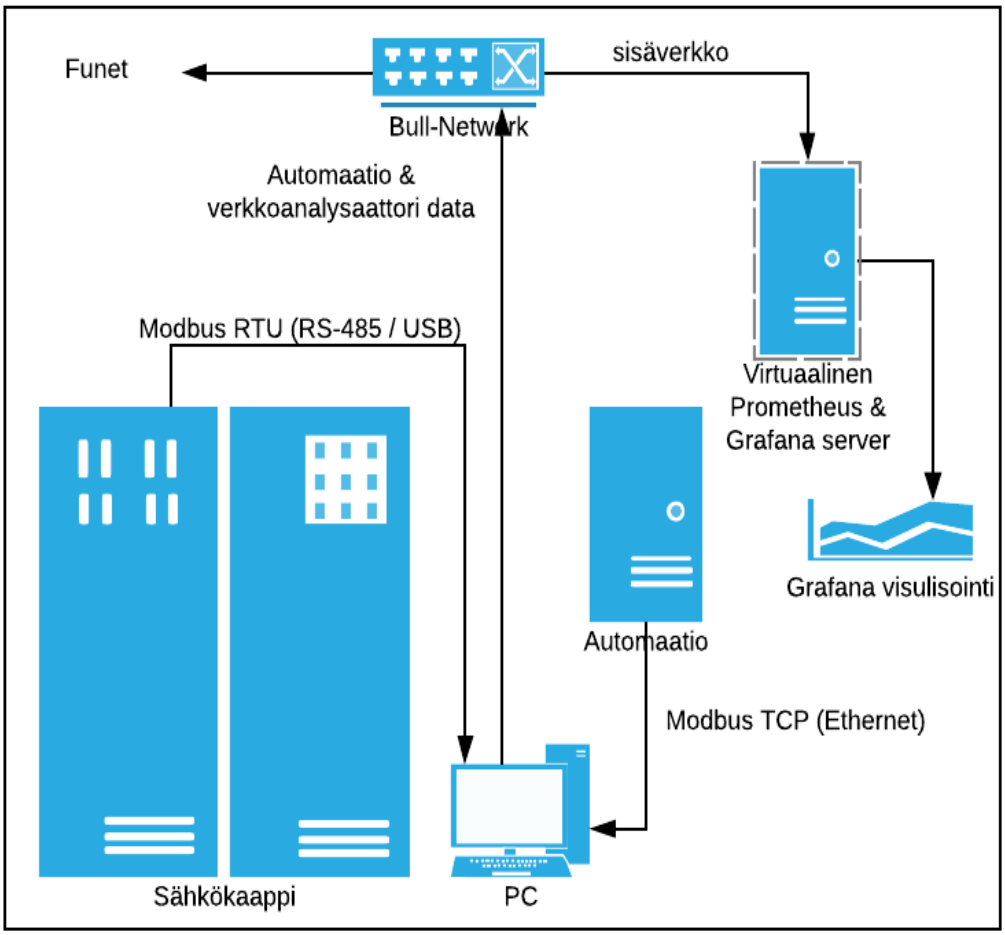
Automaation arvojen lukemisen toteuttamiseksi ohjelman kommunikointi ohjauslaitteen kanssa toteutetaan Modbus TCP-protokollalla käyttäen Ethernet-liitäntää. Mahdollista olisi myös käyttää RS-485-liitäntää, mutta tämä portti pidetään vapaana mahdollisia järjestelmän laajentamisia varten.

Automaatiojärjestelmän datan lukeminen toteutetaan Ethernet-liitännän kautta, jonka takia erikoisempia adaptereita ei tarvita. Ethernet-yhteydet eristetään toisistaan, jotta tulevaisuudessa mahdolliselta datan korruptoitumiselta voidaan välttyä. Lukemiseen käytettävän ohjelman toiminta on lähes täysin sama kuin verkkoanalysaattorin lukemiseen käytettävän ohjelman. Suurimmat erot löytyvät PyModbus-kirjaston konfiguraation parametreistä sekä datarekistereistä, joita Modbus-komennoilla laitteilta luetaan. Ohjelma määritetään muodostamaan yhteys Ethernetin kautta määritettyyn IP-osoitteeseen, jonka jälkeen Modbus-kommunikaatiolla haetaan datarekisterien tiedot.

Tietojen laitteilta lukemisen prosessin aikana tiedot myös käsitellään sopiviin muotoihin ja siirretään pushgatewayn kautta ohjattuna Prometheus-tietokantaan. Ohjelman toimintaa selkeyttävä vuokaavio on nähtävissä Modbus-ohjelmaa koskevassa luvussa 4.4.2 kuvassa 10. [7,8,11.]

4.4.1 Rakenne

Järjestelmän rakenne pohjautuu yksinkertaistettuun putki-ideaan, jossa pyritään eliminoimaan ylimääräisiä epäluotettavuuden pisteitä sekä monimutkaisuuksia. Pohjarakenne projektissa on lukea mittausdata verkkoanalysaattorilta sekä jäähdytyksen automaatiosta käyttäen Modbus-protokollaa, datan lukemisen jälkeen täytyy tiedot käsitellä ja skaalata ihmisten helpommin ymmärrettäväksi. Kuvassa 9 esiintyy lopullisen laitteiston asentamiseen sovellettava vuokaavio, joka kertoo myös, missä ja minkä tyyppisillä yhteyksillä ja protokollilla dataa siirrellään.

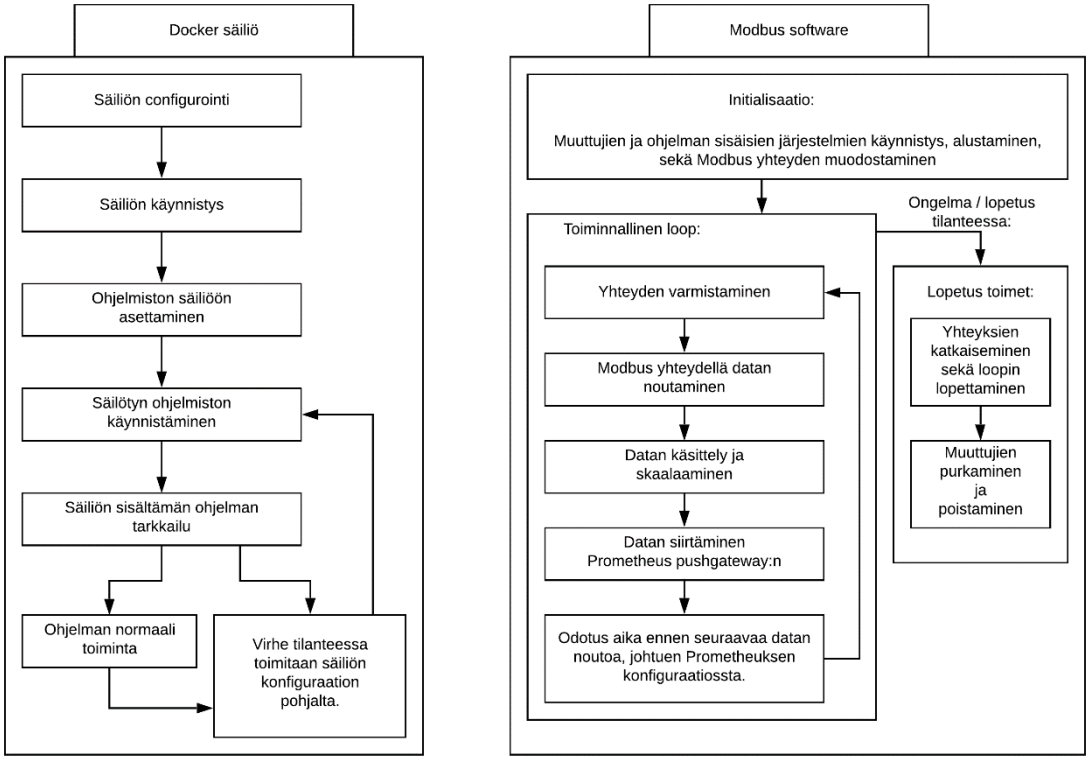


Kuva 9. Fyysisen kytkennän ja datan kulun rakennekaavio

4.4.2 Modbus-ohjelma

Modbus-kommunikointiin käytettävät ohjelman toteutettiin kahtena erillisenä ympäristöjä, joiden käsittelemä data siirretään yhdistettyyn pushgatewayhin, josta Prometheus-tietokanta sen noutaa. Nämä ohjelmat ovat toimintaperiaatteeltaan ja rakenteeltaan lähes samanlaiset, eroina ovat yhteydet, joiden kautta Modbus toimii, sekä Modbus-datarekisterien osoitteet.

Pushgateway sekä Modbus-ohjelmat ajetaan Dockerin hallinnoimissa säiliöissä, jonka avulla vältetään mahdollisia riskitekijöitä, joista voisi aiheutua datan korruptoitumista, ohjelmien kaatumista, muistivuotoja tai muita ongelmia. Dockerissa ohjelmien ajaminen tuo myös mukanaan virheistä palautumisen, tämän mahdollistaa säiliön konfiguraation asetus, joka yrittää uudelleen käynnistää ohjelman, mikäli virhetilanne ilmestyy tai ohjelma kaatuu. Docker-säiliöiden ja Modbus-ohjelmien toiminta esitetään kuvassa 10.

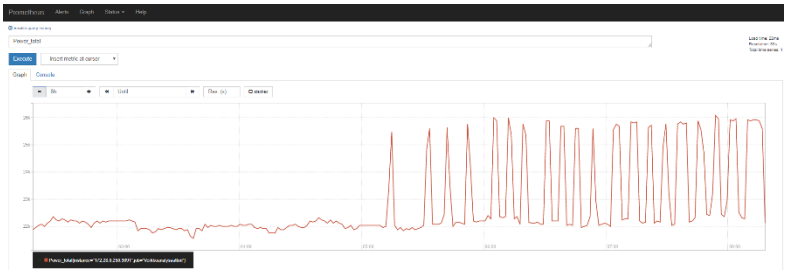


Kuva 10. Lohkokaavioesitys ohjelmien toiminnasta ja rakenteesta

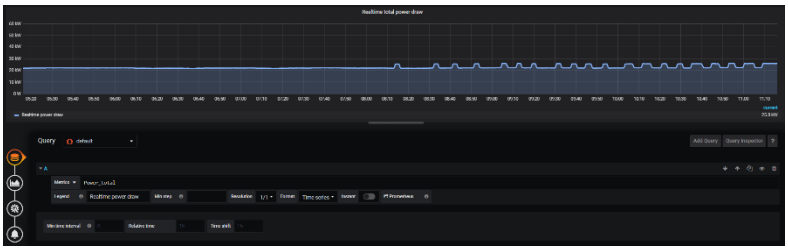
Tämä lohkokaavio esittää, millä tapaa Docker-säiliö ja Modbus-protokollalla datan hakemiseen erikoistuvat ohjelmat toimivat. Modbus-ohjelmaa ajetaan Docker-säiliön sisällä, jotta sen hallinnointi olisi mahdollisimman automatisoitua ja luotettavaa. [8 - 10,12.]

4.4.3 Käyttöliittymä

Projektissa käyttöliittymä on muutamaan osaan jakautunut, ohjelmien ohjaus toimii komentorivillä ja Docker-säiliöiden avulla, jonka pohjalta ohjelmien pitäisi toimia automatisoidusti Dockerin hallinnoimana. Prometheuksen asetuksien määrittäminen tapahtuu testitiedostojen käsittelyllä, ja Grafanan säätäminen toimii verkkokäyttöliittymän avulla. Prometheus ja Grafana molemmat käyttävät konfiguroinnin jälkeen verkkokäyttöliittymää, jonka graafinen esitys on yksinkertainen navigoida. Seuraavissa kuvissa 11 ja 12 on nähtävissä käyttöliittymäeroja Prometheuksen ja Grafanan visualisointityökalujen välillä.



Kuva 11. Prometheus-tietokannan visualisointityökalu



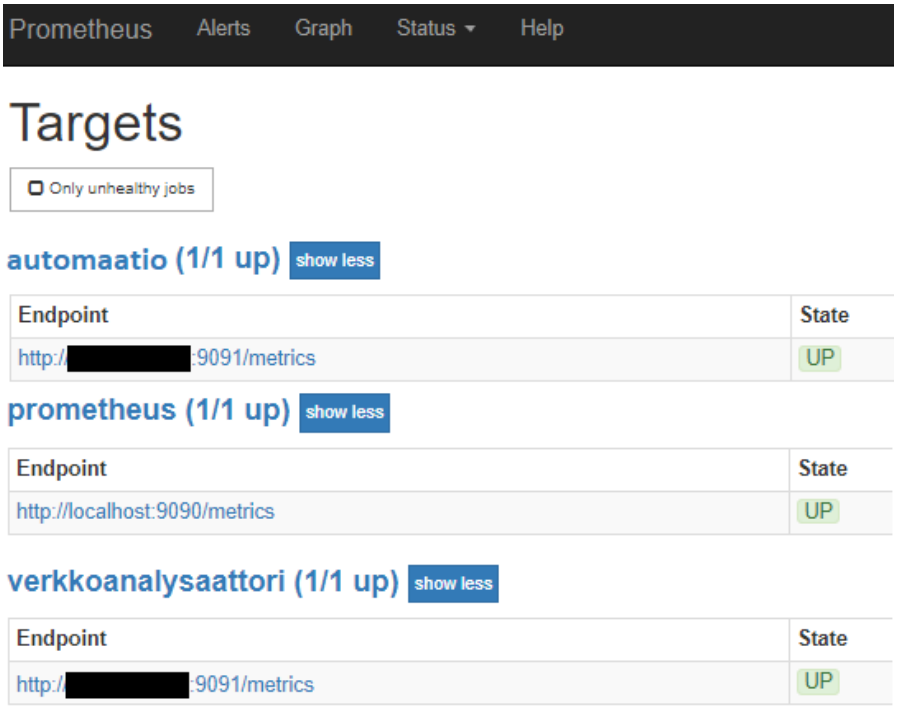
Kuva 12. Grafanan visualisointityökalu

Käyttöliittymien hyvät ja huonot ominaisuudet korostuivat suuresti visualisoinnin toteutuksen aikana sekä ohjelmistojen asetuksien säätämisessä, kun ohjelmistoja siirreltiin rautapohjaiselta toteutukselta virtualisoituun ympäristöön. Datan visualisoinnin toteutuksessa ohjelmien käyttöliittymien erot näkyivät paremmin. Suurimmat erot kuitenkin johtuvat ohjelmistojen alkuperäisistä käyttötarkoituksista. Grafana on alusta lähtien kehitetty visualisointityökaluksi ja Prometheus on tietokanta, johon on rakennettu visualisointiominaisuus. [14,15.]

4.4.4 Tietokanta

Tietojen tallentamiseen käytetään keskitettyä Prometheus-tietokantaa, joka erikoistuu aika-sarjadataan. Prometheus asennetaan ja ylläpidetään Kajaanin ammattikorkeakoulun datacenter-koulutuslinjan tiloissa sijaitsevassa opiskelijoiden ylläpitämän palvelinrakenteen virtualisoidussa käyttöympäristössä. Tämä mahdollistaa alhaallaoloajan minimoinnin ja parantaa järjestelmän luotettavuutta. Prometheus-tietokantaan täytyy konfiguroida kohteet, joista tietokanta noutaa aikasarjapisteet. Näitä kohteita voivat olla esim. Prometheus-pushgatewayt, tekstitiedostot ja muut tietokannat.

Dataa noutava tietokanta helpottaa pidemmällä aikaa järjestelmien ylläpidossa, varsinkin tilanteissa, joissa tietokannan IP-osoite vaihtuu, mutta kohteiden osoitteet eivät. Seuraavassa kuvassa 13 on nähtävissä lista Prometheus-tietokannan datan noudon kohteista. [15.]



Kuva 13. Prometheuksen datan hakukohteiden lista

4.4.5 Visualisointi

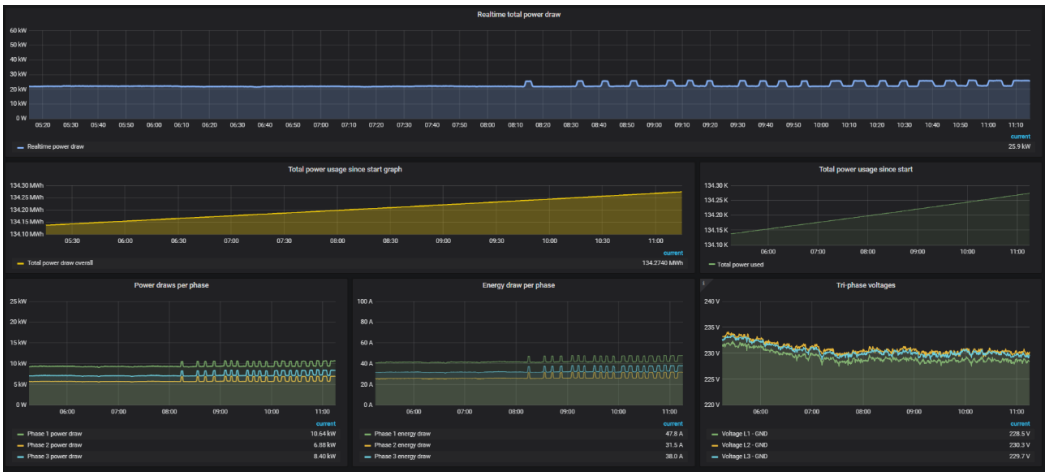
Visualisoitava data täytyy esittää muodossa, josta datan lukijan on helppoa saada tietoon, mitä dataa esitetään, mistä kyseinen data tulee sekä missä muodossa tai tyypissä data on, esim. Celsius, KPa, voltit, ampeerit, kilowatit.

Näiden vaatimuksien täyttämiseksi visualisointi perustuu aikasarjadataan ja nimettyihin muuttujiin. Tällä tapaa voidaan lukijalle esittää, mitä mittauksia graafi sisältää, mitä arvoja mittauksissa on saatu. Lisäksi Grafanan ja Prometheuksen sisäänrakennetut ominaisuudet mahdollistavat varoitussääntöjen luomisen, mutta niiden luomisen haastavuus on suuresti vaihteleva ohjelmistojen välillä.

Kuvissa 14 ja 15 on nähtävissä Grafanan ominaisuuksien mahdollistama visualisointi, jossa voidaan esittää data useammista lähteistä yhden Dashboardin sisällä. Näissä kuvissa nähdään Grafanalla toteutetut dashboardit, joissa visualisoidaan Modbus-protokollaa käyttävillä ohjelmilla luettua dataa.

Kuvassa 14 nähdään jäähdytyksen automaation puolta Ethernetin yli luetusta DEOS Open 500 EMS-laitteen datasta muodostettu dashboard. Kuvan 15 dashboardissa esitetään verkkoanalysaattorin lukemaa data siitä, minkä tyyppistä vaihtelua ja kulutusta verkossa tapahtuu. [10,11,14,15.]

Kuva 14. Grafanan automaation datan visualisointidashboard



Kuva 15. Grafanan verkkoanalysaattorin dashboard

4.5 Laitteiston ja ohjelmistojen asentaminen

Lopullinen laitteisto, jonka päälle järjestelmä rakennetaan, on pienikokoinen Linux-käyttöjärjestelmällinen tietokone, jossa on saatavilla tarvittava määrä USB- ja Ethernet-liitäntöjä sekä passiivinen jäähdytys, jotta vältytään tuulettimen takia johtuvasta vikaantumisesta.

Laitteen ei tarvitse olla tehokas, joten prioriteetti asettui enemmän liitäntöjen, ominaisuuksien ja fyysisen koon osalle. Prosessorina on energiatehokkuuteen tähtäävä Intel Celeron J1900 4-ydinprosessori, keskusmuistia on 2 gigatavua ja tallennustilan määrällä ei ole suurempaa vaikutusta, joten 32 gigatavua riittää hyvin.

Loppusijoituksena tietokone tullaan asentamaan osittain piiloon Bull-laitetilaan, kuten seuraavassa kuvassa 16 nähdään; tietokone tullaan myös merkitsemään tunnistetarroilla dokumentaatioon määritetyllä tavalla.

Kuva 16. Datanseurantajärjestelmän tietokone

Laitteelle asennetaan Python-ohjelmointikielen käyttämiseen tarvittavat työkalut sekä kirjastot. Pääosin nämä tulevat Python-paketin mukana, mutta osa projektissa käytettävistä kirjastoista haetaan ja asennetaan PIP-Python-kirjastohallintatyökalun avulla. Näihin kirjastoihin kuuluvat PyModbus sekä Prometheuksen Python client-kirjastot. Lisäksi PIP-työkalua käytetään useiden Docker-ohjelmistojen vaatimuksien täyttämiseen. [4, 12, 18].

Docker voidaan asentaa lataamalla verkosta tai Linuxin tapauksessa voidaan Docker ladata ja asentaa suoraan terminaalikomennon avulla. Dockeria käytetään projektin ohjelmistojen ajamiseen ja hallitsemiseen. DockerHub on pilvitallennusalusta, josta noudetaan Prometheuksen pushgateway sekä molemmat projektissa toteutetut Modbus-kommunikaatio-ohjelmat. [16, 17].

Ohjelmien asentamisen jälkeen konfiguroidaan tietokoneen verkkorakenne, jotta voidaan turvallisesti käyttää ohjelmistoja, ilman riskiä kirjoittaa automaation hallinnan suuntaan. Konfiguroinnin ja asentamisien jälkeen toteutetaan laitteiston fyysinen sijoittaminen ja tarvittavien johtojen paikoilleen vetäminen. Loppuasennuksessa seurataan luvussa 4.4.1 olevassa kuvassa 9 esiintyvää kaaviota.

5 Järjestelmän testaus

Projektin kehityksessä testausta ja testauksien kirjaamista pyrittiin pitämään mahdollisimman lähellä kehitystyötä, jotta pysyttäisiin aktiivisesti tietoisina siitä, miten ohjelmistot ja laitteet käyttäytyvät. Testaamisen tärkeys perustuu tietoon siitä, kuinka helposti kerätty data voi korruptoitua ja kuinka paljon virheellinen viesti voi aiheuttaa tuhoa. Esim. jäähdytysautomaation ohjelman lopullisessa ympäristössä testaamisessa olisi voinut tapahtua jäähdytysjärjestelmän kaatuminen ja supertietokoneen kuumeneminen.

Lopullisen järjestelmän asennus toteutettiin aiemmin luvussa 4.5 käydyllä tavalla, ja fyysinen tietokoneen paikka sijoitetaan luvussa 4.4.1 kuvan 9 kaavion mukaisesti automaatiohallintakaapin ja sähkökaapin läheisyyteen. Tämän tyyppisellä sijoituksella voidaan varmistaa datan siirtoon käytettävien kaapelien pituuden optimointi ja samalla vähennetään mahdollisuutta siirrettävän datan korruptoitumiseen sähkömagneettisen säteilyn takia.

Järjestelmän testaaminen perustuu luettujen ja visualisoitujen arvojen vertailuun alkuperäisen laitteiston mittauksien kanssa. Tätä vertailua toteutetaan manuaalisesti ja siihen uppoaa paljon aikaa, eikä arvojen tarkkailu ja vertailu tule loppumaan opinnäytetyön suorituksen loputtua, vaan tulee jatkumaan jatkokehityksen mukana.

6 Pohdinta

Bull-supertietokoneen ylläpidollisen automaation datan lukemisen ja visualisoinnin projektissa törmättiin useisiin ongelmiin, joista suurin osa johtui muuttuvista toteutusympäristöissä, joissa projektia toteutettiin. Osa ongelmista ja muutoksista johtui projektin pituudesta, koska projektin aloitus tapahtui jo kesällä 2019. Tuli saataville uutta laitteistoa sekä pakollisia muutoksia laitteistossa, kuten Prometheus-tietokantapalvelimen siirto fyysiseltä laitteistolta virtualisoituun ympäristöön.

Ohjelmiston toteutuksessa ongelmia ilmestyi pääosin luettujen arvojen skaalauksessa sekä osan visualisointialgoritmien antamien arvojen eroavaisuuksista verratessa ylläpitolaitteiston mittauksiin. Automaationhallintalaite DEOS Open 500 EMS käyttää omia algoritmejä, joita projektin toteutuksen aikana ei päästy näkemään ja sen pohjalta muokkaamaan projektin käyttöön sopiviksi.

Tämä algoritmien eroavaisuus aiheuttaa eroa lopullisen osan visualisoitavan datan ja automaation mittaaman datan välillä. Tämä ongelmatilanne jäi opinnäytetyön aikataulun aikana osittain ratkaisematta, mutta jatkokehityksen aikana kyseinen ongelma tullaan korjaamaan.

Suurimpina onnistumisina projektissa nähtiin Linux-käyttöjärjestelmän monipuolisuus mahdollistaa erilaisten ohjelmistojen yhtenäistä käyttöä, Docker-säiliöintityökalun ominaisuuksia luoda lähes järjestelmätason ohjelmistoja sekä ylläpitää niitä ilman käyttäjän puuttumista toimintaan.

Projektin tehtävänä oli toteuttaa automaatio- ja ylläpitolaitteistoilta datan lukeminen, tallentaminen tietokantaan sekä visualisointi, jotta luettu data olisi helpompaa ymmärtää ja esittää tarvittaessa. Lopputuloksena laitteistoista saadaan luettua raakadataa, tallennettua se turvaan tietokantaan sekä visualisoida tietokantaan siirrettyä dataa.

7 Yhteenveto

Tavoitteena työssä oli kerätä Bull-supertietokoneen infrastruktuurin automaatio- ja ylläpitolaitteista dataa, tallentaa kerätty data aikasarjatietokantaan ja visualisoida data helpommin ymmärrettävään muotoon.

Työn toteutukselta vaadittiin luotettavaa toimintaa ympärivuorokautisessa toiminnassa. Sen tuli pystyä palautumaan mahdollisista ongelmatilanteista automatisoidusti sekä toimia luotettavasti pitkällä aikavälillä. Merkittävä vaatimus sovelluksen toiminnalle asetettiin liittyen automaatiojärjestelmän kanssa kommunikointiin; sovelluksella oli lupa vain lukea tietoja, eikä kirjoittaa niitä automaatioon päin. Automaatiojärjestelmän suuntaan kirjoittaminen saattaisi aiheuttaa jäähdytysympäristön parametrien muuttumisen ja pahimmillaan jäähdytysjärjestelmän vikaantumisen.

Vaatimuksiin perustuen opinnäytetyössä suunniteltiin ja toteutettiin sovellus, joka lukee dataa supertietokoneen automaatiojärjestelmästä sekä verkkoanalysaattorilta käyttäen Modbus-protokollaa. Luettu data siirretään sisäverkon yli tietokantaan, josta se visualisoidaan.

Datan lukemista varten kehitettiin Python-ohjelmointikielellä toteutetut ohjelmat, joihin Modbus-protokolla on implementoitu PyModbus-kirjaston avulla. Noudettu data käsitellään ja siirretään Prometheus-aikasarjatietokantaan käyttäen Prometheuksen omaa Pushgateway-ratkaisua. Datan visualisointi toteutetaan noutamalla aikasarjadataa Prometheus-tietokannasta Grafanan visualisointiympäristöön.

Järjestelmän tietokanta- ja visualisointipalvelimien ylläpito toteutetaan virtualisoidussa ympäristössä KajakDC:n ylläpitämänä, mutta Modbus-kommunikointiohjelmat ajetaan fyysisellä laitteistolla. Kommunikointiohjelmat suoritetaan Docker-säiliöissä, jotta ohjelmien virhetiloista palautuminen voidaan varmentaa ja ohjelmat pystytään eristämään muusta järjestelmästä. Työssä kehitetys järjestelmän datanlukuohjelmien suoritusympäristöksi valittiin passiivisella jäähdytyksellä varustettu pienikokoinen tietokone.

Työssä kehitetty sovellus Bull-supertietokoneen ylläpidollisen datan keräämiseen ja visualisointiin täyttää sille projektin alussa asetetut vaatimukset. Toteutuksen aikana tehdyissä testauksissa huomioitiin ongelmia visualisointialgoritmien viimeistelyn osalta. Nämä ongelmat on kirjattu KAMK Desk-dokumentaatioon ja siirretty järjestelmän jatkokehityksessä korjattaviksi.

Lähteet

1. CSC [Internet]. Saatavilla: https://www.csc.fi
2. FUNET [Internet]. Saatavilla: https://en.wikipedia.org/wiki/FUNET
3. Kajak DC [Internet]. Saatavilla: https://kajakdc.fi/
4. KAMK sisäinen DESK dokumentaatio sivusto.
5. Atos Oy [Internet]. Saatavilla: https://atos.net/en/
6. Perfecting Electrical Efficiency in Data Center Power Networks [Article]. Saatavilla: https://3dfs.com/articles/perfecting-electrical-efficiency-data-center-power-networks/10/2017
7. TujuCon sähköposti kommunikaatio opinnäytetyön toteutuksen aikana, osallisina T.Holappa, H.Lehtimäki ja J.Jurvansuu.
8. Modbus [Internet]. Saatavilla: http://www.modbus.org
9. Prometheus: Time-series database [Internet]. Saatavilla: https://prometheus.io
10. MPR-63 Network Analyzer [Internet]. Saatavilla: http://www.enteselectronics.com/mpr_63.html
11. OPEN 600 EMS / OPEN 500 EMS [Internet]. Saatavilla: https://www.deos-ag.com/en/products/building-automation/ddc-controller/open600500/
12. Python [Internet]. Saatavilla: https://www.Python.org
13. PyModbus: Modbus Lib [Internet]. Saatavilla: https://pymodbus.readthedocs.io/en/latest/index.html#
14. Grafana: The open observability platform [Internet]. Saatavilla: https://grafana.com/
15. Prometheus: PushGateway [Internet]. Saatavilla: https://github.com/prometheus/pushgateway
16. Docker: Software container [Internet]. Saatavilla: https://www.Docker.com/
17. Docker Documentation [Internet]. Saatavilla: https://docs.Docker.com/
18. Pip Python package manager [Internet]. Saatavilla: https://en.wikipedia.org/wiki/Pip_(package_manager)

Liitteet