Jukka Juntunen

Konesaliympäristön monitorointi IoT-teknologian avulla

Tietotekniikan
Pro gradu -tutkielma
28. huhtikuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Jukka Juntunen
Yhteystiedot: jjjuntux@student.jyu.fi
Puhelinnumero: 050 436 4400
Ohjaaja: Risto T. Honkanen, Lasse M. Harjumaa
Työn nimi: Konesaliympäristön monitorointi IoT-teknologian avulla
in English: IoT based server room monitoring
Työ: Tietotekniikan Pro gradu -tutkielma
Sivumäärä: 79+6
Tiivistelmä: Konesaliympäristössä sijaitsee tietojenkäsittelyssä tarvittavia laitteistoja, joiden luotettava toiminta on monien organisaatioiden toiminnan kannalta tärkeää. Laitteiston toimintaa haittaavat fysikaalisten olosuhteiden muutokset ja muut ennakoimattomat tilanteet, kuten tulipalot. Tämän pro gradu -tutkielman pääkysymys on "Miten IoT-teknologia auttaa ylläpitäjää konesaliympäristön monitoroinnissa?". Työn teoreettisen osan tutkimusmenetelmänä käytetään kirjallisuuskatsausta. Työn empiirisessä osassa käytetään konstruktiivista tutkimusotetta, jota sovelletaan IoT-laitteeksi luokiteltavan konesaliympäristön olosuhteita mittaavan laiteen suunnittelussa ja toteutuksessa. Tuloksena saavutetaan kokonaisuus, joka auttaa konesalin ylläpitäjää hahmottamaan vallitsevat olosuhteet sekä vertaamaan nykyisiä olosuhteita aikaisempiin visuaalisten näkymien avulla. Toteutettu kokonaisuus mahdollistaa myös hälytysten lähettämisen ongelmatilanteen alkaessa ja päättyessä.
Avainsanat: ESP32, Zabbix, mittauslaite, monitorointijärjestelmä
Abstract: Server room is a dedicated environment for devices needed in data processing. From the organizational perspective, stability of IT equipment placed in the server room is crucial. Negative changes in physical conditions, as well as unpredictable situations, such as destructive fires, affects on the devices operability. This study aims to answer the following question: "How can IoT technology assist in data center monitoring?" This study is divided into two parts, which are theoretical and empirical. Literature review is used as a research method in the theoretical part of this study. Empirical research is based on constructive method. The empirical part includes designing and implementing an IoT device, which has the capability to measure conditions in the server room. As an outcome, the produced appliance assists administrators to detect current conditions in the server room and enables comparing them to previous measurements using visual views. The produced solution also enables to send alerts when a problematic situation occurs and after recovering.
Keywords: ESP32, Zabbix, measuring instrument, monitoring system

Sanasto

AWS	Amazon Web Services. Amazonin tuottama pilvipalvelualusta.
Bash	Bourne again shell. Komentotulkki.
BLE	Bluetooth Low Energy. Langaton tiedonsiirtoteknologia, jolla on matala energiankulutus.
CPU	Central Processing Unit. Suoritin. Komponentti, jonka tehtävänä on suorittaa ohjelmistossa määritetyt käskyt.
Cron	Unix-pohjaisten käyttöjärjestelmien ajastuspalvelu.
DDoS	Distributed Denial of Service. Hajautettu palvelunestohyökkäys.
EEPROM	Electronically Erasable Programmable Read-Only Memory. Uudelleenkirjoitettava muisti, jonka sisältö säilyy silloinkin, kun virta kytketään pois.
GPIO	General Purpose Input/Output. Yleiskäyttöinen pinni.
GPU	Graphics Processing Unit. Grafiikkasuoritin. Komponentti, joka suorittaa grafiikan renderointia CPU:ta tehokkaammin.
I2C	Inter-Integrated Circuit. Kaksisuuntainen tiedonsiirtoväylä, jossa tiedonsiirto tapahtuu sarjamuotoisesti. Toteutetaan kahdella johtimella, jotka ovat SDA ja SCL.
IDE	Integrated Development Environment. Ohjelmointiympäristö.
IEEE	Institute of Electrical and Electronics Engineers. Kansainvälinen organisaatio, joka muun massa määrittelee standardeja, järjestää tieteellisiä konferensseja ja julkaisee tieteellisiä julkaisuja.
IETF	Internet Engineering Task Force. Organisaatio, joka vastaa Internetissä käytettävien protokollien standardoinnista. Julkaisee ja päivittää RFC-standardeja.
IoT	Internet of Things. Esineiden internet.

LoRa	Long Range. Pitkän kantaman verkkoteknologia.
LoRaWAN	Long Range, Wide Area Network. Langaton tiedonsiirtoverkko.
LTO	Linear Tape-Open. Varmuuskopioinnissa käytettävä nauhatyyppi.
MQTT	MQ Telemetry Transport. Protokolla laitteiden väliseen viestintään.
MU-MIMO	Multi-User, Multiple-Input, Multiple-Output. Vapaasti suomennettuna "Useita käyttäjiä, useita syötteitä, useita lähtöjä".
NC	Normally Closed. Magneettikytkintyyppi, jonka silmukka johtaa kun magneetti on lähellä.
NFC	Near-Field Communication. Erittäin lyhyen kantaman tiedonsiirtomenetelmä.
NO	Normally Open. Magneettikytkintyyppi, jonka silmukka ei johda kun magneetti on lähellä.
OBD	On-Board Diagnostics. Ajoneuvon diagnostiikkajärjestelmä. Järjestelmä mahdollistaa pääsyn ajoneuvon tuottamaan dataan.
OSHW	Open Source Hardware. Laitteisto, josta on vapaasti saatavissa tarkat kytkentäkaaviot ja muut tiedot laitteiston kokoonpanoa varten.
RAM	Random Access Memory. Keskusmuisti, johon tallennetaan tietoja väliaikaisesti.
RFC	Request for Comments. IETF-organisaation julkaisemat standardit.
SBC	Single Board Computer. Yhden piirilevyn tietokone.
SCL	Serial Clock Line. Kellolinja.
SDA	Serial Data Line. Datalinja.

SDHC	Secure Digital High Capacity. Muistikorttityyppi.
SIG	Special Interest Group. Yhteisö, jota yhdistää sama mielenkiinnon kohde.
SOC	System on Chip. Järjestelmäpiiri, joka sisältää useita toimintoja (CPU, RAM, WLAN ja Bluetooth).
SPI	Serial Peripheral Interface. Kaksisuuntainen tiedonsiirtoväylä, jossa tiedonsiirto tapahtuu sarjamuotoisesti. Toteutetaan neljällä johtimella.
SQS	Simple Queue Services. Amazonin tuottama jonopalvelu, mahdollistaa esimerkiksi IoT-laitteiden tuottaman datan vastaanottamisen.
SSH	Secure Shell. Salatun tiedonsiirron mahdollistava protokolla.
TCP/IP	Transmission Control Protocol / Internet Protocol. Internet-tietoliikenteessä käytettävä protokollapino.
TLS	Transport Layer Security. Salausprotokolla. Voidaan käyttää esimerkiksi selväkielisen HTTP-tiedonsiirron muuntamisessa salatuksi HTTPS-tiedonsiirroksi.
UART	Universal Asynchronous Receiver Transmitter. Sarjaliikennepiiri.
UPS	Uninterruptible Power Supply. Keskeytymätön virransyöttö.
WEBGUI	Web Graphical User Interface. Selainpohjainen käyttöliittymä.
WLAN	Wireless Local Area Network. Langaton lähiverkko.

Sisältö

1 Johdanto

Lähes kaikki yritykset ja organisaatiot käyttävät päivittäisessä toiminnassaan sähköisiä palveluita. Sähköiset palvelut vaativat toimiakseen palvelimia, tietoliikennelaitteita, levyjärjestelmiä sekä muita konesaleissa sijaitsevia oheisjärjestelmiä. Näiden laitteistojen toimivuuden ja parhaan suorituskyvyn varmistamiseksi konesalien fysikaalisten olosuhteiden tulee olla laitevalmistajien määrittämien ohjearvojen mukaiset. Vaikka konesalien fysikaaliset olosuhteet pidettäisiin koneellisesti asianmukaisina, ne voivat äkillisesti muuttua esimerkiksi ilmastointijärjestelmän laiterikon vuoksi. Myös muut ennakoimattomat tapahtumat, kuten sähkölaitteiden sytyttämät tulipalot ovat mahdollisia myös konesaliympäristöissä. Näiden seikkojen vuoksi konesalien olosuhteita on pystyttävä jatkuvasti monitoroimaan ja olosuhteiden muutoksista on saatava herätteet tapahtumahetkellä, ennen kuin heikentyneet olosuhteet vaikuttavat palveluiden toimintaan ja tietojen saatavuuteen.

Tietojen saatavuuden lisäksi organisaation kannalta on tärkeää, että tallennettujen tietojen eheys ja luotettavuus voidaan taata. Datan suojaaminen tapahtuu useiden suojauskerrosten avulla. Useat suojausmenetelmistä voidaan kuitenkin ohittaa, jos datan sisältävään laitteistoon päästään fyysisesti käsiksi. Tämän vuoksi on tärkeää havaita, jos räkkikaappeihin sijoitettuihin laitteisiin on kajottu luvatta.

Tämän tutkielman pääkysymys on "Miten IoT-teknologia auttaa ylläpitäjää konesaliympäristön monitoroinnissa?". Pääkysymykseen vastataan kahden alakysymyksen avulla, jotka ovat "Mitkä ovat yleisiä konesaliympäristön olosuhteisiin epäsuotuisasti vaikuttavia tekijöitä ja tapahtumia?" ja "Millainen IoT-konstruktio mahdollistaa epäsuotuisaksi muuttuneiden olosuhteiden havaitsemisen mahdollisimman nopeasti vahinkojen minimoimiseksi?". Tutkielman teoreettisen osuuden tutkimusmenetelmänä käytetään kirjallisuuskatsausta. Kirjallisuuskatsauksen lähteinä on pääsääntöisesti käytetty tieteellisiä artikkeleita ja julkaisuja, mutta työn luonteen vuoksi myös standardien ja muiden teknisten dokumenttien käyttäminen lähteinä on välttämätöntä. Tutkielman empiirisessä osassa käytetään konstruktiivista tutkimusotetta, jota sovelletaan IoT-laitteeksi luokiteltavan mittauslaiteen suunnittelussa ja toteuttamisessa. Mittauslaite mahdollistaa konesalin ilmankosteuden ja huoneilman lämpötilan mittauksen. Mittauslaite pystyy havaitsemaan myös huoneil-

man kohonneen savupitoisuuden sekä havaitsee, jos konesalilaitteistoa sisältävän räkkikaapin ovi avataan. Mittauksen jälkeen data siirretään Zabbix-monitorointisovellukseen.

Työn tuloksena saavutetaan kokonaisuus, joka auttaa konesaliympäristön ylläpitäjää monitoroimaan konesaliympäristön olosuhteita. Epäsuotuisat muutokset olosuhteissa kyetään havaitsemaan olosuhteiden muuttumishetkellä ja raja-arvot ylittävistä muutoksista lähetetään hälytykset sähköpostitse. Lisäksi ylläpitäjä pystyy tarkkailemaan konesalin olosuhteita visuaalisten näkymien avulla ja vertaamaan nykyisiä olosuhteita aiempiin mittaustuloksiin.

Tutkielman luvussa 2 esitellään esineiden Internetin (Internet of Things, IoT) määritelmiä sekä referenssimalleja, joiden avulla IoT-laitteiden tuottamaa dataa voidaan käsitellä ja hyödyntää. Esitellään myös laitteistoa ja teknologioita, jotka mahdollistavat mittausten tekemisen ja tiedonsiirron IoT-laitteiden ja muiden laitteiden välillä. Valmiilla IoT-laitteilla on laajasti käyttökohteita, mutta myös ongelmakohtia jotka vaativat huomiointia.

Luku 3 muodostaa yleiskuvan konesalin erityispiirteistä, toiminnoista ja laitteistosta. Luvusta ilmenee eri laitevalmistajien asettamia raja-arvoja, jotka määrittävät konesalin olosuhteita. Esitetään lyhyesti, millaisia ongelmia epäsuotuisat olosuhteet aiheuttavat ja minkälaisilla tavoilla ongelmia pyritään ennaltaehkäisemään. Luvussa käsitellään myös ratkaisuja, jotka auttavat monitoroinnin toteuttamisessa ja esitetään Zabbix-monitorointisovelluksen komponentit, jotka mahdollistavat datan keräämisen IoT-laitteilta sekä datan jatkojalostamisen visuaaliseen muotoon.

Luvussa 4 esitetään tuotekehitysmenetelmiä, joita hyödyntämällä laitteiston ja sovelluksen kehitystyöstä saadaan määrämuotoista ja hallittua. Kuvataan eri menetelmien periaatteet ja esitellään, mitkä ovat sopivia käyttökohteita kullekin tuotekehitysmenetelmälle.

Luvussa 5 esitellään toteutettavan mittauslaitteen ominaisuudet, toimintaperiaate, laitteistokokoonpano sekä mittauslaitteeseen kehitetty ohjelmisto. Toteutetaan mittauslaite spiraalimallin mukaisesti ja esitetään, kuinka kehitystyö etenee iteraatiokierrosten myötä. Esitetään myös, kuinka mittauslaite integroidaan Zabbix-monitorointijärjestelmään ja kuinka Zabbixin ominaisuuksien avulla data muunnetaan konesalin ylläpitäjän näkökulmasta hyödylliseen muotoon.

Luvussa 6 pohditaan, kuinka mittauslaitteen suunnittelu ja sen ominaisuuksien toteutus onnistuivat sekä arvioidaan, kuinka mittauslaitetta voisi jatkokehittää eri näkökulmista. Luku 7 sisältää lyhyen yhteenvedon tämän tutkielman sisällöstä.

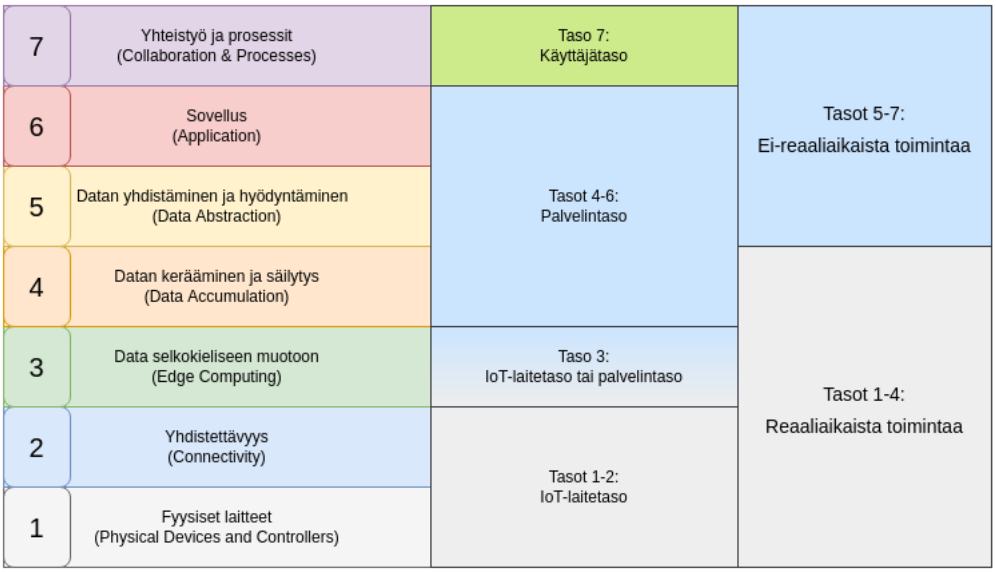
2 Esineiden Internet, Internet of Things

Termille Internet of Things (IoT), eli esineiden Internetille ei ole olemassa yhtä ainoaa virallista määritelmää. Yhtäläistä Oxfordin sanakirjan [79] ja Sanastokeskuksen [88] määritelmässä on se, että esinettä voidaan kutsua IoT-laitteeksi silloin, kun esineellä on IP-osoite, jolloin se voi lähettää tietoa muille verkossa oleville laitteille sekä vastaanottaa tietoa muilta laitteilta. Yleisesti IoT-laitteiksi käsitetään sellaiset laitteet, jotka eivät muistuta perinteisiä päätelaitteita, kuten PC-tietokoneita, älypuhelimia tai palvelimia. Tätä näkökulmaa tukevat myös Poornachandran et al. [81] artikkelissaan esittämät huomiot, joiden mukaan verkot eivät koostu enää pelkästään perinteisistä tietokoneista, vaan verkkohin on liitetty suuria määriä ei-perinteisiä verkkolaitteita, kuten televisioita ja jääkaappeja. Sivanathan et al. [91] puolestaan mainitsevat laite-esimerkkeinä mm. älylukot, termostaatit ja lääketieteelliset laitteet.

Edellä mainittujen tulkintojen perusteella IoT-laitteen määritelmä ei ole yksiselitteinen, koska tulkitsijasta riippuen näitä laitteita voivat olla käytännössä mitkä laitteet tahansa, joilla on kyvykkyys liittyä verkkoon. IoT-laitteiksi luokiteltavissa olevien laitteiden määrä on ollut voimakkaassa kasvussa kuluneen kymmenen vuoden aikana. Laitemäärät ovat niin suuria, että tarkkoja arvioita laitemääristä ei ole pystytty tekemään, mutta suuruusluokka on useita miljardeja. Arvioinnin vaikeutta kuvastaa hyvin se, että vuoden 2011 ennusteessaan Cisco [36] ennusti IoT-laitemäärän olevan vuonna 2020 50 miljardia, kun vuonna 2017 Gartner [41] ennusti laitemäärän olevan vuonna 2020 noin 20 miljardin luokkaa.

2.1 IoT referenssi- ja arkkitehtuurimallit

Eri toimijat ovat luoneet referenssi- ja arkkitehtuurimallejaan siitä, miten IoT-laitteiden tulisi toimia yhdessä muiden palveluiden kanssa, jotta laitteiden keräämää dataa voitaisiin hyödyntää. Kuvassa 2.1 on nähtävissä Cisco Systemsin [19] IoT-referenssimallin kerrokset. Malli koostuu seitsemästä kerroksesta, jotka kuvaavat koko prosessin alkaen datan keräämisestä IoT-laitteen avulla ja päättyen prosessoidun datan hyödyntämiseen loppukäyttäjän toimesta.



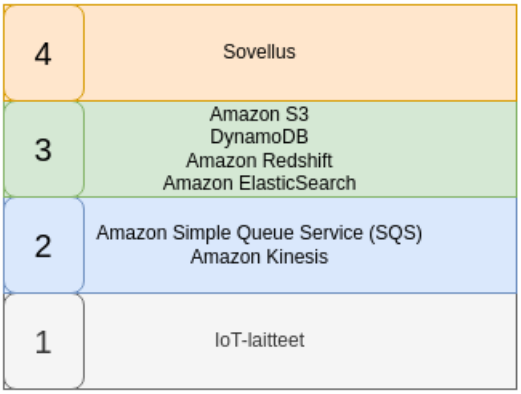
Kuva 2.1: Ciscon IoT-referenssimalli, muokattu [19]

Tasojen 1-2 toiminnot tapahtuvat IoT-laitteessa. Näissä tasoissa fyysiset laitteet, kuten esimerkiksi lämpömittarit, keräävät datan ja lähettävät sen eteenpäin määrämuotoisesti. Lähetys voi tapahtua esimerkiksi WLAN-verkossa. Tasolla 3 tapahtuva datan muuttaminen selkokieliseen muotoon voi tapahtua järjestelmäarkkitehtuurista riippuen joko IoT-laitteessa tai palvelintasolla. Tasoilla 4-6 data otetaan vastaan kohdejärjestelmään. Data tallennetaan ja sitä voidaan rikastaa muista lähteistä saadulla datalla. Datan käyttäminen mahdollistetaan sovelluskerroksen avulla, jonka jälkeen tasolla 7 oleva käyttäjätaso pystyy hyödyntämään kerättyä dataa.

Ciscon mallin mukaisesti datan kerääminen ja sen tallentaminen on reaaliaikaista toimintaa. Datan hyödyntäminen on ei-reaaliaikaista, koska useissa tapauksissa dataa voidaan kerätä pitkiäkin aikoja ennen sen jalostamista hyötykäyttöön.

Suuren mittakaavan IoT-ratkaisuissa dataa kerätään useista lähteistä. Tällöin on mahdollista, että dataa saapuu lyhyessä ajassa suuria määriä. Tämä voi aiheuttaa haasteita datan vastaanottamisessa ja sen prosessoimisessa. Patel [80] esittelee kuvassa 2.2 kuvatun arkkitehtuurimallin, joka perustuu kaupallisiin Amazon Web Services (AWS) -palveluihin. Tässä mallissa ei esitetä toimintoja yhtä tarkalla tasolla kuin Ciscon mallissa. Huomattava ja oleellinen ero on tasolla 2, jossa Amazon Simple Queue Service (SQS) tai Amazon Kinesis ottavat datan vastaan IoT-laitteista. Tämän jälkeen mainitut palvelut siirtävät datan eteenpäin tietokantaan tallennettavaksi. Tämän välikerroksen hyötynä on tietokannan rasituksen pienentäminen ja

dataa voidaan käsitellä halutulla tavalla jo ennen sen viemistä tietokantaan. Lisäksi hyötynä on datahäviön estäminen, koska mainitut palvelut on kehitetty siten, että ne voivat puskuroida ja tallentaa datan välimuistiin silloin, jos tietokanta ei ole väliaikaisesti toiminnassa.



Kuva 2.2: Malli datan IoT-datan keräämiseen ja hyödyntämiseen Amazon AWS:n tuotteiden avulla, muokattu [80]

Ciscon ja Petelin toimintamallit yhdistämällä on mahdollista toteuttaa järjestelmä, joka pystyy käsittelemään suuria määriä IoT-laitteista saapuvaa dataa. Vaikka Patelin malli perustuu Amazonin palveluihin, sitä voidaan soveltaa muissakin tilanteissa. Oleellista on suunnitella kokonaisuus siten, että järjestelmän kaikki komponentit mitoitettu oikein käsiteltävän datan määrälle. Jos mikä tahansa osa datan käsittelyketjussa ei kykene toimimaan halutulla nopeudella, siitä muodostuu pullonkaula koko järjestelmälle. Tällöin järjestelmä ei toimi käyttäjän näkökulmasta halutulla tavalla. On myös huomioitava, että pelkkä datan tallentaminen tietokantaan ei ole useimmissa tapauksessa tarkoituksenmukaista, vaan tallennettua dataa halutaan hyödyntää. Ma et al. [71] esittävät artikkelissaan ongelman, joka voi ilmetä erittäin laajoissa IoT-ympäristöissä. Artikkelissa tutkitaan miljoonan liikennevälineen aiheuttamaa kuormitusta, jotka lähettävät 100 tavua dataa kerran minuutissa. Vuorokauden aikana kertyy täten n. 1,5 miljardia kirjoitustapahtumaa tietokantaan, kokonaiskirjoitusmäärän ollessa 144 gigatavua päivässä. Jotta tietokannan suorituskyky pysyy hyvänä ja kyselyiden hakuajat mahdollisimman alhaisena, on tietokannan asianmukaiseen indeksointiin kiinnitettävä erityistä huomiota. Indeksoinnilla tarkoitetaan tässä yhteydessä sitä, että tietokannan sisältämä tieto järjestetään siten, että halutun datan löytäminen on tietokannasta mahdollisimman loogista.

2.2 IoT-teknologiat ja laitteistot

Tässä alaluvussa käsitellään yleisesti käytettyjä langattomia teknologioita, joita hyödyntämällä mahdollistetaan erityyppisten IoT-laitteiden langaton kommunikaatio muiden laitteiden kanssa. Lisäksi esitellään kohteita, joissa kunkin langattoman teknologian käyttäminen on tarkoituksenmukaista.

Alaluvussa esitellään myös Arduino Unon, ESP32:n ja Raspberry Pi:n teknisiä ominaisuuksia. Nämä kolme kehitysalustaa on valittu esiteltäviksi niiden hyvin laajan suosion vuoksi. Lisäksi näiden alustojen tekniset ominaisuudet eroavat toisistaan merkittävästi. Kehitysalustojen ominaisuuksia voidaan lisätä käyttötarpeen mukaan sensoreiden ja aktuaattoreiden avulla.

2.2.1 Langattomat teknologiat

Tässä käsiteltävät langattomat teknologiat eivät muodosta kokonaisvaltaista listausta kaikista vaihtoehdoista, vaan tarkoituksena on tehdä katsaus yleisimmin käytössä oleviin teknologioihin siten, että niiden väliset IoT-laitteiden näkökulmasta oleelliset eroavaisuudet tulevat esiin.

5G

5G on tämän tutkielman kirjoitushetkellä matkapuhelinverkkojen uusin sukupolvi. 5G:n nimi (fifth generation, viides sukupolvi) juontuu ajattelutavasta, jossa eri aikakausien teknologiat on jaettu sukupolvittain. 5G-standardien takana on järjestö 3GPP, joka koostuu seitsemän järjestön yhteistoiminnasta [3]. 5G-standardin ensimmäinen julkaisuversio, versionumeroltaan 15, on julkaistu vuonna 2018 [1]. Versiojulkaisun epälooginen numerointi johtuu 3GPP-järjestön historiasta. Järjestö on alunperin perustettu luomaan kolmannen sukupolven teknologioiden standardeja ja versionumero on kasvanut juoksevasti nykyhetkeen asti, jolloin kirjoitushetkellä kehitettävän julkaisun versionumero on 18 [2]. Ghoshin et al. [43] mukaan julkaisuversion ominaisuudet ovat painottuneet uusien käyttötarpeiden mukaisesti. 5G:n viidennentoista julkaisuversion painopisteenä oli yhteysnopeuden kasvattaminen merkittävästi aiempiin verkkosukupolviin verrattuna. Julkaisuversiossa 16 painopisteenä oli kehittää IoT-laitteille merkityksellisiä ominaisuuksia, kuten mahdollistaa yksityiset ja paikalliset 5G-verkot sekä kehittää paikannukseen liittyviä toimintoja. Julkaisun 17 myötä 5G:n ulottuvuuteen on tehty parannuksia, muun muassa satelliittiverkot huomioiden.

5G-verkot käyttävät laajasti useita taajuusalueita. Traficomin antamien taajuustietojen [67] mukaan Suomessa 5G-dataliikenne hyödyntää 700 MHz:n, 2 GHz:n, 2.6 GHz:n, 3.5 GHz:n sekä 26 GHz:n taajuuksia. 5G-verkon tiedonsiirtonopeus ja kantavuus riippuu käytettävästä taajuudesta. Nopeimmat tiedonsiirtonopeudet voidaan toteuttaa 26 GHz:n taajuuksilla toimivilla verkoilla. Korkeaa taajuutta käytettäessä kantomatka on vain noin sata metriä, jolloin laajojen verkkojen rakentaminen on haastavaa. Korkean taajuuden haittapuolena on myös heikko läpäisykyky, jolloin esimerkiksi rakennusten seinät heikentävät signaalin voimakkuutta. Vastaavasti matalilla taajuuksilla kantomatka ja läpäisykyky ovat huomattavasti paremmat, mutta yhteyden nopeus on merkittävästi alhaisempi. Nokian taajuusoppaan [76] mukaan 1-6 GHz taajuusalueilla tiedonsiirtonopeus ja kantomatka ovat ideaaliset useimpiin käyttötarkoituksiin.

5G-verkkojen maksiminopeuden määrittäminen on vaikeaa useiden taajuusalueiden ja teknologian jatkuvan kehittämisen myötä. Vuonna 2015 Samsung ilmoitti saavuttaneensa hetkellisesti 7.5 Gb/s ja vakaasti 1.2 Gb/s nopeuden [44]. Vuonna 2021 Nokia puolestaan ilmoitti saavuttaneensa 4.5 Gb/s nopeuden kaupallisesti saatavissa olevilla laitteilla [77].

Bluetooth

Bluetooth-standardien takana oleva organisaatio Bluetooth Special Interest Group (SIG) perustettiin vuonna 1998 Ericssonin, Nokian, IBM:n, Toshiban ja Intelin toimesta. Tämän pro gradun kirjoitushetkellä Bluetooth SIG:iin kuuluu maailmanlaajuisesti yli 36 000 yritystä [14]. Standardin ensimmäinen versio on julkaistu vuonna 1999 [45]. Bluetoothin kehitys on aktiivista ja siitä on julkaistu vuosien aikana useita versioita, joiden myötä Bluetooth on saanut uusia ominaisuuksia ja muun muassa tietoturvallisuuteen on kiinnitetty enemmän huomiota. Viimeisin versio on 5.3, joka on julkaistu vuonna 2021. IoT-laitteiden kannalta vuonna 2010 julkaistu versio 4.0 on merkittävä, jolloin standardiin sisällytettiin Bluetooth Low Energy (BLE) [16]. BLE:n tavoitteena on nimensä mukaisesti saavuttaa mahdollisimman pieni energiankulutus, joka mahdollistaa pidemmän käyttöajan akku- ja paristokäyttöisille laitteille.

Bluetooth toimii 2,4 GHz taajuusalueella. Kantavuudeksi ihanteellisissa olosuhteissa Bluetooth SIG ilmoittaa yli yhden kilometrin [15]. Käytännössä kantavuuteen vaikuttavat useat tekijät, kuten vastaanottimen herkkyys sekä lähettäjän ja vastaanottajan antennien vahvistimet. Lisäksi ympäristön asettamat olosuhteet, kuten mahdolliset seinät ja niiden materiaali ovat merkittäviä tekijöitä todellisen kanto-

matkan muodostamisessa.

Käytännössä kaikissa älypuhelimissa, tablet-tietokoneissa sekä kannettavissa tietokoneissa on tuki Bluetoothille. Myös useissa nykyaikaisissa autoissa on Bluetooth. Bluetooth mahdollistaa muun muassa äänen suoratoiston, esimerkiksi älypuhelimesta auton audiojärjestelmään tai kannettavasta tietokoneesta langattomiin kuulokkeisiin. Äly- ja urheilukellot tukevat tyypillisesti on Bluetooth-tiedonsiirtoa, jonka avulla kellon mittaamat liikuntatiedot voidaan siirtää kellosta älypuhelimeen ja tämän jälkeen esimerkiksi 5G tai WLAN-verkkoja hyödyntäen erilaisiin pilvipalveluihin. Äänen- ja tiedonsiirron lisäksi Bluetoothia voidaan käyttää sisäpaikannukseen. Oosterlinck et al. [78] esittelevät artikkelissaan Belgiassa sijaitsevaan kauppakeskukseen käyttöönotetun Bluetooth-skannereihin perustuvan paikannusjärjestelmän. 56 skanneria rekisteröivät noin 19 000 yksilöllistä Bluetooth-laitetta 19 vuorokauden pituisen seurantajakson aikana. Kerätyn datan avulla voitiin kartoittaa, miten asiakkaat liikkuivat kauppakeskuksessa ja kauanko he käyttivät siellä aikaa.

Vuodesta 2019 alkaen levinnyt SARS-CoV-2 -koronavirus on aiheuttanut maailmanlaajuisen COVID-19 -pandemian. Tämä pandemia on tuonut Bluetoothille uuden käyttökohteen. Useat valtiot ovat kehittäneet älypuhelimiin sovelluksen, jonka avulla pyritään jäljittämään koronavirukselle altistuneita. Suomessa käytetään Terveyden ja hyvinvoinnin laitoksen tuottamaa Koronavilkku-sovellusta [94]. Koronavilkku kerää BLE:n avulla tietoja muista laitteista, jotka ovat olleet kuuluvuusalueella. Jos henkilöllä todetaan koronavirustartunta, hän saa terveysviranomaisilta avauskoodin, joka syötetään Koronavilkkuun. Avauskoodin syöttämisen jälkeen aiemmin kuuluvuusalueella olleet Koronavilkku-käyttäjät saavat tiedon mahdollisesta altistumisesta koronavirukselle. Kommunikointi puhelimen ja Koronavilkun taustajärjestelmien välillä tapahtuu esimerkiksi 5G tai WLAN-verkkojen avulla.

LoRa ja LoRaWAN

Long Range, Wide Area Network (LoRaWAN) on verkko, joka on suunniteltu erityisesti IoT-laitteiden tiedonsiirtoa varten. Nimensä mukaisesti LoRa:n erityisenä vahvuutena on pitkä kantavuus, ihanteellisissa olosuhteissa jopa kymmeniä kilometrejä. LoRaWAN-teknologiaa kehittää vuonna 2015 perustettu LoRa Alliance, johon kuuluu tämän pro gradun kirjoitushetkellä yli 500 jäsenyritystä [68]. LoRaWAN 1.0 oli ensimmäinen virallinen spesifikaatio, joka julkaistiin vuonna 2015. Versionumerointi voi tuntua epäloogiselta ja harhaanjohtavalta, koska versio 1.1 on julkaistu vuonna 2017, mutta viimeisin versiojulkaisu 1.0.4 on tehty vuonna 2020. Versio 1.1

on suunniteltu siten, että se on yhteensopiva 1.0.x -versioiden kanssa [69].

LoRaWAN-verkkoja voidaan toteuttaa erilaisiin tarpeisiin. Pienen mittakaavan tarpeisiin verkko voidaan toteuttaa paikallisesti, jolloin LoRa-laitteet yhdistetään paikalliseen yhdyskäytävään, joka tyypillisesti välittää liikenteen TCP/IP-verkkoon. Mikäli tarve on kaupungin tai maanlaajuinen, voidaan Suomessa hyödyntää Digita Oy:n ylläpitämää verkkoa. Digita Oy:n [27] antamien tietojen mukaan kaupallisen verkon ulkokuuluvuus kattaa lähes koko Suomen, pois lukien pienet katvealueet, joita on lähinnä paikoitellen Lapin erämaa-alueilla. Myös sisäkuuluvuus on varsin hyvällä tasolla, erityisesti kaupunkien alueilla.

LoRaWAN-verkko soveltuu hyvin laitteille, jotka lähettävät tai vastaanottavat pieniä datamääriä harvakseltaan. Esimerkkejä soveltuvista käyttökohteista ovat esimerkiksi säiliöiden täyttöasteen mittaukset sekä vesimittareiden kulutustiedot. Kohteisiin, jotka vaativat reaaliaikaista kommunikaatiota, LoRaWAN ei ole soveltuva vaihtoehto, koska datasanomia voidaan siirtää vain useiden minuuttien välein [102].

The Things Network -organisaation mukaan LoRa-laiteille on useita mahdollisia taajuusalueita, koska eri valtiot asettavat säännöksiä, joiden puitteissa laitteiden on toimittava. Suomessa ja muissa EU-maissa sallitut taajuusalueet ovat EU863-870 ja EU443. [101] Yhteistä kaikille LoRa-laitteille on se, että ne toimivat alle 1 GHz taajuuksilla.

Dönmez ja Nigussie [30] tutkivat versiota 1.1 ja sen taaksepäin yhteensopivuutta vuonna 2016 julkaistun version 1.0.2 kanssa. Heidän mukaan yhteensopivuus aiheuttaa tietoturvaan liittyviä haasteita. Versiossa 1.1 on korjattu aiemmissa versioissa havaittuja tietoturvapuutteita, mutta näitä haavoittuvuuksia voi silti hyödyntää yhteensopivuuden vuoksi.

IEEE 802.11

Institute of Electrical and Electronics Engineers (IEEE) -järjestön [48] julkaisema 802.11 on standardi Wireless Local Area Network (WLAN) -verkoille. Standardin ensimmäinen versio on julkaistu vuonna 1997. Standardia on vuosien aikana jatkokehitetty aktiivisesti. Jatkokehitys on johtanut siihen, että 802.11-standardin alle on julkaistu useita eri protokollaversioita. Vaikka protokollat kuuluvat saman standardin alle, ne eroavat toisistaan merkittävästi eivätkä täten ole kaikissa tapauksissa yhteensopivia toistensa kanssa. Esimerkiksi protokollat 802.11a ja 802.11b ovat nimensä ja julkaisuajankohdan perusteella lähellä toisiaan, mutta siitä huolimatta ne eroavat toisistaan muun muassa käytettävän taajuuden osalta [49, 53]. Varhai-

siin protokollaversioihin pohjautuvat laitteet kärsivät myös keskinäisistä yhteensopivuusongelmista, koska tuolloin IEEE ei määrännyt laitevalmistajia validoimaan laitteitaan siten, että niiden standardienmukaisuus varmistettaisiin. Tätä ongelmaa ratkaisemaan perustettiin vuonna 1999 Wi-Fi Alliance, joka on voittoa tavoittelematon järjestö [107]. Järjestö myöntää luvan "Wi-Fi Certified-logon käyttämiseen niissä tuotteissa, jotka läpäisevät standardienmukaisuustestit.

Laajimmassa käytössä olevat standardit ovat olleet 802.11a, 802.11b, 802.11g, 802.11n ja 802.11ac. Tämän pro gradun kirjoitushetkellä 802.11ax standardiin perustuvat laitteet ovat yleistymässä. IoT-käyttötarkoitusta ajatellen on luotu standardi 802.11ah. Tähän protokollaan perustuvat laitteet kuluttavat muun muassa vähemmän energiaa ja kantavuus on pidempi, kun vertailukohteena on protokollan muut versiot. 802.11ah-protokollaan perustuvat laitteet eivät kuitenkaan ole yleistyneet. Yksi syy tähän voi olla muista poikkeava radiotaajuus, jonka vuoksi käytettävissä tukiasemissa pitäisi olla erillinen radiolähetin tätä protokollaa varten. Taulukossa 2.1 on esitelty perustietoja edellä mainituista protokollista. Mainitut tiedot on kerätty kunkin protokollan virallisesta, IEEE:n julkaisemasta standardista. Viittaukset standardiin on mainittu kunkin protokollan jälkeen.

Abdelrahman et al. [4] vertailevat artikkelissaan 802.11b, 802.11g, 802.11n ja 802.11ac-protokollien ominaisuuksia ja tuovat esiin niiden eroavaisuuksia. Suorituskyvyn kehittymisen lisäksi he mainitsevat muutamia seikkoja, joilla protokollat erottuvat toisistaan. 802.11n ja 802.11ac -protokollat sisältävät ominaisuuden, joka mahdollistaa suuntaavan säteenmuodostuksen (beamforming). Tällöin säteilyä ei suunnata tasaisesti kaikkiin suuntiin vaan säteily suunnataan siihen suuntaan, missä vastaanottaja sijaitsee. Ac-protokollaan kuuluu Multi-User, Multiple-Input, Multiple-Output (MU-MIMO) -ominaisuus. Vapaasti suomennettuna MU-MIMO tarkoittaa "Useita käyttäjiä, useita syötteitä, useita lähtöjä". Tällöin useampi päätelaite voi muodostaa yhtäaikaisen yhteyden tukiasemaan, jolloin päätelaitteiden ei tarvitse odottaa vuoroa. He tuovat myös esiin, että 5 GHz taajuudella toimivat protokollat ovat vähemmän alttiita häiriöille kuin 2,4 GHz taajuudella toimivat, koska 2,4 GHz taajuusalueella toimii myös muita langattomia verkkoja, jotka voivat aiheuttaa häiriöitä WLAN-verkoille.

Taulukko 2.1: Perustietoja tärkeimmistä 802.11-protokollista

Protokolla	Julkaisu-vuosi	Taajuus (GHz)	Tiedonsiirtonopeus, teoreettinen maksimi (Mb/s)	Kantavuus (metriä)
802.11 [48]	1997	2,4	2	100
802.11a [49]	1999	5	54	120
802.11b [53]	1999	2,4	11	140
802.11g [54]	2003	2,4	54	140
802.11n [55]	2009	2,4 ja 5	600	250
802.11ac [50]	2013	5	3467	250
802.11ah [51]	2016	Alle 1	78	1000
802.11ax [52]	2021	2,4 ja 5	9608	250

ZigBee

ZigBee on verkkoteknologia, joka on kehitetty vähävirtaisten IoT-laitteiden lyhyen kantaman tiedonsiirtoa varten. ZigBee-teknologiaa kehittää vuonna 2002 perustettu Connectivity Standards Alliance (CSA), joka tunnettiin aiemmin nimellä ZigBee Alliance. CSA:han kuuluu tämän pro gradun kirjoitushetkellä noin 400 jäsenyritystä [22]. Vuonna 2004 allianssi hyväksyi ZigBee-protokollasta luonnosversion 1.0 [119]. Uusin versio ZigBee-protokollasta on 3.0, joka on taaksepäin yhteensopiva aiempien versioiden kanssa [95].

Teknisesti ZigBee perustuu IEEE-järjestön julkaisemaan 802.15.4 -standardiin. ZigBee voi toimia alle 1 ja 2,4 GHz taajuusalueilla. Alle 1 GHz taajuusalueita käytettäessä tarkempi taajuus Euroopassa on 868,3 MHz ja Yhdysvalloissa 902-928 MHz [118]. Pääsääntöisesti ZigBee-laitteiden kantomatka on 10-100 metriä, mutta alle 1 GHz taajuuksilla toimivien laitteiden kantomatka voi olla jopa kilometrin [23].

ZigBee mahdollistaa erityyppisten verkkotopologioiden käyttämisen käyttökohteen ja tarpeen mukaan. Verkkotopologian valinta vaikuttaa verkon monimutkaisuuteen, vikasietoisuuteen sekä mahdollisiin pullonkauloihin, jotka voivat aiheuttaa suorituskykyyn liittyviä ongelmia.

Protokollan mukaan verkossa olevilla laitteilla voi olla yksi tai useampi rooli [118]. Jollakin verkon laitteella on oltava koordinaattorin rooli, joka luo verkon ja

mahdollistaa pääsyn verkkoon muille laitteille. Osa verkon laitteista voi toimia reitittimenä, jolloin muiden laitteiden kommunikaatio kulkee tämän laitteen kautta. Jos laite ei toimi koordinaattorina eikä reitittimenä, laitteen roolina on toimia tavanomaisena päätelaiteena.

Tyypillisiä käyttökohteita ovat koti- ja rakennusautomaatioon liittyvät laitteet, kuten älyvalaisimet, vesivuotosensorit sekä liiketunnistimet. ZigBee-protokollaan perustuvien laitteiden yleistymisen myötä erityisesti kotiautomaatio on kehittynyt, koska monet aiemmat kotiautomaatiojärjestelmät ovat noudattaneet laitetoimittajien omia standardeja, jotka eivät ole yhteensopivia muiden laitetoimittajien kanssa. Tällöin kuluttajat ovat olleet sidottuja yhden laitevalmistajan tuotteisiin, mutta ZigBee:n tuoman yhteensopivuuden myötä eri laitevalmistajien laitteet kommunikoivat keskenään.

Monipuoliset verkkotopologiamahdollisuudet tarjoavat kuitenkin muitakin laajoja käyttömahdollisuuksia, kuten Leccensen esimerkki älykkäistä katuvaloista [66]. ZigBee-protokollaan perustuvalla hallinnalla saavutetaan katuvaloihin kaksisuuntainen kommunikaatio, jolloin niitä voidaan monitoroida yksilökohtaisesti mahdollisten vikatilanteiden varalta.

Yhteenveto langattomista teknologioista

Edellä on esitelty muutamia, IoT-laitteiden yleisesti käyttämiä langattomia teknologioita. Useat tekijät vaikuttavat siihen, mikä teknologia on käyttökohteen kannalta tarkoituksenmukaisin. Isoimmat käyttötarkoitukseen vaikuttavat tekijät ovat suurin sallittu virrankulutus, tarvittava kantomatka, siirrettävän datan määrä, reaaliaikaisuuden tarve sekä mahdolliset yhteensopivuusvaatimukset muiden laitteiden kanssa.

Sallittuun virrankulutukseen vaikuttaa merkittävästi se, onko laite akkukäyttöinen vai kytkettynä jatkuvasti sähköverkkoon. Akkukäyttöisissä laitteissa laitteen koko vaikuttaa siihen, kuinka isolla kapasiteetilla varustettu akku voidaan kytkeä laitteeseen. Erityisesti pienikapasiteettisten akkujen varassa toimivien laitteiden kohdalla on myös mietittävä, miten laite saadaan toimimaan halutulla tavalla mahdollisimman pienellä tiedonsiirtomäärällä ja siten, että laitteen radio voisi olla mahdollisimman paljon ei-aktiivisena virrankulutuksen pienentämiseksi.

Laitteiden toiminnalle on myös usein vaatimuksia, jotka käytännössä rajaavat pois useita käytettäviä teknologiavaihtoja. Näitä vaatimuksia esiintyy usein silloin, kun kehitettävän laitteen pitää pystyä kommunikoimaan olemassa olevien laittei-

den kanssa.

Laitteiden maantieteellinen sijainti ja liikkuvuuden tarve asettavat myös rajoituksia. Osa esitellyistä verkoista on rakennetaan paikallisesti, joillekin verkoille puolestaan eri operaattorit tarjoavat maan- ja maailmanlaajuinen kattavuuden. Tässä yhteydessä on punnittava myös kustannustehokkuutta. Verkon rakentaminen paikallisesti on kertaluontoinen kuluerä, kun operaattoreiden verkoissa on tyypillisesti kuukausimaksu tai datansiirron määrään perustuva maksu. Jos laitteet ovat pysyvästi paikallisen verkon kuuluvuuden piirissä, operaattorin verkon laajasta peitosta ei saada hyötyjä ja tällöin kuukausimaksut tulevat pidemmän ajan kuluessa kalliimmaksi kuin paikallisen verkon rakentaminen.

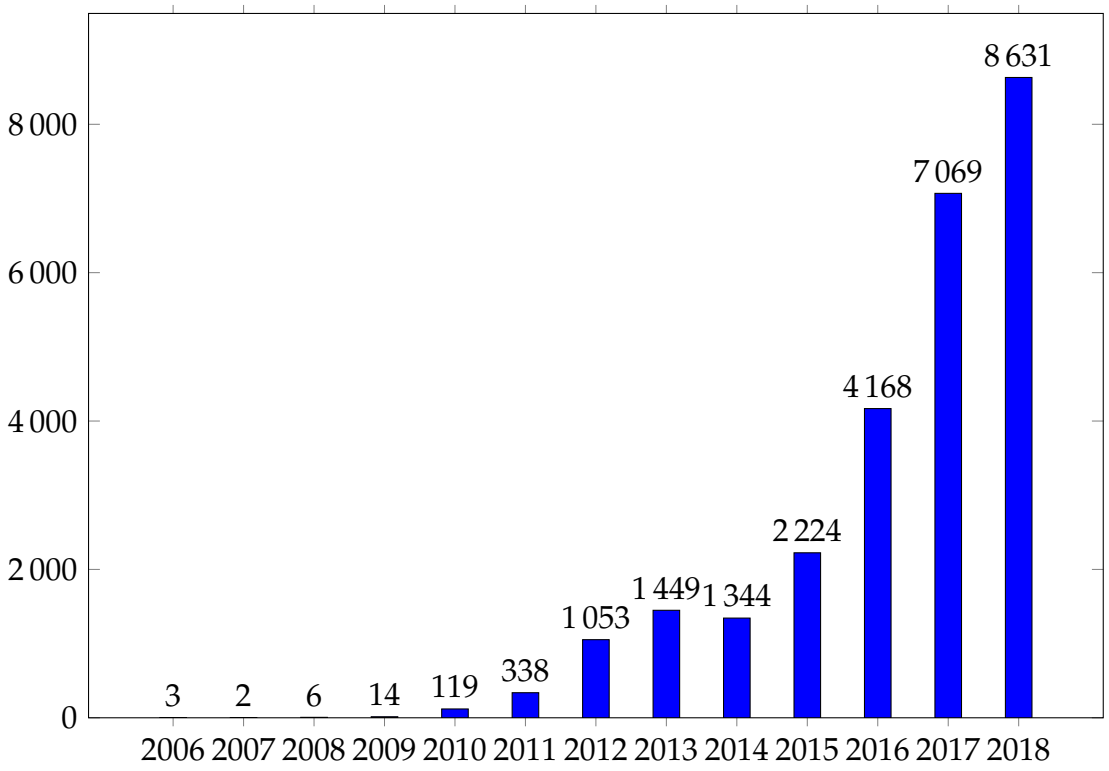
Taulukossa 2.2 kuvataan langattomien teknologioiden teknisiä ominaisuuksia. Teknologioiden välillä on merkittäviä eroja kantomatkassa sekä tiedonsiirron maksiminopeudessa, joten teknologiavalintaa tehtäessä on tärkeää huomioida tiedonsiirron vaatima kapasiteetti. Taulukosta voidaan havaita, että virrankulutuksen ollessa vähäistä tiedonsiirron nopeus on alhainen. Jos käyttötarve vaatii säännöllistä, nopeasti tapahtuvaa tiedonsiirtoa vähäistä suuremmalle datamäärälle, tämä tarve rajaa pois vähävirtaisimmat teknologiat. Useiden IoT-laitteiden siirtämän datan määrä on niin pientä, ettei teknologioiden tiedonsiirron nopeuteen liittyvät rajoitteet aiheuta käytännössä ongelmia.

Taulukko 2.2: Langattomien teknologioiden teknisiä ominaisuuksia

Teknologia	Virrankulutus	Kantomatka	Max. nopeus (sekunnissa)	Reaaliaikainen	Verkon peitto	Kuukausimaksu
5G	Hyvin suuri	Kilometrejä	Gigatavuja	Kyllä	Maanlaajuinen	Kyllä
Bluetooth	Suuri	Muutamia metrejä	Kymmeniä megatavuja	Kyllä	Paikallinen	Ei
Bluetooth (BLE)	Pieni	Kymmeniä metrejä	Megatavuja	Kyllä	Paikallinen	Ei
IEEE 802.11 (WLAN)	Hyvin suuri	Kymmeniä metrejä	Gigatavuja	Kyllä	Paikallinen	Ei
LoraWAN	Hyvin pieni	Kilometrejä	Kilotavuja	Ei	Maanlaajuinen	Kyllä
					Paikallinen	Ei
ZigBee	Pieni	Kymmeniä metrejä	Kilotavuja	Kyllä	Paikallinen	Ei

2.2.2 Kehitysalustat

Dachyar et al. [24] etsivät Scopus-viitetietokannasta IoT-aihepiiriin liittyviä artikkeleita ja konferenssijulkaisuja. Heidän mukaan Internet of Things -termiä käytettiin tieteellisissä julkaisuissa ensimmäisen kerran vuonna 2006. Vielä tuolloin aihepiiriin liittyviä tutkimuksia tehtiin kuitenkin hyvin vähän. Kaaviosta 2.3 voidaan havaita, kuinka voimakkaasti IoT-aiheisten tutkimusten määrä on noussut 2010-luvulla. Eräs todennäköinen vaikuttava tekijä IoT-laitteiden voimakkaan yleistymisen taustalla on kehitystyön merkittävä helpottuminen, jonka on mahdollistanut edullisten kehitysalustojen valikoiman kasvaminen. Useat toimijat, kuten Arduino AG, Espressif Systems Ltd ja Raspberry Pi -säätiö, ovat kehittäneet ja julkaisseet 2010-luvulla IoT-kehitysalustoja, jotka ovat saavuttaneet merkittävän suosion kehittäjien keskuudessa.



Kuva 2.3: IoT-tutkimusten määrän kehittyminen vuosina 2006-2018 Dachyar et al. [24] mukaan, muokattu

Eräs merkittävä ero erityyppisten kehitysalustojen välillä on niiden teknisissä resursseissa. Tehokkaimmat alustat ovat kyvykkäitä ajamaan esimerkiksi Linux-ytimeen pohjautuvaa käyttöjärjestelmää ja niihin on mahdollista kytkeä samoja lisälaitteita kuin perinteisiin tietokoneisiin, jolloin ne muistuttavat käytännössä enemmän perinteistä PC-tietokonetta kuin sulautettua järjestelmää. Osa alustoista on huomattavasti rajoittuneempia, jolloin niissä ei välttämättä ole edes käyttöjärjestelmää, vaan ne suorittavat pelkästään laitteeseen kirjoitettua ohjelmistokoodia. Taivalsaari ja Mikkonen [93] toteavat artikkelissaan, että ohjelmistokehitys IoT-laitteille ei eroa merkittävästi perinteisestä ohjelmistokehityksestä. Suurin ero syntyy siitä, että ohjelmisto toteutetaan spesifisti tietylle laitteistolle tai laitteistoille, jotka on kytketty toisiinsa. Toisaalta he myös huomauttavat, että reaalimaailman IoT-järjestelmiin saattaa kuulua erittäin suuria määriä laitteita, joka tuo kompleksisuutta myös ohjelmistokehitykseen.

Arduino Uno, Raspberry Pi sekä ESP32 -kehitysalustoja yhdistää se, että kaikista näistä laitteista valmistaja on julkaissut hyvin paljon informaatiota vapaaseen käyttöön. Arduino noudattaa Open Source Hardware (OSHW), eli avoimen laitteis-

ton ajattelumallia [12]. Tämä on avoimin mahdollinen ajattelumalli ja tarkoittaa sitä, että kaikki mahdollinen tieto laitteen valmistamista varten on vapaasti kaikkien saatavissa. Näiden tietojen vapaa saatavuus mahdollistaa eri toimijoille laitteiden valmistamisen sekä myynnin, jolloin toistensa kanssa yhteensopivia, ominaisuuksiltaan identtisiä laitteita on laajasti myynnissä edulliseen hintaan. Ehtona kuitenkin on, että näissä kloonilaitteissa ei saa käyttää Arduino-nimeä tai logoa [11]. Vastaavasti myös ESP32-piirin kehittäjä Espressif tarjoaa dokumentaatiot muun muassa ESP32-kehitysalustojen valmistamista varten [35]. Raspberry Pi:n toimintamalli ei ole yhtä avoin, joten laitteesta ei ole mahdollista tehdä vastaavia klooneja kuin Arduino-laitteista. Raspberry Pi -laitteista on kuitenkin julkisesti saatavissa kytkentäkaaviot ja mekaaniset piirustukset, jotka mahdollistavat kolmansien osapuolien valmistamat lisälaitteet [85].

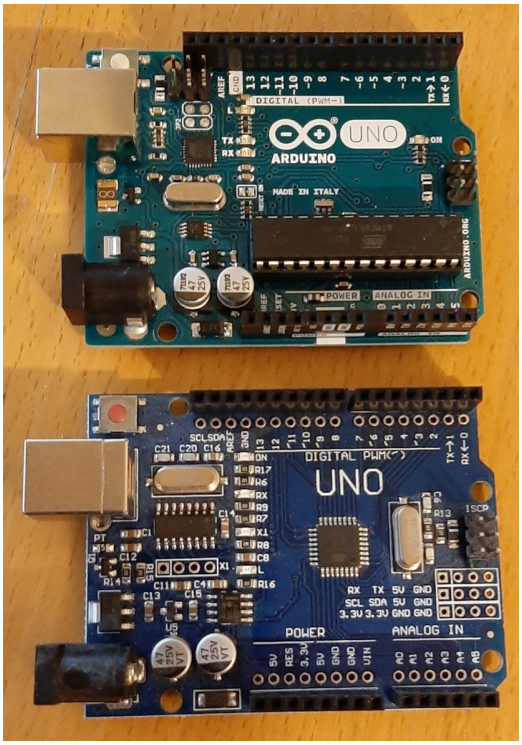
Arduino Uno

Arduino Uno on yksi suosituimmista Arduino AG:n kehittämistä kehitysalustoista. Vuonna 2010 julkaistu Uno korvasi muutamaa vuotta aiemmin julkaistun Duemilanove-mallin [9]. Arduino Unon tärkeimmät komponentit ovat 16 megahertsin kellotaajuudella toimiva ATMega328P-prosessori, 32 kilotavua flash-muistia sovellusta varten, 2 kilotavua keskusmuistia sekä 1 kilotavun Electronically Erasable Programmable Read-Only Memory (EEPROM) [10]. Näiden teknisten ominaisuuksien perusteella laitteen suorituskykyä voidaan kuvata verrattain heikoiksi.

Vaatimattomasta suorituskyvystä huolimatta Arduino Uno:ssa on 6 kappaletta analogisia sisääntuloja sekä 14 kappaletta digitaalisia General Purpose Input/Output (GPIO) -liitäntöjä. Näihin liitäntöihin voidaan kytkeä useita lisälaitteita, kuten erityypisiä sensoreita ja aktuaattoreita. Lisälaitteita on saatavissa hyvin laajasti ja niiden käyttämistä helpottaa laitevalmistajien ja Arduino-yhteistön tuottamat ohjelmakirjastot. Arduino Uno pystyy kommunikoimaan muiden laitteiden kanssa Universal Asynchronous Receiver Transmitter (UART) -piirin avulla, mutta esimerkiksi sisäänrakennettua WLAN tai Bluetooth-ominaisuutta ei ole. Arduino Unon liittäminen erityyppisiin verkkoihin ei kuitenkaan ole mahdotonta, koska useat valmistajat tarjoavat lisäosia, joilla kehitysalusta voidaan liittää verkkoon halutulla tavalla. Yhtenä esimerkkinä voidaan mainita Dragino Technologyn valmistama moduuli, joka mahdollistaa kommunikoinnin LoRa-verkossa [29].

Kuvassa 2.4 on ylempänä virallinen, Arduino AG:n valmistama Arduino Uno, alempana kolmannen osapuolen valmistama klooni. Laitteet ovat kooltaan, liitän-

nöiltään ja muilta teknisiltä ominaisuuksiltaan lähes identtiset. Arduino AG:n valmistamassa mallissa ATMega328P-mikrokontrolleri on kiinnitetty jousikantaan, kun kloonissa se on juotettu suoraan piirilevyyn. Aiemmin mainittujen Arduino AG:n asettamien ehtojen täyttämiseksi kloonilaitteessa ei lue sanaa *"Arduino"* eikä siinä käytetä Arduinon logoa.



Kuva 2.4: Arduino Uno -kehitysalustat

Arduino Uno:a on käytetty alustana monissa pienissä järjestelmissä tai yhtenä osana isommissa kokonaisuuksissa, joiden tarkoituksena on ratkaista arkipäiväisiä ongelmia. Tyypillisinä esimerkkitoteutuksena voidaan mainita Kumarin et al. [65] toteuttama laitteisto, jonka tehtävänä on seurata maaperän kosteuspitoisuutta. Arduino Uno:n lisäksi oleellisimmat komponentit ovat kosteussensori sekä ledlamput, joiden avulla ilmaistaan maaperän kosteuspitoisuus. Toisena esimerkkinä Arduino Uno:n hyödyntämisestä on Mittalin et al. [73] toteuttama kokonaisuus. Tässä kokonaisuudessa Arduino Uno on osana prototyyppikonseptia, joka mahdollistaa pääsynhallintaa toteutetaan biometristen tunnisteiden avulla.

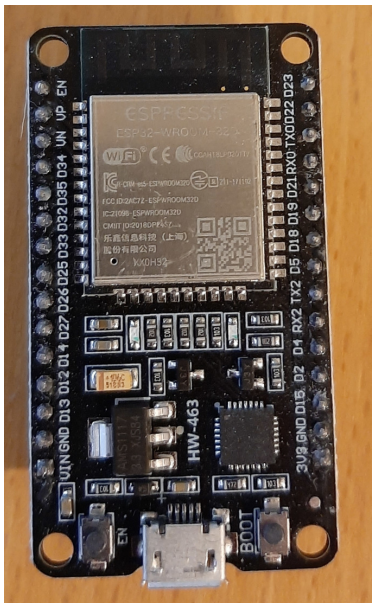
ESP32

ESP32 on Espressif Systemsin kehittämä System on a chip (SOC), eli järjestelmäpiiri,

joka sisältää useita toimintoja. Espressif on kehittänyt ESP32-alustoja hyvin aktiivisesti. Ensimmäinen versio ESP32-piiristä on julkaistu vuonna 2016 [34], mutta tämän tutkielman kirjoitushetkellä Espressif on julkaissut jo 28 erilaista ESP32-piiriin perustuvaa kehitysalustaa [32]. Nämä kehitysalustat eroavat toisistaan esimerkiksi muistimäärien ja liitäntöjen osalta. Laajan kehitysalustavalikoiman ja kattavien ohjelmakirjastojen ansiosta ESP32 on käyttäjäystävällinen vaihtoehto sulautettujen järjestelmien kehitystyöhön.

Tämän tutkielman yhteydessä rakennetun mittauslaiteen prototyyppi pohjautuu NodeMCU-ESP32 Devkit V1 -kehitysalustaan. Tämän kehitysalustaversion järjestelmäpiirinä on ESP32-WROOM-32D jonka tärkeimmät tekniset ominaisuudet ovat 240 megahertsin taajuudella toimiva kahden ytimen prosessori, 4 megatavun flash-muisti sekä 520 kilotavun ram-muisti [33], jolloin tekninen suorituskyky on kaikilla osa-alueilla merkittävästi parempi kuin Arduino Uno:ssa.

Kuvassa 2.5 olevassa NodeMCU-ESP32 Devkit V1 -kehitysalustassa on 25 kappaletta GPIO-liitäntöjä lisälaitteita varten. Vastaavasti kuin Arduino Uno:ssa, näitä liitäntöjä voidaan käyttää lisälaitteiden liittämiseen. ESP32-piireihin on sisäänrakennettuna Wi-Fi 802.11b/g/n ja Bluetooth v4.2, jolloin tietoliikennemahdollisuudet ovat lähtökohtaisesti paremmat kuin Arduino Uno:ssa.



Kuva 2.5: ESP32-piiriin perustuva kehitysalusta, NodeMCU-ESP32 Devkit V1

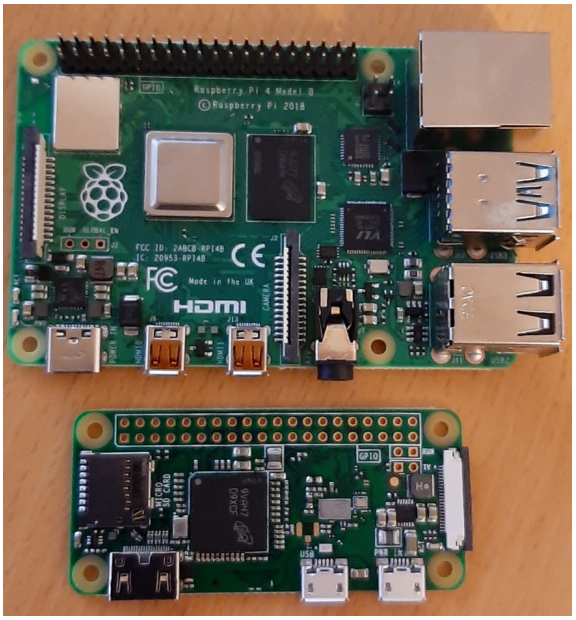
Vastaavasti kuin Arduino Uno:a, myös ESP32-kehitysalustaa on käytetty alustana monissa pienissä projekteja tai osana isompaa kokonaisuutta. Rainin ja Rehmanin [82] toteuttama konsepti valvontakamerajärjestelmästä on esimerkki kokonaisuudesta, jossa ESP32 toimii alustana sekä palvelimen ja asiakaan roolissa. Palvelimen roolissa olevaan ESP32-alustaan on kytketty kameramoduuli, joka lähettää kuvamateriaalin WLAN-verkon kautta asiakkaana toimivaan toiseen ESP32-alustaan, johon kytketystä LCD-näytöstä nähdään kameran tuottama kuvamateriaali. Huomionarvoisena seikkana voidaan pitää myös palvelimen roolissa toimivaa Arducam ESP32 Uno -alustaa. Teknisiltä ominaisuuksiltaan se pohjautuu ESP32-kehitysalustaan, mutta liittimien lukumäärän ja fyysisen sijoittelun osalta se on täysin yhteensopivia Arduino Uno:n pinnien kanssa [8]. Tämä yhdistelmä tarjoaa ESP32:n suorituskyvyn sekä mahdollisuuden käyttää Arduino Uno:lle suunniteltuja lisälaitteita.

Raspberry Pi

Raspberry Pi on yleisnimitys Raspberry Pi -säätiön kehittämille Single-board computer (SBC), eli yhden kortin tietokonelaitteille. Tuoteperheen ensimmäinen laiteversio julkaistiin vuonna 2012. Raspberry Pi -tuoteperheeseen kuuluvat laitteet pohjautuvat Broadcomin valmistamiin järjestelmäpiireihin, jotka on kehitetty nimenomaan kyseisiä laitteita varten, eikä niitä ole tarjolla muille valmistajille [84]. Kyseiset järjestelmäpiirit sisältävät ARM-arkkitehtuuriin perustuvan prosessorin ja Broadcomin VideoCore-grafiikkasuorittimen. Mallista riippuen Raspberry Pi voi sisältää yhden tai useamman CPU-ytimen ja enimmillään useita gigatavuja RAM-muistia. Laitteissa itsessään ei ole tallennuskapasiteettia, mutta se voidaan lisätä SDHC tai MicroSDHC -muistikortin tai USB-väylään liitettävän muistin avulla.

Raspberry Pi -laitteissa on kattavat liitännät ulkoisia laitteita ajatellen. GPIO-liitäntöjä voi olla enimmillään 40 kappaletta, mallista riippuen laitteissa voi olla myös Ethernet, WLAN ja Bluetooth -kyvykkyydet. Rasperry Pi -laitteisiin voidaan myös kytkeä ulkoinen näyttö, HDMI-liitäntää käyttäen.

Poikkeuksellisena laitteena tuoteperheeseen kuuluu Raspberry Pi Pico, joka samankaltaisesta nimestään on täysin erityyppinen laite muihin Pi-sarjan laitteisiin verrattuna. Raspberry Pi Pico on ominaisuuksiltaan verrattavissa Arduino Uno ja ESP32-mikrokontrollereihin. Tämä laite sisältää 133 megahertsin kellotaajuudella toimivan RP2040-prosessorin, 2 megatavua flash-muistia sekä 264 kilotavua keskusmuistia. Lisälaitteiden liittämistä varten Pico:ssa on 26 kappaletta GPIO-liitäntöjä, joista kolmea voidaan käyttää analogisina sisääntuloina [83].



Kuva 2.6: Ylempänä Raspberry Pi 4. Alempana Raspberry Pi Zero W.

Raspberry Pi:n merkittävästi parempi suorituskyky Arduino Uno ja ESP32 -alustoihin verrattuna ja tavallista PC-laitetta muistuttavat ominaisuudet mahdollistavat toimintoja, jotka ovat haastavia tai mahdottomia toteuttaa mikrokontrollereihin perustuvissa kehitysalustoissa. Raspberry Pi:n vahvuudet liittyvät IoT-käyttötapauksissa esimerkiksi tilanteisiin, joissa vähäistä suuremman datamäärän tallentaminen on tarpeen, käyttötarve vaatii USB-väylään kytkettäviä laitteita tai alustaan halutaan saada hallinta- tai tiedonsiirtoyhteys esimerkiksi Secure Shell (SSH) -protokollaa käyttäen.

Yksi esimerkki Raspberry Pi:n ominaisuuksia hyödyntävästä käyttökohteesta on BinMasoudin ja Chengin [13] toteuttama ajoneuvon monitorointijärjestelmä, jossa Raspberry Pi kytketään ajoneuvon On-board diagnostics (OBD) -väylään. Tällä menetelmällä voidaan kerätä perustietoja ajoneuvon tilasta, kuten nopeudesta, moottorin kierrosluvusta ja lämpötilasta sekä mahdollisista vikakoodeista. BinMasoudin ja Chengin esimerkissä mittaustiedot tallennetaan väliaikaisesti Raspberry Pi:n paikalliseen muistiin ja ne lähetetään palvelimelle, kun Raspberry Pi liitetään WLAN-verkkoon. Mittaustietoja yhdistämällä voidaan havaita vikatilanteiden olosuhteisin liittyviä riippuvuuksia.

Erityyppisenä käyttötapauksena voidaan mainita Jusakin et al. [59] kehittämä lääkinnällisen laitteen konsepti, jossa potilaan tilaa seurataan useiden sensoreiden

avulla. Sensorit on kytketty Arduino-alustaan, joka kommunikoi Bluetoothin avulla Raspberry Pi -tietokoneeseen. Raspberry Pi:n tärkein tehtävä on välittää tiedot eteenpäin varsinaisen kohdejärjestelmän palvelimelle 5G-yhteyden kautta. Tässä konseptissa Raspberry Pi toimii yhdyskäytävän roolissa, vastaanottaen dataa useammasta Arduino-alustasta. Teknisesti järjestelmä voitaisiin toteuttaa pelkästään Raspberry Pi -laitteillakin, mutta tarkoituksenmukaisesti eri alustojen kyvykkyyksiä yhdistämällä saavutetaan kustannussäästöjä, koska Arduino-alustat ovat selvästi edullisempia kuin Raspberry Pi.

Yhteenveto kehitysalustoista

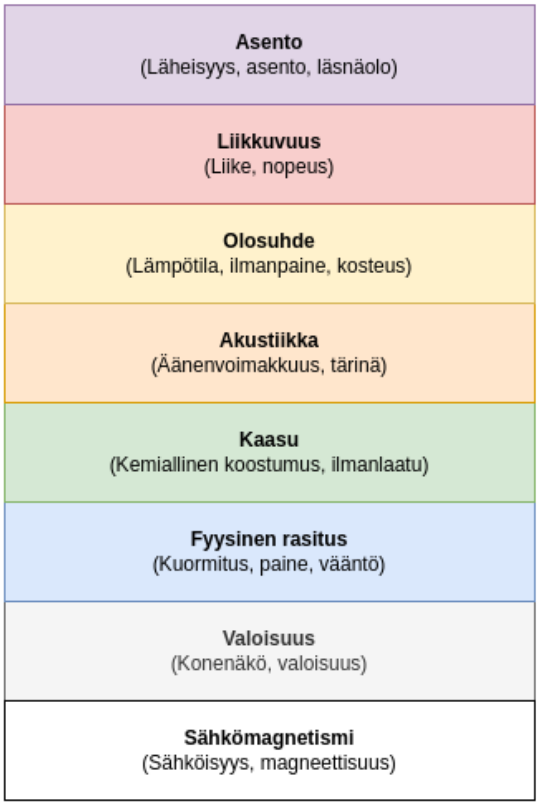
Kehitysalustojen välillä on merkittäviä eroavaisuuksia. Sopivaa kehitysalustaa valittaessa on syytä miettiä, minkälaisia tarpeita ja vaatimuksia on täytettävänä. Valintaan vaikuttaa olennaisesti se, kuinka suurta teknistä suorituskykyä laitteelta tarvitaan ja minkälaisia lisälaitteita on kyettävä käyttämään. Jos suorituskykyä tarvitaan runsaasti ja laitteeseen halutaan kytkeä esimerkiksi USB-laitteita, on edellä mainituista laitteista Raspberry Pi -tuoteperheeseen kuuluva laite oikea vaihtoehto. Jos puolestaan suorituskykytarve on vähäinen ja laitteeseen halutaan kytkeä lisälaitteita vain GPIO-liitäntään, sopivin laite on todennäköisesti Arduino Uno tai ESP32. Tällöin valinta voidaan tehdä muiden ominaisuuksien perusteella. Merkittävin ero näiden välillä on verkkokyvykkyyksien välillä. Arduino Uno:ssa ei ole sisäänrakennettua tukea millekään verkkoteknologialle, kun ESP32 tarjoaa mahdollisuuden WLAN-verkkojen ja Bluetoothin hyödyntämiseen.

Merkittävä suunnannäyttäjä kehitysalustaa valittaessa on myös ohjelmointikieli, jota halutaan käyttää. Raspberry Pi mahdollistaa käytännössä minkä tahansa ohjelmointikielen käyttämisen, kuten myös useat Linux-jakeluversiot ja Berkeley Software Distribution (BSD) -pohjaiset käyttöjärjestelmät. Arduino Uno ja ESP32 eivät mahdollista käyttöjärjestelmien asentamista. Arduino Unon ja ESP32 välillä merkittävä ero on se, että ESP32 tukee MicroPython-ohjelmointikieltä, joka on puolestaan laajasti yhteensopiva Python 3 -ohjelmointikielen kanssa.

2.2.3 Sensorit ja aktuaattorit

Tässä alaluvussa esitellään muutamia erityyppisiä, yleisesti käytettyjä sensoreita. Kaikki esiteltävät sensorit ovat yhteensopivia edellisessä alaluvussa esiteltyjen kehitysalustojen kanssa, joko sensorivalmistajan tai yhteisön kehittämien laiteohjelmistojen ja kirjastojen avulla.

Useiden IoT-laitteiden toimintamalli perustuu erilaisiin havaintoihin ja mittauksiin. Nämä toiminnot mahdollistetaan erityyppisten sensoreiden avulla. Sehrawat ja Gill [89] käsittelevät artikkelissaan IoT:ta on mullistavana ilmiönä, joka integroi erityyppisiä sensoreita toisiinsa, jolloin yksittäisillä sensoreilla tehdyt havainnot ja mittaukset tulevat osaksi isompia kokonaisuuksia. Esimerkkinä mainitaan älyrakennusjärjestelmät ja terveysjärjestelmät, joissa mitataan useita ominaisuuksia erityyppisten sensoreiden avulla kokonaiskuvan muodostamiseksi. Kuvassa 2.7 sensorit on jaettu Sehrawatin ja Gillin [89] artikkelin mukaisesti kahdeksaan alajoukkoon. Näihin alalajeihin kuuluu lukuisia erityyppisiä sensoreita, joita valmistetaan useiden laitevalmistajien toimesta. Käyttökohde vaikuttaa tarkan sensorimallin valintaan, koska sensorimallien välillä on eroavaisuuksia esimerkiksi mittaustarkkuudessa ja mittausvälin suuruudessa.



Kuva 2.7: Sensorityypit Sehrawatin ja Gillin [89] mallin mukaisesti, muokattu

Sensoreiden ja aktuaattoreiden toimintaperiaattesta

Collinin ja Saarelaisen [20] mukaan sensorit ovat laitteita, jotka muuntavat informaatiota sähköiseen muotoon tietoa sellaisista ilmiöistä, jotka eivät luonnostaan ole sähköisessä muodossa. Esimerkkejä tämäntyyppisistä ilmiöistä he mainitsevat lämpötilan, sijainnin, sameuden ja nestepinnan korkeuden. Sensoreiden toiminta voi olla luonteeltaan passiivista tai aktiivista. Passiivinen sensorin toiminta perustuu datan vastaanottamiseen, aktiivinen sensori lisäksi lähettää dataa.

Aktuaattorit ovat laitteita, joiden tehtävänä on tuottaa liikettä. Yksi esimerkki aktuaattorista on solenoidi. Solenoidin tärkeimmät osat ovat sähkömagneetti ja sähkömagneetin sisällä oleva magneettinen, liikkuva osa, jota kutsutaan ankkuriksi. Kun solenoidiin syötetään virtaa, sähkömagneetti magnetisoituu ja tämän myötä ankkuri liikkuu. IoT-laitteiden yhteydessä solenoideja voidaan hyödyntää esimerkiksi kohteissa, joissa oven lukitus halutaan etäavata.

Bosch Sensortec BME280

BME280 on sensori, jonka on kehittänyt Bosch Sensortec [18]. BME280 mahdollistaa lämpötilan, ilmankosteuden ja ilmanpaineen mittaamisen. Anturi tukee tiedonsiirrossa sekä Inter-Integrated Circuit (I2C) ja Serial Peripheral Interface (SPI) -väyliä. Valmistajan mukaan sensori soveltuu käytettäväksi erityisesti liikkuvissa kohteissa sekä puettavissa IoT-laitteissa alhaisen virrankulutuksen ansiosta. BME280 sijaitsee ylimpänä kuvassa 2.8.

Winsen Electronics MQ-2

MQ-2 on sensori, jonka on kehittänyt Winsen Electronics [117]. MQ-2 pystyy tunnistamaan savun lisäksi erilaisia palavia kaasuja, joten sen pääasiallinen käyttötarkoitus on Winsen Elenctonicsin mukaan erilaisissa kaasuvuotoihin liittyvissä kohteissa. Sensori tarjoaa mittaustulokset digitaalisessa ja analogisessa muodossa, eli sensorissa on kaksi ulostuloa. Digitaalisesta ulostulosta saadaan tieto, jos määritetty raja-arvo on ylittynyt. Analogisesta ulostulosta saadaan mittauksen tarkka arvo. Kuvassa 2.8 MQ-2 on komponentti, jonka piirilevy on sininen.

Keyestudio KS0052

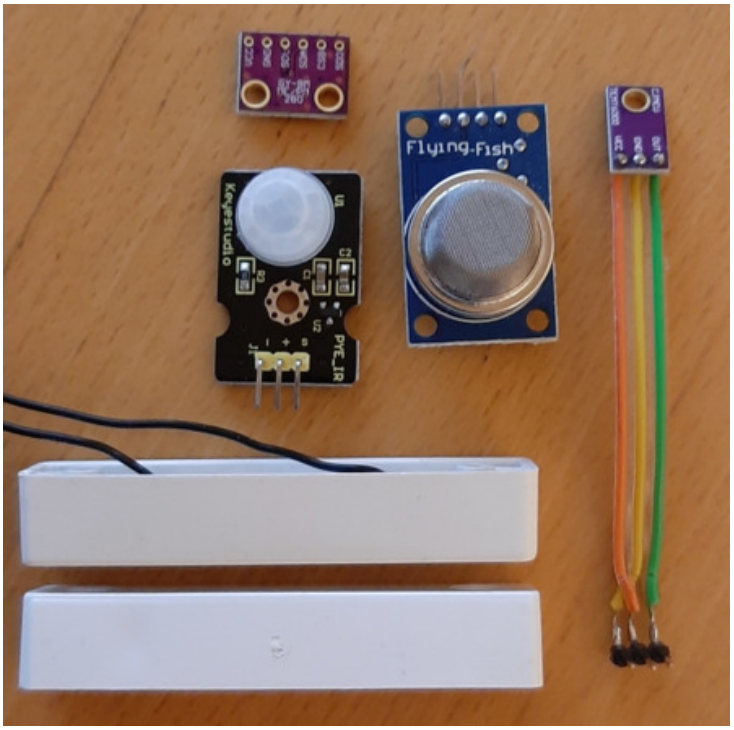
KS0052 on sensori, jonka on kehittänyt Keyestudio [90]. KS0052 on Passive Infrared (PIR)-sensori, eli passiivinen infrapuna-anturi. KS0052 mahdollistaa liikkeen havaitsemien luotettavasti korkeintaan 3-4 metrin etäisyydeltä. Sensorissa on yksi digitaalinen ulostulo, jonka kautta saadaan tieto havaitusta liikkeestä. Valmistajan antamien tietojen mukaan sensori reagoi havaittuun liikkeeseen alle kolmessa sekunnissa. Kuvassa 2.8 KS0052 on sensori, jonka piirilevy on musta.

Vishay TEMT6000

TEMT6000 on sensori, jonka on kehittänyt Vishay Semiconductors [103]. TEMT6000 mittaa valoisuutta. Sensorissa on yksi analoginen ulostulo, jonka kautta saadaan mitatun valoisuuden tarkka arvo. Valmistajan mukaan sensoria voidaan käyttää esimerkiksi älypuhelimien näytön taustavalon automaattiseen himmentämiseen.
TEMT6000 sijaitsee kuvan 2.8 oikeassa reunassa.

Magneettikytkin

Magneettikytkimen toiminta koostuu kahdesta osasta. Ensimmäinen osa sisältää releen, joka yhdistää kytkimen sisällä olevan johtimen tai katkaisee sen. Releen materiaali on magneettista. Magneettikytkimen toinen osa sisältää magneetin. Kun nämä kaksi osaa tuodaan lähelle toisiaan, kytkimen asento muuttuu. Kytkimen asento muuttuu myös silloin, kun nämä osat erotetaan toisistaan. Magneettikytkimiä on kahta eri tyyppiä, normally closed (NC), eli normaalisti suljettu ja normally open (NO), eli normaalisti auki. Nimensä mukaisesti näiden ero on siinä, onko johdin yhdistetty vai katkaistu silloin, kun magneetti on tuotu releen lähelle. Magneettikytkimen avulla voidaan saada tieto esimerkiksi siitä, onko ovi avoinna vai suljettu. Kuvassa 2.8 olevat valkoiset osat muodostavat magneettikytkimen.



Kuva 2.8: Sensorit BME280, KS0052, MQ-2, TEMT6000 ja magneettikytkin

Collinin ja Saarelaisen [20] mukaan ratkaisevia tekijöitä IoT-laitteiden voimakkaaseen yleistymiseen ovat sensorien hintojen laskeminen, sekä niiden koon ja virrankulutuksen pienentyminen. Heidän mukaansa tähän kehitykseen on puolestaan vaikuttanut älypuhelinmarkkinoiden voimakas kilpailu. Älypuhelimiin on kehitetty runsaasti toimintoja, jotka vaativat toimiakseen useita sensoreita. Hintakilpailun lisäksi älypuhelinvalmistajat ovat joutuneet tekemään paljon tuotekehitystä tuotteiden houkuttelevuuden parantamiseksi kaikilla osa-alueilla, jonka takia sensoreidenkin kokoa ja virrankulutusta on täytynyt saada pienemmäksi. Toisena esimerkkinä he mainitsevat autoteollisuuden, jossa sensoreiden tarve on lisääntynyt autoihin kehitettyihin ominaisuuksien myötä. He pitävät varmana, että autoteollisuudessa sensoreiden tarve tulee lisääntymään entisestään itseajavien autojen myötä.

2.3 IoT-laitteiden potentiaali ja käyttökohteet

Jo suuresta laitemäärästä sekä Poornachandranin [81] ja Sivanathanin et al. [91] mainitsemista esimerkkilaitteista voidaan tehdä päätelmä, jonka mukaan IoT-laitteilla on käyttökohteita useilla eri teollisuudenaloilla. Yleisellä tasolla voidaankin sanoa,

että potentiaalisia käyttökohteita on kaikkialla, missä erityyppisten kohteiden säännöllinen mittaus on tarpeen, sekä tarvetta saada tämä mittausdata reaaliaikaisesti tai lähes reaaliaikaisesti eteenpäin sen jatkojalostamista varten. Useissa tapauksissa IoT:n avulla on mahdollista saavuttaa konkreettisia hyötyjä vanhoihin toimintamalleihin verrattuna sekä lisäksi saada rahallisia säästöjä.

2.3.1 Kiinteistöt ja kaupunkiympäristö

Esimerkkitapauksena kiinteistöjen liittämisestä IoT-ympäristöön voidaan mainita Digita Oy:n ja Kojamo Oyj:n (Ennen VVO-yhtymä Oyj) pilottihanke [26], jonka tarkoituksena oli kerätä tietoja asuntojen lämpötilasta ja kosteudesta. Mittaukset tehtiin antureilla, joilla oli kyvykkyys välittää mittausdata eteenpäin LoRa-tekniikkaa hyödyntäen. Ensisijainen tavoite oli säätää kiinteistöjen järjestelmiä siten, että asumisolosuhteet ovat asukkaiden kannalta ihanteelliset. Mittauksen tarkkuudesta riippuen, pidemmällä ajanjaksolla voidaan todennäköisesti säästää kiinteistöjen saneerauskustannuksissa, mikäli esimerkiksi kosteuden aiheuttamat ongelmat voidaan havaita varhaisessa vaiheessa. Digitan ja Kojamon pilottihanke onnistui [28], jonka myötä Kojamosta tuli Digitan valtakunnallisen IoT-verkon ensimmäinen asiakas.

IoT:n tuomia mahdollisuuksia voidaan hyödyntää myös isommassa mittakaavassa. Esimerkiksi käsite älykaupunki (Smart City) pitää sisällään useita kaupunkiympäristöön liittyviä osa-alueita, kuten liikenteenvalvonnan, ilmansaasteen ja vedenlaadun mittauksen sekä vapaiden parkkipaikkojen reaaliaikaisen tilannetiedon selvittämisen. Kim et al. [61] tiivistävätkin asian siten, että älykaupunkien IoT:ssa on kyse siitä, että sensoreita asennetaan kaikkialle, jolloin mahdollistetaan havainnointikykyä eri asioihin, paikannusta ja seurantaa.

2.3.2 Terveydenhuolto

Myös terveydenhuollossa voidaan havaita useita käyttökohteita IoT-laitteille. Rodriguesin et al. [87] mukaan IoT on yksi lähitulevaisuuden lupaavimmista teknologioista, joka tulee vaikuttamaan terveydenhuoltoon ja hyvinvointiin. He esittelevät termin Internet of Health Things (IoHT), vapaasti suomennettuna terveysteknologian esineiden Internet. Tämän termin alle sisältyy kokonainen järjestelmäarkkitehtuuri, joka sisältää koko ketjun alkaen tiedonkeruusta, päättyen potilastietojärjestelmään, jossa tiedot on esitetty lääketieteen ammattilaisen näkökulmasta helposti käsiteltävässä muodossa. Teknologisesta näkökulmasta arkkitehtuuri on usein

monimutkainen: Ennen kuin potilaan tiedot päätyvät potilastietojärjestelmään, data kulkee useiden laitteiden kautta, eri tiedonsiirtomenetelmiä hyödyntäen. Mainittuja IoHT:n kannalta oleellisia tiedonsiirtotekniikoita ovat muun muassa BLE, ZigBee ja WLAN. Tärkeimmät laitteet ovat tiedonkeruuseen käytetyt sensorit, päätelaitteet, joihin sensorit kytketään tiedonlähetystä varten, reitittimet, pilvipalvelut, tietokannat ja potilastietojärjestelmien käyttämät palvelimet.

Oikein toteutettuna IoT:n avulla saadaan kerättyä potilaasta arvokasta dataa, jota ei saataisi talteen perinteisellä lääkärin vastaanotolla. Esimerkkejä näistä ovat erilaiset pitkäaikaiseen datankeruuseen perustuvat seurannat sekä harvoin toistuvat, mitattavissa olevat oireet. Joissain tilanteissa on myös mahdollista vähentää potilaan läsnäoloa sairaalassa. Tämä on luonnollisesti potilaan kannalta mielekästä, mutta isommassa mittakaavassa tällä olisi mittavat rahalliset säästöt. Erityisesti erikoissairaanhoito on yhteiskunnalle kallista. Kuntaliiton antamien tietojen mukaan vuonna 2020 erikoissairaanhoito maksoi Suomessa vuositasolla 1309 euroa asukasta kohden [92].

2.4 IoT-laitteiden ongelmat, uhkakuvat ja näihin varautuminen

IoT:n todetuista hyödyistä huolimatta kokonaiskuvaa tarkasteltaessa kuitenkin myös ongelmia ja uhkakuvia on havaittavissa. Tietoturvaan liittyviin asioihin on nykyisin kiinnitettävä erityistä huomiota, koska tiedot haavoittuvuuksista ja niiden käyttömahdollisuuksista leviävät nopeasti. Yleisellä tasolla voidaan sanoa, että IoT-laitteiden tietoturvassa on puutteita ja tämä aiheuttaa useita erilaisia haasteita. On kuitenkin huomattava, että termi IoT pitää sisällään hyvin kirjavan joukon erityyppisiä laitteita, jolloin mikään mahdollisista riskeistä ei ole sellaisia, jotka koskisivat koko joukkoa.

Kolias et al. [63] pitävät IoT-laitteita erityisesti Distributed Denial of Service (DDoS) -hyökkäysten mahdollistavina alustoina. Koska laitteet ovat jatkuvasti liitettynä Internetiin, niitä on suuri määrä, haavoittuvuuksia on paljon ja tietoturvaan liittyvät määritykset ovat puutteelliset, niiden haltuunottaminen ja muuntaminen hyökkääjän omiin tarkoituksiin on helppoa. Esimerkkinä he mainisevat Miraibottiverkon, johon kuului vuonna 2016 jopa 400 000 laitetta [63]. Vaikka yksittäiset IoT-laitteiden suorituskyky on heikko verrattuna tavanomaisiin toimisto- tai kotikäytössä oleviin työasemiin, ne pystyivät joukkona tuottamaan suuren määrän verkkoliikennettä, joka pystyi aiheuttamaan useita tunteja kestäviä palvelukatkoja

tunnettuihin palveluihin, kuten Twitter-yhteisöpalveluun, Netflix-suoratoistopalveluun ja GitHub-versionhallintapalveluun.

Haavoittuvuudet ja puutteelliset tietoturvakonfiguraatiot altistavat myös väärinkäytöksille ja tietomurroille. Vojkovic et al. [105] esittävät artikkelissaan, että haavoittuvuudet kotien älyjärjestelmissä (Smart home) voivat mahdollistaa murtautumisen rakennuksiin, mikäli älylukko on avattavissa haavoittuvan järjestelmän kautta. He tuovat myös esiin huolensa valvontakameroihin liittyvistä ongelmista, joiden puutteelliset konfiguraatiot mahdollistavat luvattoman katselun. He tuovat myös esiin Yhdystaltain elintarvike- ja lääkeviraston vahvistaman haavoittuvuudet vuodelta 2017, joka mahdollisti eräiden sydäniskurien ja sydämentahdistimien väärinkäytön ja antoi hyökkääjälle laitteiden täyden hallinnan. Tämäntyyppiset haavoittuvuudet voivat pahimmillaan aiheuttaa potilaan kuoleman.

IoT-laitteiden keräämään dataan voidaan liittää myös yksityisyyteen liittyviä haasteita. Yksilöiden kannalta on ongelmallista, jos kerätyn datan käyttötapaa ja tarkoitusta ei ole raportoitu. Kun erityyppisiä datalähteitä on suuri määrä, kerättyjä tietoja yhdistelemällä on mahdollista koostaa yksilöstä hyvin tarkkoja tietopaketteja. Yksilöllä on mahdollisuus valita itse, minkälaisia laitteita käyttää henkilökohtaisesti, jolloin esimerkiksi kotien älyjärjestelmistä voi kieltäytyä kokonaan. Myöskään IoHT-laitteiden hyödyntäminen ei ole välttämätöntä. Sen sijaan esimerkiksi julkisilla paikoilla liikkuessaan yksilö ei voi olla varma, minkä tyyppisiin tarkoituksiin tietoja kerätään. Valvontakameroiden avulla voidaan tehdä esimerkiksi kasvojentunnistamista ja autojen rekisterinumeroiden seurantaa. Anastasi et al. [7] esittelivät jo vuonna 2003 konseptin, jonka avulla Bluetooh-teknologiaa voidaan käyttää paikantamiseen. Yhdistämällä edellä mainittuja menetelmiä voidaan tehdä valvontaa tarkalla tasolla. Tämän vuoksi lainsäädännöllä on suuri merkitys tietojenkeräämisen ja hyödyntämisen näkökulmasta. Esimerkiksi Yhdysvalloissa, San Franciscon kaupungissa säädettiin vuonna 2019 laki [21], jonka perusteella poliisilla ei ole oikeutta hyödyntää kasvojentunnistusta.

IoT-laitteiden ongelmakohtiin liittyen on oleellista havaita, että osa ongelmista on puhtaasti tahattomia ja usein ratkaistavissa teknisin menetelmin. Osa ongelmista puolestaan liittyy kerätyn datan epäsopivaan hyödyntämiseen ja väärinkäytökseen, jolloin tilannetta ei voida ratkaista teknisin menetelmin, vaan avuksi tarvitaan lainsäädäntöä ja muita säädöksiä, jotka ohjaavat IoT-laitteiden kehitystä.

3 Konesaliympäristö

Konesalit ovat tiloja, joissa sijaitsee tietojenkäsittelyyn tarvittavia palvelimia, levyjärjestelmiä, tietoliikennelaitteita sekä näiden toimintaa tukevia laitteita, kuten Uninterruptible Power Supply (UPS) -laitteistoja. Tyypillisesti näiden palvelimien varassa on organisaatioiden kannalta tärkeitä toimintoja ja palveluita, joiden tulee olla käytettävissä ympärivuorokautisesti. Korkean kriittisyysluokan palveluissa halutaan välttää lyhyitäkin ennakoimattomia käyttökatkoja.

Konesaliympäristöissä voidaan tunnistaa useita tekijöitä ja uhkakuvia, jotka vaikuttavat toteutuessaan negatiivisesti palveluiden toimivuuteen. Tässä tutkielmassa keskitytään lämpötilan nousun aiheuttamiin ongelmiin, fyysisten hyökkäysten havaitsemiseen sekä vikaantuneen laitteiston aiheuttamiin sähköpaloihin.

3.1 Laitteisto ja virtualisointi

Konesaliympäristöissä käytettävät laitteistot ovat muuttuneet vuosikymmenten aikana. Aiemmin kaikki konesaleissa sijaitsevat laitteistot ovat olleet fyysisiä, mutta jo vuosien ajan palvelimien virtualisointi on ollut yleistä. Muidenkin laitteiden, kuten reitittimien, palomuurien ja kuormantasaajien virtualisointi on nykyisin mahdollista myös tuotantoympäristöissä. Virtualisointi parantaa merkittävästi energiatehokkuutta ja tuo kustannussäästöjä, koska vähemmällä palvelinlaitteistomäärällä voidaan ajaa useampaa virtualisoitua palvelinta tai palvelua. Motochin et al. [74] mukaan virtualisointia hyödyntäen voidaan säästää konesaliympäristön energiankulutuksessa 20 %. Kasvanut virtualisointiaste tuo hyvien puolien mukana myös haasteita. Kun laitteiston varassa toimii useampia virtualisoituja palveluja, nämä palvelut ovat luonnollisesti riippuvaisia tämän laitteiston toimivuudesta. Vikasietoisia ratkaisuja voidaankin toteuttaa erilaisin menetelmin, kuten kahdentamalla kaikki palvelun kannalta oleelliset komponentit. Tämä mahdollistaa sen, että palvelu toimii tietojärjestelmän käyttäjän näkökulmasta saumattomasti, vaikka yksittäisessä konesalissa olevat laitteet muuttuisivat äkillisesti käyttökelvottomiksi.

Konesaliympäristöjen energiankulutukseen vaikuttavat useat tekijät. Kataoka et al. [60] osoittavat, että suoritettavien prosessien määrällä on suora yhteys virranku-

lutukseen. Työkuorman painottuessa voimakkaasti Graphics Processing Unit (GPU) -laskentaan, voidaan Tom's Hardwaren [40] tekeminen testien mukaan todeta, että pääsääntöisesti näytönohjaimien suorituskyvyn noustessa myös energiankulutus kasvaa. On myös huomioitava, että vikasietoisuutta ajatellen työkuormia ei voi täysin optimoida. Mikäli tarve on varmistaa virtuaalipalvelinten toimivuus myös laitteiston häiriötilanteessa, on kahdennetun laitteiston oltava identtiset ja työkuorma on jaettava laitteiston välille tasaisesti. Kummankin laitteiston resurssien käyttöaste saa olla teoriassa maksimissaan 50 %, jotta se voi vikatilanteessa ottaa vastaan toisen laitteiston työkuorman. Käytännössä resurssien käyttöaste on oltava vieläkin alhaisempi, koska missään tilanteissa resurssien käyttöaste ei saisi olla 100 %. Jos normaalikuormituksella käyttöaste on 100 %, hetkellisissä kuormituspiikeissä laitteisto ei kykene suoriutumaan ilman palveluihin näkyvää hidastumista. Tämän vuoksi vikasietoisissa virtualisointiympäristöissä virransäästön optimointi jää näennäisesti alhaiseksi. Virtualisoitujen palvelimien ja muiden laitteistojen kustannus- ja energiatehokkuus on tästä huolimatta merkittävää verrattuna siihen, jos kaikki laitteet olisivat fyysisiä. Myös konesalien jäähdytyksen tarve vähenee laitemäärän vähentyessä.

3.2 Konesaliympäristön erityispiirteet

Alaluvussa käsitellään konesaliympäristön olosuhteisiin liittyviä erityispiirteitä, jotka on tärkeää huomioida konesalilaitteiden tarjoamien palveluiden saatavuuden, sekä datan luottamuksellisuuden ja eheyden varmistamiseksi. Käsitellään myös olosuhteiden monitorointiin ja sen toteuttamiseen liittyviä asioita.

3.2.1 Fysikaaliset olosuhteet

Vaikka konesalissa käytettävät laitteistot ovat muuttuneet ja kehittyneet vuosikymmenten aikana, ne tuottavat operoidessaan yhä merkittävän määrän hukkalämpöä. Lämpötilan kohoaminen konesalitiloissa on ongelma, joka aiheuttaa palvelimille suorituskyvyn alenemista ja lämpötilan noustessa merkittävästi laitteiston suojausmekanismit sammuttavat palvelimet peruuttamattomien tuhojen ehkäisemiseksi. Jotkin konesaliympäristössä käytettävästä laitteista ja tarvikkeista, kuten varmuuskopiointinauhat, ovat vaarassa tuhoutua suhteellisen pienenkin lämpötilan kohoamisen seurauksena.

Laitevalmistaja Dell EMC [25] ilmoittaa räkkiasennettavalle PowerEdge R7515 palvelimelle korkeimmaksi operointilämpötilaksi 40 °C. Toinen laitevalmistaja, Hewlett Packard Enterprise (HPE) [47], ilmoittaa räkkiasennettavalle DL380 Gen10 palvelimelle lähes vastaavan operointilämpötilan, 35 °C. HPE ilmoittaa myös, että DL380 Gen10 -palvelimen suorituskykyä rajoitetaan lämpötilan ylittäessä 30 °C. Käytännössä tällöin lasketaan suorittimen kellotaajuutta. Tämä on toiminto, joka ilmenee palvelimen käyttäjälle hitautena.

Puolijohdevalmistaja Intel [58] ilmoittaa Xeon-tuoteperheen prosessoreille maksimilämpötilaksi noin 70 °C. Tämä on raja-arvo, jonka ylittyminen aiheuttaa palvelimen hätäsammutuksen. On huomioitava, että tämä lämpötila tarkoittaa lämpöanturin suoraan prosessorista mittaamaa lämpötilaa, joka on huomattavasti korkeampi kuin huonelämpötila tai palvelinlaitteiston muu lämpötila.

Syyskuussa 2021 julkaistut Linear Tape-Open (LTO) G9 -varmuuskopiointinauhat ovat Fujifilmin [39] antamien tietojen mukaan erityisen herkkiä lämmölle, sillä niiden maksimilämpötila on vain 25 °C. Mikäli konesalissa on käytössä tätä teknologiaa hyödyntävät varmuuskopiointimenetelmät, tämä raja-arvo on erittäin tärkeää huomioida, koska lämpötilan ylittyessä varmuuskopiointinauha voi tuhoutua.

Korkean lämpötilan tavoin myös väärä ilmankosteus voi aiheuttaa ongelmia. Wanin et al. [106] mukaan liian alhainen ilmankosteus kasvattaa konesalin laitteisiin kohdistuvaa sähköstaattisen purkauksen riskiä. Liian korkea ilmankosteus puolestaan lisää heidän mukaansa konesalien energiankulutusta. Howardsin [72] mukaan korkea ilmankosteus aiheuttaa korroosiota liittimille sekä kondensaatiota, joka puolestaan aiheuttaa sähkölaitteille oikosulkuja. Hänen mukaansa äärimmäisissä tapauksissa korkea ilmankosteus voi aiheuttaa virtalähteissä valokaaren, joka sytyttää tulipalon.

Ongelmien välttämiseksi konesalien fyysiset olosuhteet eroavat monin tavoin esimerkiksi tavallisista varasto- tai toimistotiloista. Yksi tärkeimmistä lämpötilaa laskevista tekijöistä on ilmanvaihdon turvaaminen. Tämän vuoksi laitteistot asennetaan laiteräkkeihin. Konesalikäyttöön tarkoitettujen räkkiasennettavien laitteiden leveysstandardi on 19", jolloin ne voidaan asentaa EIA-310 -standardin mukaisiin räkkikaappeihin [31]. Pääsääntöisesti räkeissä on 42 laitepaikkaa. Useimmat räkkipalvelimet vievät yhden laitepaikan, mutta osa konesalilaitteistosta vaatii useampia laitepaikkoja. Räkkiasennettavat palvelimet säästävät myös tilaa verrattuna perinteisiin, tornimallisiin laitteisiin. Lisäksi räkkilaitteiden huollettavuus on parempi, koska niitä voidaan tarpeen mukaan liikuttaa räkkikiskojen varassa.

Konesalien lattia on korotettu, jolloin sähkökaapelointi on mahdollista kuljettaa lattialevyjen alla palvelimien ja UPS-laitteiden välillä. Samoin onnistuu myös tietoliikennekaapelointi. Korotetun lattian ansiosta kaapelit ovat suojassa, mutta niihin pääsee kuitenkin tarvittaessa helposti käsiksi. Lu et al. [70] toteavat artikkelissaan, että korotetulla lattialla on roolinsa myös jäähdytyksessä ja ilmanvaihdossa. Lattialevyjen avulla voidaan erottaa lämmin ja kylmä ilma toisistaan ja ilmaa voidaan ohjata haluttuun paikkaan puhallinlaattojen avulla. Lämpimän ja kylmän ilman erillään pitäminen on tärkeää, koska lämpötila pyritään pitämään konesaleissa optimaalisena. Liiallinen jäähdyttäminen lisää merkittävästi energiankulutusta, vähäinen jäähdyttäminen ja huono ilmanvaihto puolestaan nostavat lämpötilaa liiaksi.

3.2.2 Fyysinen pääsy ja kulunhallinta

Palvelimet ja levyjärjestelmät sisältävät usein tietoja, joiden suojaaminen on organisaation tai yksilön kannalta tärkeää. Tämän vuoksi konesalien pääsynhallintaan ja kulunvalvontaan on kiinnitettävä erityistä huomiota. Konesaleihin on tavallisesti pääsy vain pienellä määrällä ihmisiä ja tiloja valvotaan kameravalvonnan avulla. Perinteisesti pääsynhallinta on toteutettu avaimien ja lukkojen avulla, mutta Kolhe et al. [62] esittävät artikkelissaan, että parempi tapa pääsynhallintaan olisi älylukko, jonka avaaminen tapahtuu älypuhelimella Bluetooth-yhteyden välityksellä. Tämä toimintamalli mahdollistaa myös kulkuoikeuksien poistamisen reaaliaikaisesti. He esittelevät artikkelissaan prototyypin, jossa on muutamia puutteita tuotantokriittistä käyttöä ajatellen, kuten vikasietoisuuden puuttuminen. Markkinoilla on kuitenkin myös käyttövalmiita tuotteita, jotka poistavat perinteisten lukkojen aiheuttamat kulkuoikeuksien poistamiseen liittyvät haasteet. Esimerkiksi ILOQ:in tuoteesitteessä esitellään konesalikäyttöön tarkoitettu lukko, joka voidaan avata älypuhelimella. Bluetoothin sijaan tämä lukko hyödyntää near-field communication (NFC)-teknologiaa [57]. iLOQ perustelee NFC:n käyttöä sillä, että sen käyttöetäisyys on lyhyempi kuin Bluetoothilla, jolloin voidaan välttää epähuomiossa tehtävät ovenavaukset. Lisäksi NFC-teknologiaa käytettäessä ei tarvita ulkoista virtalähdettä lukolle, koska lukko saa toimintaenergian sähkömagneettisen induktion avulla [56].

Myös teknisen tietoturvan näkökulmasta on ongelmallista, jos hyökkääjä pääsee hyökkäyksen kohteena olevaan laitteistoon fyysisesti käsiksi. Kun hyökkääjällä on fyysisesti pääsy laitteeseen, hän voi vaihtaa pääkäyttäjän salasanan haluamakseen, jonka jälkeen hyökkääjä saa rajattoman pääsyn kohdejärjestelmään. Fyysinen pääsy mahdollistaa myös passiiviset hyökkäykset, kuten verkkoliikenteen kuuntelemisen.

Näitä hyökkäyksiä voi olla mahdotonta täysin estää, mutta räkkien valvontaa voidaan parantaa räkkikaappien ovien monitoroinnilla. Kun räkkikaappien ovien tilatietoja seurataan jatkuvasti, voidaan räkin tarkkuudella selvittää, mihin kohteisiin ulkopuolinen henkilö on kajonnut.

3.2.3 Tulipalot ja savuvahingot

Sähkölaitteiden aiheuttamien tulipalojen ennaltaehkäiseminen on haastavaa, koska laitteiden tulevaa vikaantumista ei pääsääntöisesti pysty päättelemään ennakkoon. Usein sähkölaitepaloissa syynä pölyn ja muun lian aiheuttamat ongelmat, mutta konesaliympäristöissä epäpuhtaudet eivät useimmissa tapauksissa ole ongelmien syynä.

Turvallisuus- ja kemikaaliviraston (TUKES) raportissa [46] korostuvat UPS-laitteiden aiheuttamat tulipalot. *Muu rakennuksen sähköverkoston osa* -ryhmään luokitelluista tulipaloista 15 % oli UPS-laitteiden aiheuttamia. On huomioitava, että UPS-laitteita käytetään useissa käyttökohteissa, ei pelkästään konesalitiloissa. On mahdollista, että joidenkin UPS-laitteiden tulipalojen osasyynä voi olla myös esimerkiksi ympäristön epäpuhtaus.

Mikäli konesalin fyysinen turvallisuus on suunniteltu asianmukaisesti, konesali muodostaa oman palotilan, jolloin palo ei pääse leviämään muualle rakennukseen. Tulipalot on kuitenkin pystyttävä havaitsemaan mahdollisimman varhaisessa vaiheessa, koska konesalin sisäinenkin tulipalo on tuhoisa. Pelkät sähköpalon aiheuttamat savuvahingot ovat haastavia ja pistävän hajun poistaminen vaatii tilan otsonoinnin. Korhosen et al. [64] mukaan otsonointi on toteutettava siten, että tila on tyhjä ihmisistä ja tavaroista. Konesaliympäristössä tilan tyhjentäminen on haasteellista, joten tulipalojen varhaisen havaitsemisen tarve korostuu.

Tulipalojen sammutuksessa oikeantyyppinen, automaattinen sammutusjärjestelmä pienentää vahinkoja. Yksittäisiä palavia tietoteknisiä laitteita ei ole järkevää sammuttaa vedellä konesaliympäristössä, koska kosteus aiheuttaa oikosulkuja ja hapettumia. Tämän vuoksi konesaleissa käytetään pääsääntöisesti kaasupohjaisia sammutusjärjestelmiä. Gengin [42] mukaan inerttikaasuja käyttävän sammutusjärjestelmän toiminta perustuu kaasun nopeaan vapautukseen, jolloin vapautuvat kaasut syrjäyttävät hapen ja tämän myötä tulipalo sammuu. Tämä on tehokas menetelmä isommissa tulipaloissa. Haittapuolena on kaasujen vapauttamisesta muodostuva voimakas paineaalto, joka aiheuttaa tuhoja perinteisille, mekaanisille kiintolevyille. Pienissä tulipaloissa vähiten ympäröiville laitteistoille haitallisen sammu-

tusjärjestelmän toiminta perustuu synteettisin kaasuihin. Näiden kaasujen sammutusteho perustuu niiden kykyyn sitoa lämpöä, jolloin tulipalo sammuu lämpötilan laskemisen myötä [42].

3.3 Monitorointiratkaisut

Häiriötilanteiden syntymistä voidaan ennaltaehkäistä monitorointiratkaisujen avulla. Monitoroinnilla tarkoitetaan tässä yhteydessä tilojen olosuhteiden jatkuvaa koneellista seurantaa. Jos olosuhteet muuttuvat niin paljon, että määritetyt raja-arvot ylittyvät, hälytysjärjestelmä lähettää välittömästi tiedon ylläpitäjille. Tällöin ylläpitäjät voivat reagoida ja ongelmatilanne voidaan korjata, ennen kuin palvelinjärjestelmiin aiheutuu käyttökatkoa.

Monitorointia voidaan toteuttaa eri menetelmin ja markkinoilla on tarjolla useita kaupallisia monitorointijärjestelmiä. Useat tarjolla olevista järjestelmistä ovat ominaisuuksiensa puolesta kattavia, mutta samalla myös moniin käyttötarkoituksiin nähden raskaita ja kalliita [38]. Joidenkin valmisratkaisujen kipukohtina on tekninen monimutkaisuus, jolloin yksinkertaisenkin tarpeen täyttämiseksi tarvitaan useita laitteita [5]. Osa hallintaohjelmistoista on vanhentuneita ja niiden käyttäminen vaatii ei-tietoturvallisten apuohjelmien, kuten Javan selainlisäosien asentamista.

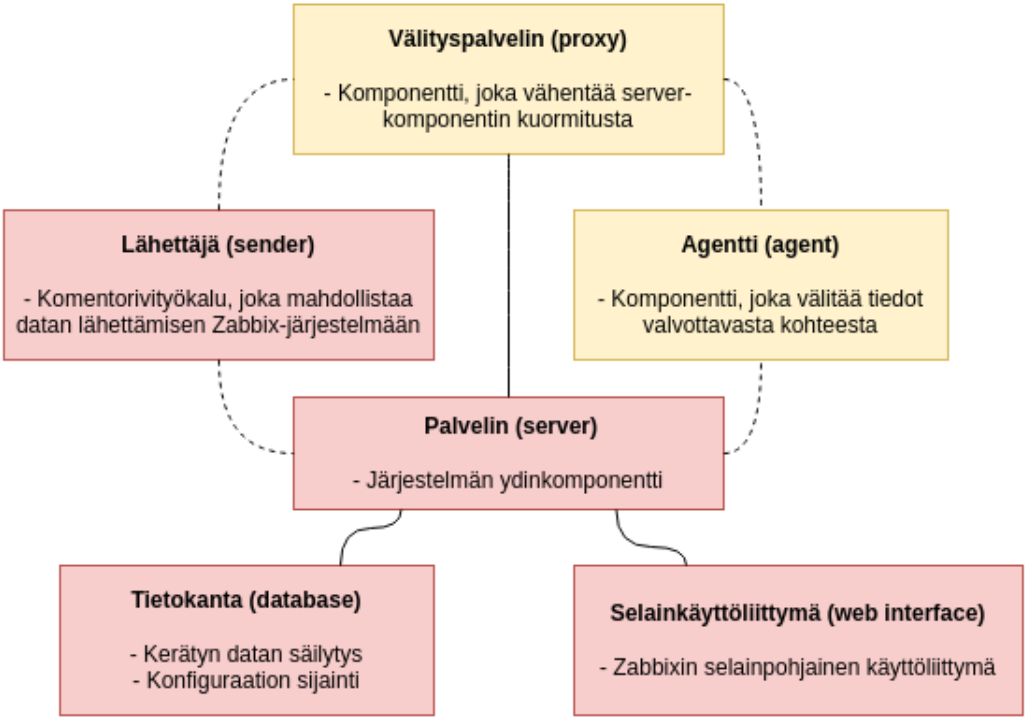
Lähtökohtaisesti hyvänä konesaliolosuhteiden monitorointiratkaisuna voidaan pitää järjestelmää, joka kykenee tekemään tarvittavien kohteiden monitoroinnin, skaalautuu erikokoisiin konesaliympäristöihin sekä integroituu olemassa oleviin monitorointiohjelmistoihin, jolloin ylläpito on yksinkertaisempaa. Tämän työn empiirisessä osassa suunniteltava ja toteutettava mittauslaite tuottaa dataa, jonka hyödyntäminen tapahtuu monitorointisovelluksen avulla. Yksi laajasti käytetyistä monitorointisovelluksista on Zabbix, jonka monipuoliset ominaisuudet mahdollistavat sen käyttämisen osana erityyppisiä kokonaisuuksia. Zabbix-järjestelmän laajuuden vuoksi tässä tutkielmassa käsitellään vain tutkielman kannalta merkitykselliset ominaisuudet ja toiminnallisuudet.

3.3.1 Zabbix-monitorointisovellus

Zabbix on monitorointisovellus, joka on saatavissa vapaasti GNU GPLv2-lisenssin mukaisesti [111]. Projektin on käynnistänyt Alexei Vladishev vuonna 1998 havaitessaan, että saatavissa olevat vaihtoehdot ovat hankinta- ja ylläpitohinnaltaan kalliita

[104]. Ensimmäinen julkisesti saatavissa oleva versio Zabbixista julkaistiin vuonna 2001. Vuosien aikana Zabbix-monitorointijärjestelmän ympärille on muodostunut yhtiö, Zabbix LLC, joka tarjoaa kaupallisia tukipalveluita Zabbix-järjestelmään liittyen. Yhtiöllä on toimistoja viidessä valtiossa ja sen johtohenkilönä toimii yhä projektin käynnistänyt Vladishev [109].

Zabbix-järjestelmän arkkitehtuuri koostuu sovelluskomponenteista [112]. Kaikki komponentit eivät ole pakollisia, vaan niitä voidaan asentaa ja käyttää käyttötarpeen mukaisesti. Kuvassa 3.1 esitetään komponentit ja niiden väliset riippuvuudet. Komponentteja yhdistävät, yhtenäiset viivat kuvaavat komponenttien keskinäistä riippuvuutta. Katkoviivalla yhdistetyt komponentit ovat riippuvaisia toisistaan, mutta eivät yhtäaikaisesti. Tämän tutkielman empiirisessä osassa rakennettavan mittauslaitteen tuottaman tietojen tallentamisen ja hyödyntämisen kannalta merkityksellisten komponenttien taustaväri on punainen. Komponentteja, joiden taustaväri on keltainen, ei tässä yhteydessä hyödynnetä. Mittauslaitejärjestelmän Zabbix-ympäristössä käytetään neljää komponenttia, jotka ovat palvelin, selainkäyttöliittymä, lähettäjä ja tietokanta.



Kuva 3.1: Zabbix-monitorointisovelluksen komponentit

Palvelin

Palvelinkomponentti on Zabbixin keskeisin osa, joka mahdollistaa kerätyn datan hyödyntämisen sekä muun muassa hälytysten lähettämisen. Palvelimen vastaanottamat tiedot tallennetaan relaatiotietokantaan, esimerkiksi MariaDB-tietokantasovellukseen [114].

Selainkäyttöliittymä

Selainkäyttöliittymä on nimensä mukaisesti komponentti, joka mahdollistaa selainpohjaisen käyttöliittymän. Tärkeimmät selainpohjaisen käyttöliittymän tarvitsemat komponentit ovat PHP ja webpalvelinohjelmisto, kuten Nginx [116].

Agentti

Jos monitoroitavan laitteen käyttöjärjestelmä on Linux, Windows, Mac OS X tai se on BSD-pohjainen, datan kerääminen tapahtuu helpoiten agenttikomponentin avulla. Agentin avulla saadaan helposti selville laitteen perustiedot, kuten kiintolevyjen vapaa tila, suorittimen kuormitus sekä tieto palveluiden käynnissäolosta [110].

Lähettäjä

Lähettäjä on komentorivityökalu, joka mahdollistaa datan lähettämisen palvelinkomponentille [115]. Koska IoT-laitteissa ei useinkaan ole käyttöjärjestelmää, agenttia ei voida asentaa. Tällöin tarvittavat tiedot kerätään manuaalisesti IoT-laitteen rajapintojen kautta. Kerätyt tiedot lähetetään palvelimelle lähettäjä-komponentin avulla. Usein manuaalinen tietojenkeräys ja tietojen lähettäminen voidaan automatisoida esimerkiksi Bourne again shell (Bash) -skriptien ja cron-ajastuksen avulla.

Välityspalvelin

Välityspalvelin on valinnainen komponentti, jota voidaan käyttää datalähteen ja palvelimen välisenä kauttakulkupisteenä [113]. Välityspalvelimella on kaksi käyttötarkoitusta. Ensimmäinen on palvelimen kuormituksen vähentäminen. Pienissä ympäristöissä tämä ei ole välttämätöntä, mutta monitoroitavien kohteiden määrän lisääntyessä palvelimen kuormitus kasvaa. Välityspalvelimia voidaan ottaa käyttöön myös useampia, jos käyttötarve niin vaatii. Toinen käyttötarkoitus liittyy verkkotopologiaan. Joissain tapauksissa voi olla tarpeen rajoittaa suoria yhteyksiä Zabbix-palvelimeen. Tällöin voidaan estää kokonaan lähdejärjestelmien ja Zabbix-pal-

velimen suorat yhteydet ja kierrättää kaikki liikenne välityspalvelinten kautta.

Kaikki edellämainitut komponentit voidaan asentaa yhdelle palvelimelle, mutta tarpeen mukaan ne voivat toimia myös erillisillä palvelimilla. Zabbix-järjestelmä ei ole riippuvainen yksittäisistä kolmannen osapuolen tietokanta- tai web-palvelinohjelmistoista eikä sitä ole välttämätöntä käyttää yhdessä mainittujen MariaDB:n tai Nginx:n kanssa [114]. Tuettuja vaihtoehtoja ovat muun muassa myös PostreSQL-tietokantasovellus sekä Apache-webpalvelin. Zabbix-palvelinkomponentit voidaan asentaa useille Linux-jakeluversioille BSD-pohjaisille käyttöjärjestelmille.

3.3.2 IoT-monitorointiratkaisut

Zabbixin tai muun itseasennettavan monitorointijärjestelmän käyttäminen ei ole ainoa vaihtoehto IoT-laitteiden tuottaman datan vastaanottamiseksi ja hyödyntämiseksi. Useat toimijat tarjoavat pilvipalveluita, jotka ovat kehitetty erityisesti IoT-laitteiden tarpeita varten.

Yksi vaihtoehdoista on ThingSpeak, joka mahdollistaa tietojen vastaanottamisen mistä tahansa laitteesta, jolla on kyvykkyys tehdä lähetystä HTTP- tai MQ Telemetry Transport (MQTT) -protokollien avulla [98]. Palvelun ominaisuuksiin kuuluu datan reaaliaikainen visualisointi sekä hälytyksien lähettäminen sähköpostiin. Integraatiot muihin verkkopalveluihin, kuten Twitter-viestipalveluun, ovat myös mahdollisia [99]. ThingSpeak-palvelun pienimuotoinen käyttäminen on ilmaista eikaupallisessa käytössä, kun vastaanotettavien sanomien lukumäärä on alhaista [97]. ThingSpeakin käyttäminen on maksullista, kun palvelua hyödynnetään kaupallisessa toiminnassa. Tämän lisäksi hintaan vaikuttavia tekijöitä ovat käytettävien IoT-laitteiden lukumäärä ja niiden aktiivisuus, tukipalveluiden tarve sekä kehittyneiden data-analytiikkaan liittyvien työkalujen tarve [100].

ThingSpeak-palvelinohjelmisto on saatavissa vapaasti GNU GPL v3-lisenssin mukaisesti, jolloin palvelu voidaan ottaa käyttöön myös pilvipalveluriippumattomasti. Julkisesta GitHub-versiohallinnasta saatavissa olevat ohjelmistoversiot [96] ovat kuitenkin selkeästi vanhentuneet, eikä selkeää kehityspolkua ole löydettävissä.

Zohari et al. [120] ovat toteuttaneet prototyypin, jonka tehtävänä on mitata konesalin lämpötilaa ja ilmankosteutta. Mittauslaitteen prototyyppi pohjautuu Arduino Uno ja Espresso Lite V2.0 -kehitysalustoihin sekä toiminnan kannalta tarpeellisiin sensoreihin. Dataa ei lähetetä pilvipalveluun, vaan paikallisesti asennettuun ThingSpeak-palvelimeen, jonka alustana toimii Raspberry Pi -tietokone. Zoharin et

al. mukaan toteutettu kokonaisuus mahdollistaa konesalin asianmukaisen valvonnan, jolloin ThingSpeakia voidaan pitää käyttökelpoisena IoT-laitteiden tuottaman datan säilytys- ja käyttöalustana ainakin tässä kontekstissa.

Myös suuret IT-alan toimijat, kuten Amazon, Google ja Microsoft tarjoavat palveluitaan IoT-laitteiden tuottaman datan keräämiseen ja prosessointiin. Muhammedin ja Ucuzin [75] tekemän vertailun perusteella palveluissa on käyttäjän näkökulmasta havaittavia eroavaisuuksia. Vertailu osoittaa, että sopivimman palvelun löytämiseksi käyttötarpeet on oltava tarkasti tiedossa. Toisaalta on myös huomioitava, että palveluntarjoajien kova keskinäinen kilpailu asiakkaista ja markkinaosuuksista ohjaa heitä kehittämään palveluitaan jatkuvasti, jolloin tutkimuksissa ja vertailuissa olevat tiedot vanhenevat nopeasti.

4 Tuotekehitysmenetelmät

Tässä luvussa tarkastellaan laitteiston ja sovelluksen kehitystyötä erilaisten tuotekehitysmenetelmien avulla. Esitellään erilaisia tekniikoita sekä tehdään havaintoja tekniikoiden vahvuuksista sekä haasteista. Näkökulmana on uuden IoT-laitteen toteutus, jolloin tuotekehitysprosessi alkaa laitteistovaihtoehtojen kartoittamisella ja valinnoilla sekä sisältää ohjelmiston toteuttamisen. Tuotekehityksen lopputuloksena on kokonaisuus, joka täyttää sille asetetut vaatimukset.

4.1 Tuotekehityksen perusvaiheet

Tuotekehityksessä on vaiheita, jotka on tehtävä riippumatta käytetystä menetelmästä. Rastogin [86] mukaan onnistuneen tuotekehityksen edellytys on, että seuraavat seitsemän vaihetta on kartoitettu ja huomioitu asianmukaisesti. Vaiheet ovat projektisuunnittelu, vaatimusanalyysi, ominaisuuksien mallintaminen ja suunnittelu, toteutusvaihe, testausvaihe, käyttöönotto sekä ylläpito ja jatkokehitys.

Projektisuunnittelussa on nimensä mukaisesti tavoitteena suunnitella tulevaa projektia. Tämän vaiheen tärkein tavoite on selvittää, minkälaisen ongelman ratkaisemisesta projektissa on kyse, sekä hahmottaa ongelman laajuus esimerkiksi resurssitarvetta varten.

Vaatimusanalyysissa perehdytään ratkaistavaan ongelmaan tarkemmin ja kartoitetaan, minkälaisia toimintoja toteutettavassa laitteessa tulee olla, jotta ongelma saadaan ratkaistua. Tässä vaiheessa tulee myös huomioida rajoitteita asettavat tekijät, kuten esimerkiksi laitteiston suorituskyky.

Ominaisuuksien mallintamisen ja suunnitteluvaiheen aikana tehdään tarkat suunnitelmat siitä, miten aiemmissa vaiheissa määritetty ongelma aiotaan ratkaista. Laitetasolla tässä vaiheessa valitaan sopivimmat komponentit. Sovellustasolla suunnitellaan ohjelmiston rakenne sekä valitaan käytettävä ohjelmointikieli. Erityisen tärkeää on myös selvittää, että kaikki tehdyt valinnat ovat keskenään yhteensopivia.

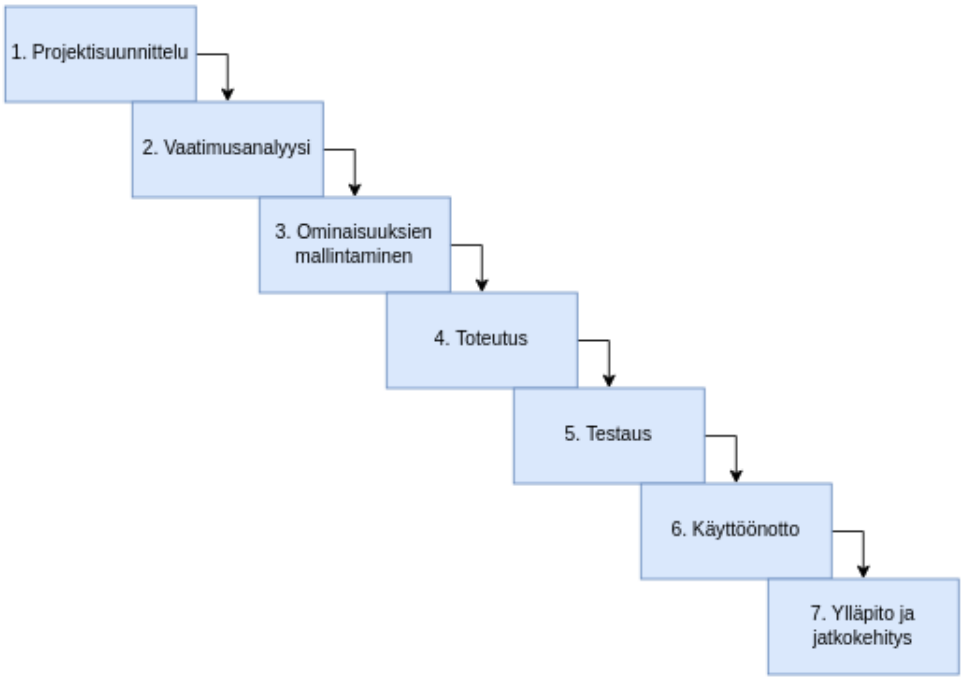
Toteutusvaiheessa laitteisto kootaan sekä ohjelmisto toteutetaan laadittujen suunnitelmien mukaisesti. Testausvaiheen aikana on tavoitteena havaita mahdolliset ohjelmointivirheet sekä havaita muut käyttäjäkokemusta heikentävät ongelmat.

Kun projektille asetettujen vaatimusten toteutus on valmis ja testit on tehty onnistuneesti, projektin tuotos on valmis käyttöönottoa varten. Käyttöönoton jälkeen siirrytään olemassa olevan tuotteen ylläpitoon sekä jatkokehitykseen. Ylläpitovaiheeseen kuuluu mahdollisten virheiden korjaus. Jatkokehitystä tarvitaan esimerkiksi silloin, jos valmiiseen tuotteeseen halutaan lisätä uusia ominaisuuksia, jotka eivät kuuluneet alkuperäiseen suunnitelmaan.

4.2 Vesiputousmalli

Alshamranin ja Bahattabin [6] mukaan vesiputousmalli on yksi vanhimmista ja tunnetuimmista ohjelmistokehitysmalleista. Mallin ominaispiirteenä on peräkkäin tapahtuvat vaiheet, jolloin seuraavaan vaiheeseen ei siirrytä ennen kuin meneillään oleva vaihe on valmistunut. Heidän mukaansa mallin vahvuutena on sen helppo ymmärrettävyys ja selkeys, jolloin se sopii käytettäväksi kokemattomillekin kehitystiimeille.

Mallin heikkoutena voidaan pitää sen joustamattomuutta. Koska kehitystyö etenee vaihe kerrallaan suoraan kohti käyttöönottoa, loppukäyttäjällä ei ole mahdollisuutta päästä näkemään kehitysversioita, vaan tulokset on nähtävissä vasta projektin loppupuolella. Tämä aiheuttaa ongelmia erityisesti silloin, jos vaatimusanalyysissa on epäselvyyksiä tai kaikkia tarpeita ei ole otettu huomioon. Kuvassa 4.1 esitetään vesiputousmallin tehtävät sekä niiden vaiheistus.



Kuva 4.1: Vesiputousmallin vaiheistus

Vesiputousmalli sopii käytettäväksi silloin, kun asetetut vaatimukset on hyvin selkeät. Alshamrani ja Bahattab mainitsevat vesiputousmallin olevan käyttökelpoinen silloin, kun lopullinen tuote on jo olemassa, mutta tuotetta ollaan siirtämässä erilaiselle alustalle [6].

4.3 Prototyyppimalli

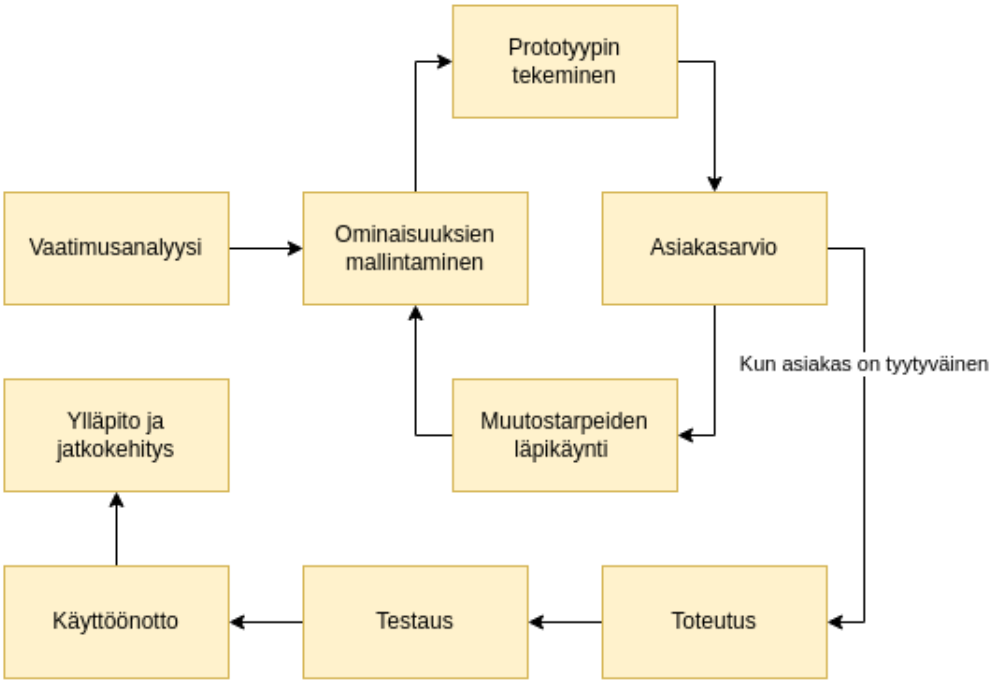
Vesiputousmallin yksi isoimmista haasteista on se, että tulokset nähdään vasta projektin loppupuolella. Tämän epäkohdan aiheuttamia ongelmia voidaan välttää prototypoinnin avulla. Wiegersin ja Beattyn [108] mukaan prototyyppien avulla on mahdollista selventää tai täydentää uudelle tuotteelle asetettuja vaatimuksia. He korostavat, että prototyyppiin ei ole tarkoitus sisällyttää koko tuotteen kaikkia ominaisuuksia, vaan prototypointia on tarkoituksenmukaisinta tehdä niissä osuuksissa, joiden toimintaperiaatteessa on suurimmalla todennäköisyydellä epäselvyyksiä.

Prototyypit voidaan jakaa kahteen alakategoriaan niiden sisältämien toimintojen perusteella. Nämä alakategoriat ovat malli ilman toimintalogiikkaa (engl. mock-up) sekä konseptitodistus (engl. proof-of-concept). Malli ilman toimintalogiikkaa

sopii esimerkiksi käyttöliittymän mallintamiseen, jolloin loppukäyttäjä saa käsityksen siitä, miltä lopputulos tulisi ulkoisesti näyttämään. Konseptitodistuksen avulla voidaan todentaa esimerkiksi sovelluksen jonkin osion tekninen toimivuus sekä suorituskyky.

Oikein toteutetuista prototyypeistä on hyötyä loppukäyttäjälle ja tuotekehitystiimille. Parhaimmillaan prototyypit säästävät merkittävästi aikaa ja työpanosta, kun ongelmakohdat voidaan havaita ajoissa. Prototyyppien tekemisessä on kuitenkin muistettava realiteetit, eli on vältettävä sellaisten mallien tekemistä, jotka poikkeavat lopullisesta tuotteesta. On myös muistettava, että erityisesti loppukäyttäjille näytettävien prototyyppien tulisi olla sellaisia, jotka eivät ole liian viimeistelemättömiä, koska liian viimeistelemätön prototyyppi voi antaa vääränlaisen kuvan tuotekehitystiimin työskentelystä.

Kuvassa 4.2 esitetään prototyyppimallin tehtävät sekä niiden vaiheistus. Huomioitavaa on, että prototyyppejä voidaan tehdä niin useasti kuin on tarpeen.



Kuva 4.2: Prototyyppimallin vaiheistus

Prototyyppimalli sisältää samoja tuotekehitysvaiheita kuin vesiputousmalli. Merkityksellisin ero on vesiputousmalliin verrattuna on säännöllinen asiakasarvioiden tekeminen.

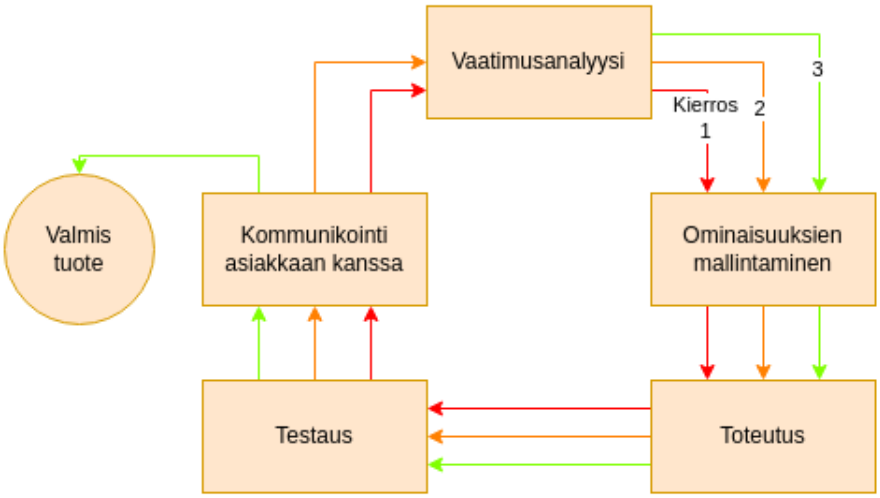
4.4 Spiraalimalli

Bohmin [17] mukaan spiraalimalli on menetelmä, joka yhdistää vesiputousmallin ja prototyyppimenetelmän. Spiraalimallissa kehitys etenee vaiheittain, kuten vesiputousmallissa. Spiraalimallissa kokonaisuutta ei kuitenkaan tehdä kerrallaan alusta loppuun saakka, vaan kehitys tapahtuu osa kerrallaan. Jokaisen kehityskierroksen lopputulosta voidaan kutsua prototyypiksi. Kierrosmäärän kasvaessa tavoitteellisuus kasvaa. Tämän kehityskaaren myötä varhaisimpien kehityskierrosten lopputulokset voivat olla myöhempien vaiheiden kannalta merkityksettömiä, kun puolestaan myöhäisemmät prototyypit ovat käytännössä käyttövalmiita, lopputuloksen kannalta merkittäviä osia.

Spiraalimalli tarjoaa loppukäyttäjälle näkyvyyden tuotekehityksen etenemiseen prototyyppimenetelmän tavoin. Spiraalimallin hyvänä puolena on myös vastaavanlainen selkeys kuin vesiputousmallissa. Alshamranin ja Bahattabin [6] mukaan spiraalimallin heikkoutena on tehottomuus tilanteissa, joissa kehityskierrosten aikana tehdään useita erittäin pieniä muutoksia. Tällöin suunnitteluun ja prototyyppien tekemiseen voi kulua runsaasti aikaa.

Spiraalimalli on tehokas ja käyttökelpoinen malli silloin, kun projektilla arvioidaan olevan merkittäviä riskitekijöitä, kuten useampia vaatimuksien sekä tarpeiden muutoksia tuotekehitysprojektin aikana.

Kuvassa 4.3 esitetään spiraalimallin kehityskierrokset, jotka johtavat lopulta valmiiseen tuotteeseen. Kuvassa on kolme kierrosta, joista kahden ensimmäisen kierroksen tuloksena on prototyyppi. Viimeisen kierroksen tuloksena on valmis tuote. Kehityskierrosten määrä vaihtelee projektikohtaisesti.



Kuva 4.3: Spiraalimallin vaiheistus

Spiraalimallissa yhdistetään vesiputousmallin järjestelmällinen eteneminen prototyyppimallin iterointiin, jolloin vaiheet tapahtuvat peräkkäin kuten vesiputousmallissa, mutta jokaisen vaiheen jälkeen loppuvaiheessa kommunikoidaan asiakkaan kanssa. Tällöin asiakas hahmottaa paremmin projektin edistymisen ja pystyy osaltaan vaikuttamaan siihen, että projekti etenee kohti halutunlaista lopputulosta. Asiakas pääsee myös vaikuttamaan jokaisen iteraatiokierroksen alussa tehtävään vaatimusmäärittelyyn, jolloin jokaisen iteraation lopputuloksena valmistuvan prototyypin tulisi olla asiakastarpeiden mukainen.

4.5 Yhteenveto kehitysmenetelmistä

Tuotekehitystä voidaan tehdä erilaisin kehitysmenetelmin. Merkityksellistä on ymmärtää, että ei ole olemassa yhtä kehitysmallia, joka sopisi kaikkiin tarpeisiin. Sopivan tuotekehitysmallin valintaan vaikuttaa useat tekijät, joita ovat muun muassa projektin tavoitteiden selkeys, riskit tavoitteiden muuttumiseen projektin aikana, projektitiimin kokemustaso sekä projektin suuruus. Väärän kehitysmenetelmän valinta voi johtaa projektin viivästymiseen, kustannuksien nousuun tai pahimmillaan lopputuloksen epäonnistumiseen. On myös muistettava, että kehitysmenetelmiä voi muokata tarpeen mukaan, jolloin tuotekehityksestä saadaan ketterämpää.

5 Mittauslaitteen suunnittelu ja toteutus

Tässä luvussa esitellään mittauslaitteen fyysinen toteutus ja mittauslaitteeseen toteutettu ohjelmisto. Lisäksi tarkastellaan, kuinka mittauslaitteella kerättävää dataa voidaan hyödyntää konesaliympäristön olosuhteiden valvonnassa.

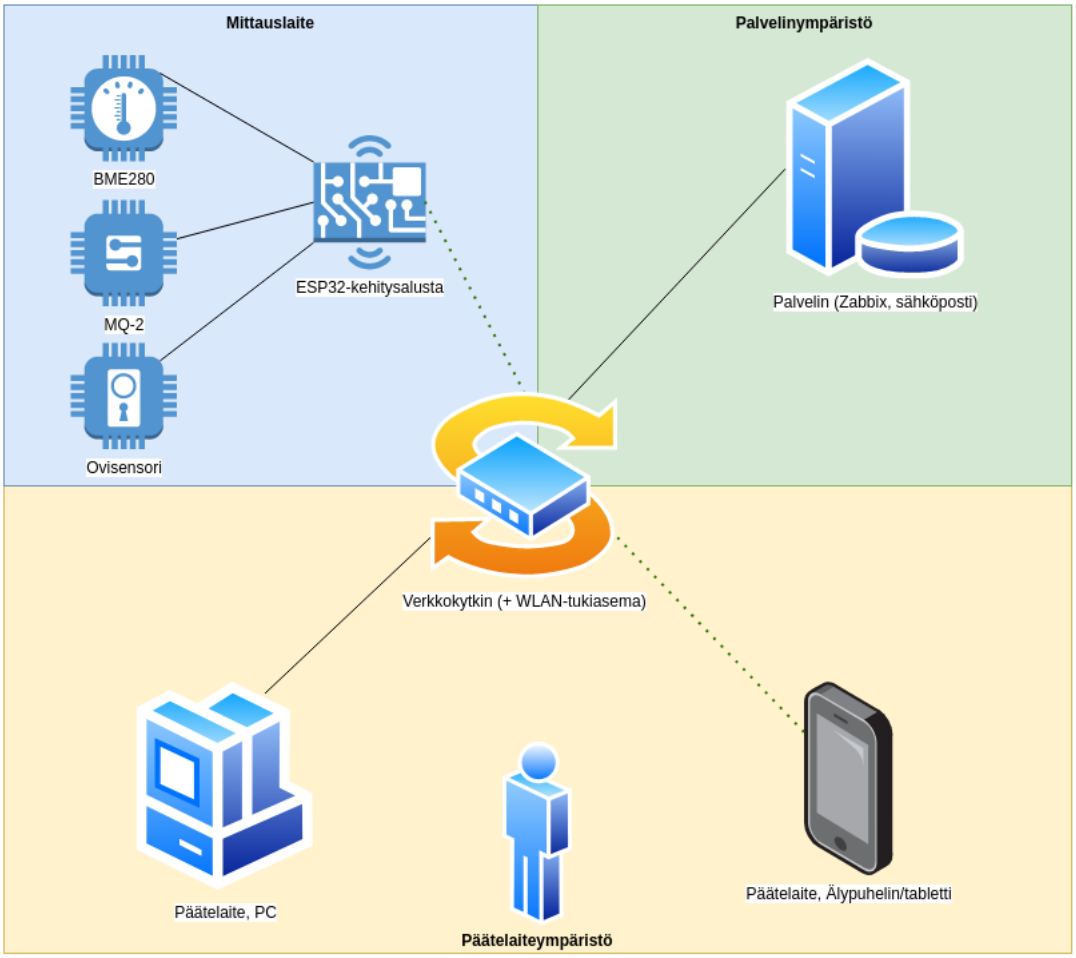
Alaluvussa 5.1 esitellään lyhyesti mittauslaitekokonaisuuden arkkitehtuuri. Alaluvussa 5.2 kuvataan mittauslaitteen toimintaperiaate sekä esitellään, kuinka mittauslaite on integroitu Zabbix-monitorointijärjestelmään. Alaluvussa 5.3 esitellään tärkeimmät mittauslaitteessa käytettävät komponentit ja syyt, miksi ne ovat valittu käytettäväksi mittauslaitteessa. Alaluvussa 5.4 esitellään mittauslaitteen suunnittelu ja toteutus spiraalimallin mukaisesti. Alaluvussa 5.5 esitellään mittauslaitteeseen toteutettu ohjelmisto.

5.1 Arkkitehtuurin yleiskuvaus

Mittauslaitteen ohjelmiston suunnittelussa on huomioitu laitteen integroitavuus eri kohdejärjestelmiin, jolloin dataa voidaan tarvittaessa helposti hyödyntää Zabbix-monitorointijärjestelmän lisäksi muissakin kohdejärjestelmissä ilman mittauslaitteeseen tehtäviä ohjelmistomuutoksia. Liitettävyys WLAN-verkkoon ja HTTPS-palvelun kautta tapahtuva määrämuotoinen tiedonsiirto mahdollistavat mittauslaitteen liittämisen osaksi mihin tahansa palveluun, joka pystyy vastaanottamaan dataa HTTPS-kyselyiden kautta.

Kuvassa 5.1 olevassa arkkitehtuurimallissa on kuvattu, miten komponentit ja laitteet kytkeytyvät toisiinsa. Arkkitehtuuriin kuuluu kolme osiota, jotka ovat mittauslaite, palvelinympäristö ja päätelaiteympäristö. Tietoliikennetekniikan näkökulmasta edellä mainitut kolme osiota kuuluvat samaan lähiverkkoon. Mittauslaitteessa käytetyt sensorit ovat kytketty ESP32-kehitysalustan GPIO-pinneihin. ESP32 on yhdistetty WLAN-verkkoon. Palvelin ja päätelaitteet voidaan yhdistää WLAN-verkkoon tai Ethernet-kaapelilla verkkokytkimeen.

Järjestelmän tarkempi arkkitehtuurikuva tiedonsiirtojen toteutuksesta on tutkielman liitteessä C.



Kuva 5.1: Monitorointijärjestelmän yksinkertainen arkkitehtuurimalli

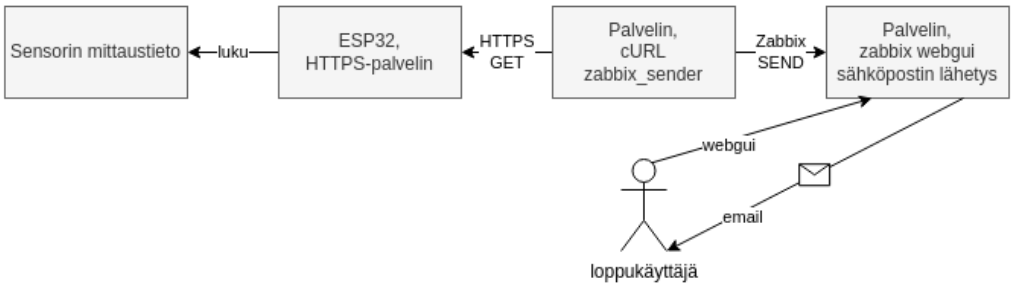
5.2 Mittauslaitteen toimintamalli

Mittauslaitteen tehtävänä on tuottaa dataa, jonka perusteella voidaan tehdä analyysejä konesalin olosuhteista. Mittauslaitteen avulla voidaan monitoroida kolmea suuretta, jotka ovat konesalin lämpötila, ilmankosteus ja huoneilman savupitoisuus. Lisäksi laitteen avulla saadaan tieto, onko räkkikaapin ovi auki vai kiinni. Kattavan tilannekuvan muodostamiseksi tarvitaan säännöllistä mittausta sekä datan tallentamista, jotta nykyhetken tilannetta voidaan verrata aiempiin ajankohtiin. Numeerisessa muodossa oleva data visualisoidaan graafien avulla, jotta sen tulkinta olisi mahdollisimman yksinkertaista. Mittaustuloksille määritetään raja-arvot, joiden ylittyminen aiheuttaa sähköpostiherätteen lähettämisen ylläpitäjälle.

Mittauslaitteessa itsessään ei ole riittävästi tallennuskapasiteettia datan pitkäaikaista säilyttämistä varten ja laitteen rajalliset tekniset resurssit eivät myöskään mahdollista visuaalisten näkymien luomista. Datan säilytys mittauslaitteessa ei ole tarkoituksenmukaistakaan, koska mittauslaitteen tehtävänä on pelkästään tehdä mittaukset, sekä mahdollistaa mittausdatan siirtäminen palvelinympäristöön. Mittauslaitekokonaisuuden arkkitehtuuri mukailee kuvassa 2.1 kuvattua Ciscon IoT-referenssimallia. Palvelimeen on asennettu Zabbix-monitorointisovelluksen tarvitsemat komponentit sekä sähköpostipalvelut herätteiden lähettämistä varten.

Kokoonpanon ensisijaisena tavoitteena on tarjota konesalin ylläpitäjälle työkalu, jonka avulla saadaan reaaliajassa tiedot konesalin olosuhteiden muuttumisesta epäsuotuisaksi, kuten fysikaalisten olosuhteiden muutokset, luvattomat räkkikaappien ovien avaukset sekä huoneilman savupitoisuuden kasvaminen. Toissijaisena tavoitteena on luoda visuaaliset näkymät, joiden avulla voidaan nähdä, ovatko konesaliolosuhteiden muuttuneet pidemmällä aikavälillä. Tällöin voidaan havaita, jos esimerkiksi jäähdytysjärjestelmän suorituskyky on liian alhainen suhteessa konesalilaitteiston tuottamaan hukkalämpöön. Tämäntyyppinen tilanne voi ilmetä, jos laitteiston määrää lisätään konesalissa.

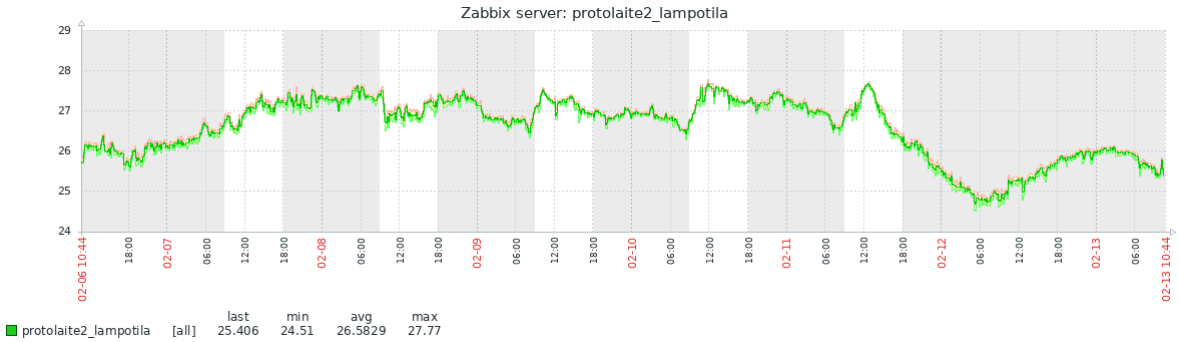
Kuvassa 5.2 esitetään, kuinka data kerätään mittauslaitteesta monitorointijärjestelmään ja havainnollistetaan prosessiin liittyvät osat.



Kuva 5.2: Datan kerääminen ja sen käsittelyprosessi

Datan keräämisen alkaa, kun palvelimen cron-ajastuspalvelussa ajastettu komentosarja lähettää ESP32-kehitysalustassa kuuntelevalle HTTPS-palvelimelle GET-kutsun. Kutsun vastaanottamisen jälkeen mittauslaitteen sensorit tekevät olosuhteiden mittaukset. Tämän jälkeen HTTPS-palvelin palauttaa mittausarvot määrämuotoisesti. Komentosarja käsittelee vastaanotetun datan ja erottelee datasta kunkin sensorin mittaamat arvot. Erottelun jälkeen arvot lähetetään Zabbix-palvelimelle

lähettäjä-komponentin avulla. Kommentoitu Bash-komentosarja on tämän tutkielman liitteessä B. Kun Zabbix-palvelin on vastaanottanut datan, se tallennetaan tietokantaan. Tämän jälkeen data on luettavassa muodossa ja mittaustulokset ovat käyttäjälle nähtävissä selainkäyttöliittymän avulla. Kuvassa 5.3 olevasta esimerkkigraafista nähdään lämpötilan kehitys seitsemän vuorokauden ajalta.



Kuva 5.3: Mitatun lämpötilan visualisointi seitsemän vuorokauden ajalta

Selainkäyttöliittymän avulla voidaan myös tehdä määritykset raja-arvoille, joiden ylittyessä ylläpitäjälle lähetetään sähköposti. Kuvassa 5.4 on ylempänä esimerkkiviesti havaitusta huoneilman savupitoisuuden kasvamisesta ja alempana viesti, joka lähetetään kun havaittu ongelmatilanne on korjaantunut.

From: zabbix@a530.jjlan.local
To: jukka@a530.jjlan.local
Subject: Ongelma: Protolaite2, savuarvo on yli 300
Date: Wed, 20 Apr 2022 11:36:05 +0300

```
Ongelma on: Protolaite2, savuarvo on yli 300
Ongelma on alkanut: 2022.04.20 klo 11:36:03
Mitattu arvo, joka aiheutti hälytyksen: 301
Ongelman ID: 1303
```

From: zabbix@a530.jjlan.local
To: jukka@a530.jjlan.local
Subject: Korjaantunut: Protolaite2, savuarvo on yli 300
Date: Wed, 20 Apr 2022 11:37:05 +0300

```
Ongelma oli: Protolaite2, savuarvo on yli 300
Ongelma korjaantunut: 2022.04.20 klo 11:37:02
Ongelma kesti: 59s
Ongelman ID: 1303
```

Kuva 5.4: Sähköpostiviesti käyttäjälle ongelmatilanteen alkaessa ja korjaantuessa

Hälytysviestien tietosisältö on muokattavissa tarpeiden mukaisesti. Viestit on muotoiltu siten, että niiden avulla ylläpitäjä saa nopeasti perustiedot ongelmasta. Reaaliaikainen kuittausviesti ongelman korjaantumisesta on tärkeä, koska tieto ongelman korjaantumisesta muuttaa vallitsevaa tilannekuvaa merkittävästi.

5.3 Mittauslaitteen laitteistokokoonpano

Mittauslaitteen laitteistokokoonpanossa komponenttien tärkeimpänä valintakriteerinä on valittujen komponenttien kyky toteuttaa toimintamallissa määritetyt tehtävät. Komponenttien keskinäinen yhteensopivuus on perusedellytys järjestelmän toiminnalle. Osa komponenteista on sellaisia, jotka olisi voitu lopputuloksen näkökulmasta korvata toisellakin osalla. Tällöin valittiin osa, jonka käyttäminen on tarkoituksenmukaisinta. Tässä alaluvussa esitellään ne komponentit, jotka valittiin toteutettuun mittauslaitteeseen sekä kerrotaan, miksi valittu komponentti on tarkoituksenmukaisin. Mittauslaitteen tarkka kytkentäkaavio on tutkielman liitteessä D.

Järjestelmän ydin
Mittauslaitteen ydinkomponentiksi on valittu ESP32-järjestelmäpiirin sisältävä kehitysalusta, NodeMCU-ESP32 Devkit V1. Teknisten ominaisuuksien näkökulmasta kyseinen kehitysalusta sopii käyttötarkoitukseen hyvin, sillä ESP32-järjestelmäpiirin suorituskyky on riittävä ja liitäntämahdollisuudet sensorien kytkemiseksi ovat kattavat. ESP32 sisältää myös tässä käyttötarkoituksessa tarvittavan kyvykkyyden WLAN-verkkoon liittämistä varten. Teknisesti toimiva ratkaisu voitaisiin toteuttaa myös Raspberry Pi -tietokoneen tai Arduino Uno -kehitysalustan avulla, mutta kokonaisuutta ajatellen ESP32-järjestelmäpiiriin perustuva alusta on paras vaihtoehto. Raspberry Pi -tietokoneeseen perustuvan alustaratkaisun haittapuolena on laitteen huomattavasti korkeampi yksikköhinta, suurempi energiankulutus sekä monimutkaisempi toteutustapa verrattuna mikrokontrollereihin perustuviin toteutustapoihin. Arduino Uno puolestaan tarvitsee lisämoduulin WLAN-liitettävyyttä varten, joka monimutkaistaa hieman mittauslaitteen suunnittelua ja toteutusta.

Lämpötilan ja kosteuden mittaus
Lämpötilan ja ilmankosteuden mittaamiseen käytetään BME280-sensoria. Se on ominaisuuksiltaan tarkoituksenmukainen, sillä yhdellä sensorilla voidaan tehdä kummatkin mittaukset. Lisäksi sen tukemat mittausalueet soveltuvat hyvin normaalei-

hin konesaliympäristöihin. BME280 toimii sekä 3.3V ja 5V käyttöjännitteillä, jolloin se on yhteensopiva ESP32-kehitysalustan kanssa. Adafruit Industries tarjoaa ohjelmakirjaston, joka mahdollistaa laitteistoyhteensopivuuden sekä helpottaa laitteen ohjelmistosuunnittelua.

Savupitoisuuden mittaus
Huoneilman savupitoisuuden havaitsemiseen käytetään MQ-2-sensoria. MQ-2 toimii 5 voltin käyttöjännitteellä. Sensori sopii käytettäväksi ESP32-kehitysalustan kanssa, mutta on huomioitava, että ESP32:n GPIO-pinnit kestävät vain 3.3 voltin jännitettä, kun MQ-2:n ulostulojännite on tätä suurempi [117]. Kun jännitettä pienennetään vastuksien avulla, vältetään ESP32-kehitysalustan rikkoontuminen. Huoneilman savupitoisuuden mittaamisessa käytetään MQ-2 sensorin analogisesta ulostuloa, josta luetaan mittausarvo raakamuodossa. Mittauslaitteen ohjelmatasolla dataa ei tulkita eikä muunneta, vaan datankäsittely jätetään kohdejärjestelmän tehtäväksi. Tämä mahdollistaa mittauskohdekohtaisten asetusten määrittämisen. On myös huomioitava, että MQ-2-sensori havaitsee huoneilmasta myös erilaisia palavia kaasuja, jolloin virhehavainnot savupitoisuudesta on mahdollisia. Tavallisissa konesaliympäristöissä tämä ei kuitenkaan aiheuta ongelmia, koska normaalitilanteissa niiden huoneilmassa ei ole mittaustuloksia vääristäviä tekijöitä.

Räkkikaapin oven tilatieto
Oven tilatiedon tunnistamiseen käytetään geneeristä, laajasti saatavissa olevaa magneettikytkintä. Magneettikytkin on tyypiltään normally closed (NC). Toteutuksessa voitaisiin käyttää myös magneettikytkintä, joka on tyypiltään normally open (NO). Tässä käyttötapauksessa mittauslaitteen ohjelmakoodiin olisi tarpeen tehdä pieniä muutoksia, koska nämä kytkintyypit toimivat toisiinsa nähden päinvastaisesti.

5.4 Mittauslaitteen toteutus spiraalimallin mukaisesti

Tässä alaluvussa kuvataan mittauslaitteen toteutus spiraalimallin mukaisesti. Spiraalimalli on mittauslaitteen kehitystyössä käyttökelpoinen tuotekehitysmenetelmä, koska mittauslaitetta voidaan pitää kokonaisuutena, joka koostuu toisistaan riippumattomista moduuleista. Tällöin tuotekehitystä voidaan tehdä moduuli kerrallaan, jatkaen prototyypin kehittämistä edellisen iteraatiokierroksen tuloksen pohjalta. Iteraatiokierrosten vaatimuksien ydinajatuksena on, että laitteen toiminnan kannalta

tärkeimmät ominaisuudet toteutetaan mahdollisimman varhaisessa vaiheessa. Mittauslaitetta ei toteuteta ulkoiselle asiakkaalle. Tämän vuoksi kommunikaatio asiakkaan kanssa on korvattu omilla havainnoilla ja huomioilla.

5.4.1 Ensimmäinen iteraatiokierros

Vaatimusanalyysi
Ensimmäisen iteraatiokierroksen vaatimuksena on mittauslaitteen kyvykkyys tuottaa testidataa. Tämä testidata on oltava haettavissa HTTP GET -kutsujen avulla. Tämä vaatimus on määritetty, koska mittauslaitteen toimintamalli rakentuu HTTP-palvelimen ympärille. Jos HTTP-palvelin todetaan mahdottomaksi toteuttaa, kehitysprojekti ei voi edetä seuraaviin iteraatiokierroksiin ennen kuin laitteen toimintamalli on suunniteltu erilaiseksi.

Ominaisuuksien mallintaminen
Esivaatimuksena HTTP-palvelimen käyttöönotolle on ESP32-kehitysalustan liittäminen WLAN-verkkoon. WLAN-verkkoon liittämistä ja HTTP-palvelun käyttöönottoa varten otetaan käyttöön ohjelmakirjastot, jotka mahdollistavat tarvittavat toiminnot.

Toteutus
Toteutusvaiheessa otetaan käyttöön ohjelmakirjastot *WiFi.h* ja *WebServer.h* sekä kirjoitetaan vaatimukset täyttävä ohjelmistokoodi. Sensoreita ei kytketä kehitysalustaan tässä vaiheessa. Iteraation toteutus on valmis, kun laite voidaan liittää WLAN-verkkoon ja HTTP-palvelu vastaa asiakasohjelman pyyntöihin, palauttaen testidataa.

Testaus
Ensimmäisen iteraatiokierroksen testausvaiheen kuuluu laitteen liittäminen WLAN-verkkoon ja HTTP-palvelun kutsuminen. Testi on suoritettu onnistuneesti, jos laitteen liittäminen WLAN-verkkoon onnistuu ja laite vastaa HTTP-kutsuihin onnistuneesti. Tämän iteraatiokierroksen testien onnistumisella on riippuvuus tuleviin iteraatiokierroksiin, koska tulevat testit eivät voi onnistua, jos kehitysalusta ei kykene vastaamaan HTTP-kutsuihin.

Havainnot ja huomiot
Toteutus tehtiin vaatimusanalyysin mukaisesti. Testaus suoritettiin onnistuneesti. Koska HTTP-palvelu toimii halutulla tavalla, voidaan kehitysprojektissa jatkaa seuraavaan iteraatiokierrokseen eikä laitteen toimintamallia tarvitse muuttaa.

5.4.2 Toinen iteraatiokierros

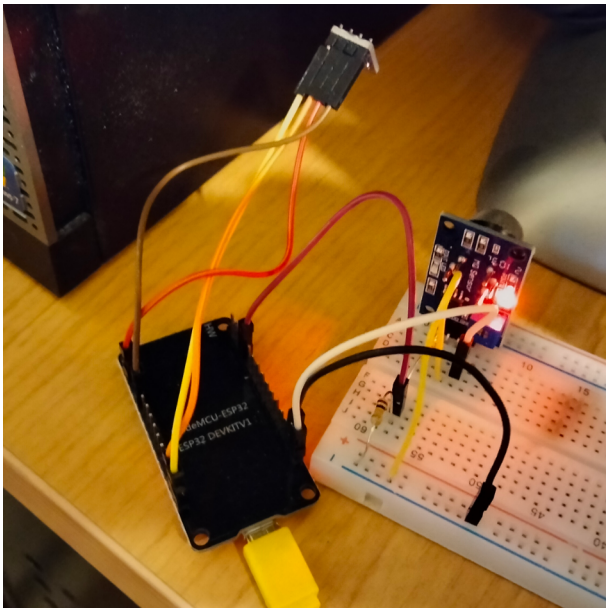
Vaatimusanalyysi
Toisen iteraatiokierroksen jälkeen mittauslaitteella tulee olla kyvykkyys mitata lämpötilaa, ilmankosteutta ja ilman savupitoisuutta. Laitteen pitää palauttaa edellä mainitut arvot, kun niitä kysytään HTTP-kutsujen avulla.

Ominaisuuksien mallintaminen
Vaatimuksien toteuttamiseksi ESP32-kehitysalustaan kytketään laitteistokokoonpanossa 5.3 mainitut sensorit BME280 ja MQ-2. Kytkentöjen helpottamiseksi sensorit kiinnitetään koekytkentäalustaan, josta tehdään johdotukset kehitysalustaan.

Toteutus
Kytketään edellämainitut sensorit kytkentäkaavion D mukaisesti. Mittauslaitteen ohjelmistossa otetaan käyttöön kirjasto *Adafruit_BME280.h*, joka mahdollistaa lämpötilan ja ilmankosteuden mittaukset. Tehdään tarvittavat muutokset ohjelmakoodiin. Iteraation toteutus on valmis, kun laite mittaa onnistuneesti huoneilman savupitoisuutta, lämpötilaa ja kosteutta sekä palauttaa mittaustiedot HTTP-palvelun kautta asiakkaalle.

Testaus
Toisen iteraatiokierroksen testausvaiheessa todennetaan, että kaikkien mitattavien kohteiden mittaustulos palautuu asiakkaalle HTTP-palvelun kautta. Testit on suoritettu onnistuneesti, jos laite palauttaa reaaliajassa mittausten tulokset.

Havainnot ja huomiot
Toteutus tehtiin vaatimusanalyysin mukaisesti ja testaus suoritettiin onnistuneesti. Mittauslaitteen kehitystä voidaan jatkaa tämän iteraariokierroksen pohjalta. Kuvassa 5.5 esitetään mittauslaite toisen iteraatiokierroksen jälkeen.



Kuva 5.5: Mittauslaite rakennettuna koekytkentäalustalle. Toinen iteraatiokierros.

5.4.3 Kolmas iteraatiokierros

Vaatimusanalyysi

Edellisen iteraation vaatimusten lisäksi mittauslaitteella tulee olla kyvykkyys havaita, onko räkkikaapin ovi auki vai kiinni. Oven tilatieto palautetaan, kun sitä kysytään HTTP-kutsun avulla. Lisäksi vaaditaan, että laite on käyttökelpoinen konesaliympäristössä, jolloin se ei voi olla rakennettuna koekytkentäalustalle.

Ominaisuuksien mallintaminen

Jotta räkkikaapin oven tilatieto voidaan selvittää, ESP32-kehitysalustaan lisätään magneettikytkin kytkentäkaavion D mukaisesti. Konesalikäytön mahdollistamiseksi mittauslaitteen kokoonpano siirretään koekytkentäalustalta laitekoteloon.

Toteutus

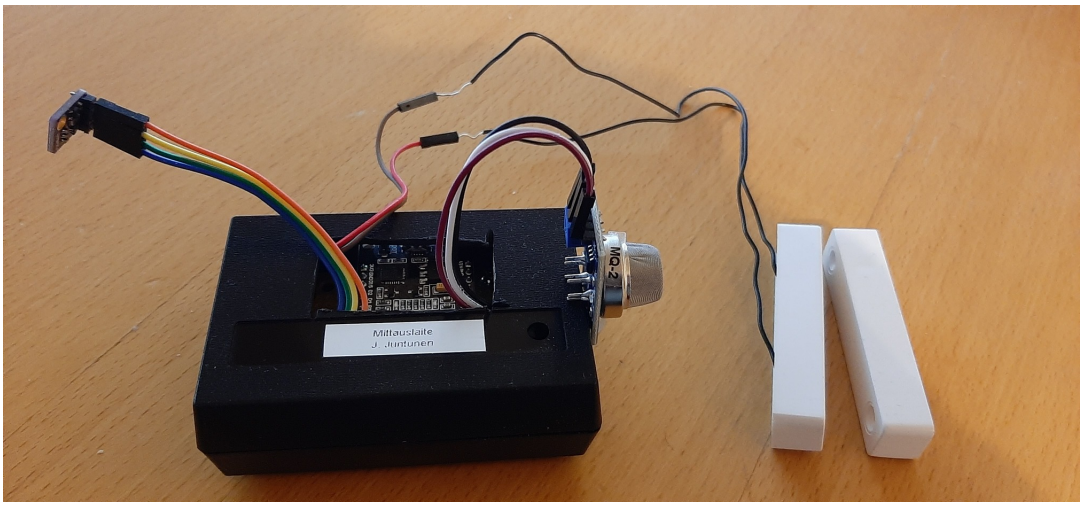
Kolmannen iteraatiokierroksen toteutusvaiheessa kytketään magneettikytkin kehitysalustaan ja tehdään tarvittavat muutokset ohjelmakoodiin sekä asennetaan komponentit laitekoteloon konesalikäyttöä varten. Iteraation toteutus on valmis, kun laitekoteloon asennettu mittauslaite mittaa edellisen iteraation toteutuksen lisäksi räkkikaapin oven tilatietoa ja palauttaa mittaustiedot HTTP-palvelun kautta asiakkaalle.

Testaus

Kolmannen iteraatiokierroksen testausvaiheessa todennetaan, että kaikkien mitattavien kohteiden mittaustulos palautuu asiakkaalle HTTP-palvelun kautta. Todennetaan lisäksi silmämääräisesti, että komponentit on asennettu laitekoteloon asianmukaisesti. Testit on suoritettu onnistuneesti, jos laite palauttaa reaaliajassa mittausten tulokset ja jos komponenttien asennuksessa ei havaita puutteita.

Havainnot ja huomiot

Toteutus tehtiin vaatimusanalyysin mukaisesti ja testaus suoritettiin onnistuneesti. Mittauslaitteen fyysinen toteutus on valmis kolmannen iteraatiokierroksen jälkeen ja se mahdollistaa toimintamallissa 5.2 kuvatut mittaukset. Kuvassa 5.6 esitetään mittauslaite kolmannen iteraatiokierroksen jälkeen.



Kuva 5.6: Mittauslaitteen koteloitu versio. Kolmas iteraatiokierros.

5.4.4 Neljäs iteraatiokierros

Vaatimusanalyysi
Edellisen iteraation vaatimusten lisäksi mittauslaitteen ja asiakasohjelman välinen tiedonsiirto on muutettava selväkielisestä salatuksi.

Ominaisuuksien mallintaminen
Jotta tiedonsiirto voidaan toteuttaa salatusti, on siirryttävä HTTP-protokollasta HTTPS-protokollaan.

Toteutus
Mittauslaitteen ohjelmistossa vaihdetaan HTTP-palvelun käyttämä kirjasto *WebServer.h* toiseen kirjastoon *ESPWebServerSecure.hpp*, joka tukee HTTPS-protokollaa. Luodaan Transport Layer Security (TLS)-salauksen vaatimat sertifikaattitiedostot *serti.h* ja *avain.h* sekä toteutetaan muut tarvittavat muutokset ohjelmakoodiin.

Testaus
Neljännen iteraatiokierroksen testausvaiheessa todennetaan, että kaikkien mitattavien kohteiden mittaustulos palautuu asiakkaalle HTTPS-palvelun kautta. Testi on suoritettu onnistuneesti, jos laite palauttaa reaaliajassa mittausten tulokset. Mittauslaite ei saa enää vastata salaamattomiin HTTP-kyselyihin.

Havainnot ja huomiot
Toteutus tehtiin vaatimusanalyysin mukaisesti. Testaus suoritettiin onnistuneesti, mutta lopullisessa toteutuksessa havaittiin ongelma. Alkuperäisen suunnitelman mukaisesti jokaisen sensorin mittaustulos haettiin erillisellä HTTP/HTTPS-kutsulla, jolloin neljän sensorin mittaustulosten saamiseen tarvittiin neljä kutsua. HTTP-protokollaa käytettäessä tämä oli toimiva tapa. HTTPS-protokollaan siirtyessä kyselyt muuttuvat salauksen vuoksi raskaammiksi, jolloin ESP32-järjestelmäpiirin suorituskykyyn liittyvät rajoitteet tulevat vastaan. Käytännössä mittauslaite ei pysty vastaamaan tiheästi tapahtuviin HTTPS-kyselyihin, vaan menee asiakasohjelman näkökulmasta jumiin. Tämän havainnon perusteella voidaan todeta, että iteraatioiden testitapaukset ovat olleet osittain puutteellisia, koska ne eivät huomioineet riittävästi lopullista käyttötapausta. Ongelma oli kuitenkin mahdollista kiertää muuttamalla sovelluksen toimintalogiikkaa siten, että kaikkien sensoreiden mittaustulos palautetaan yhdellä HTTPS-kyselyllä, jolloin suorituskykyongelmia ei ilmene.

5.5 Mittauslaitteen ohjelmisto

Mittauslaitteen ohjelmisto on toteutettu Arduino IDE -kehitysympäristön avulla. Ohjelmoinnissa on käytetty C++ -ohjelmointikieltä. Ohjelmiston suunnittelun lähtökohtana on mahdollistaa sensoreiden käyttäminen ohjelmistokirjastojen avulla ja mahdollistaa tiedonsiirto HTTPS-protokollan avulla, WLAN-verkkoa hyödyntäen. Tässä alaluvussa esitellään otteita ohjelmiston tärkeimmistä osuuksista. Mittauslaitteen kommentoitu lähdekoodi on kokonaisuudessaan tämän työn liitteessä A, pois lukien sertifikaattitiedostot ja kirjastojen lähdekoodit.

Kirjastot

Mittauslaitteen ohjelmisto hyödyntää valmiita ohjelmistokirjastoja kolmessa toiminnossa, jotka ovat WLAN-tietoliikenne, HTTPS-palvelu sekä BME280-sensorin mittaukset. Tarvittavat kirjastot otetaan käyttöön *#include*-direktiivin avulla. Näiden kirjastojen käyttäminen nopeuttaa ohjelmistokehitystä huomattavasti, koska ne sisältävät kaiken tarpeellisen ohjelmakoodin mainittujen toimintojen hyödyntämiseksi. Tällöin ohjelmoijan tarvitsee ainoastaan kutsua funktioita haluttujen toimintojen toteuttamiseksi. Listauksessa 5.1 esitetään tarvittavien kirjastojen käyttöönotto.

```
1 // Kirjasto WLAN varten
2 #include <WiFi.h>
3 // Kirjasto HTTPS-palvelua varten
4 #include <ESPWebServerSecure.hpp>
5 // Kirjasto anturille BME280
6 #include <Adafruit_BME280.h>
```

Listaus 5.1: Ote lähdekoodista, kirjastot

Setup-funktio

Setup-funktiolla on erityinen rooli, sillä se suoritetaan automaattisesti laitteen käynnistymisen yhteydessä. Tärkeimmät tässä yhteydessä tapahtuvat toiminnot ovat sensoreihin tehtävät asetusmääritykset, WLAN-verkkoon yhdistäminen sekä HTTPS-palveluun tehtävät määritykset, kuten TLS-sertifikaatin käyttöön ottaminen ja HTTPS-kyselyissä käytettävien polkujen määrittäminen. Listauksessa 5.2 esitetään setup-funktion tärkeimmät toiminnot.

```
1  // Setup-funktio
2  // Suoritetaan laitteen kaynnistymisen yhteydessa
3
4  void setup(void) {
5    // BME280: Aloitetaan I2C-liikenne
6    bme.begin(0x76);
7
8    // Yhdista WLAN-verkkoon
9    WiFi.begin(ssid, salasana);
10
11   // Asetetaan HTTPS-sertifikaatti
12   // 'avain' ja 'avain_pituus' loytyy tiedostosta avain.h
13   // 'serti' ja 'serti_pituus' loytyy tiedostosta serti.h
14   server.setServerKeyAndCert(
15     avain,          // privaattiavain DER-muodossa, byte array
16     avain_pituus,   // pituus: privaattiavain DER-muodossa, byte array
17     serti,          // sertifikaatti DER-muodossa, byte array
18     serti_pituus    // pituus: sertifikaatti DER-muodossa, byte array
19   );
20
21   // Kaynnistetaan HTTPS-palvelu
22   server.on("/kaikki", handle_Kaikki);
23   server.begin();
24 }
```

Listaus 5.2: Ote lähdekoodista, setup-funktio

Mittaustulosten palauttaminen

Kun HTTPS-palvelu vastaanottaa GET-kutsun polkuun */kaikki*, suoritetaan funktio *handle_Kaikki*. Tämän funktion sisällä tehdään kutsuhetkellä vallitsevan lämpötilan, ilmankosteuden ja savuarvon mittaaminen sekä räkkikaapin oven tilatiedon tarkastaminen. Kaikkien palautettavien arvojen tietotyyppi muutetaan String-tyyppiseksi. Mittausten ja tilatiedon tarkastamisen sekä tietotyyppimuunnoksen jälkeen palvelin palauttaa asiakkaalle tiedon onnistuneesta HTTPS-kyselystä. IETF-organisaation RFC 7231-standardissa [37] on määritetty onnistuneen kyselyn statuskoodiksi 200. Statuskoodin lisäksi palautetaan hyötykuormana mitatut tiedot määrämuotoisesti formaatissa *lämpötila*|*kosteus*|*savuarvo*|*ovitieto*. Listauksessa 5.3 esitetään *handle_Kaikki* -funktion toimintaperiaate.

```
1  // handle_Kaikki -funktio
2  // Suoritetaan, kun vastaanotetaan GET-kutsu polkuun /kaikki
3
4  void handle_Kaikki() {
5    // Luetaan 'lampotila' muuttujaan lampotila BME280-sensorilta
6    float lampotila = bme.readTemperature();
7    // Luetaan 'kosteus' muuttujaan kosteus BME280-sensorilta
8    float kosteus = bme.readHumidity();
9    // Luetaan 'savuarvo' muuttujaan mq2pinni:sta arvo
10   int savuarvo = analogRead(mq2pinni);
11   // Luetaan 'ovitila' muuttujaan ovipinnin tila
12   ovitila = digitalRead(ovipinni);
13
14   // Jos ovipinni on HIGH, ovi on kiinni. Jos LOW, ovi on auki.
15   // Asetetaan 'ovipalautus' muuttujaan joko 'kiinni' tai 'auki'
16   if (ovitila == HIGH) {
17     ovipalautus = "kiinni";
18   } else {
19     ovipalautus = "auki";
20   }
21
22   // Palautetaan arvot muodossa: 'lampotila|kosteus|savu|ovi'
23   server.send(200, "text/html", String(lampotila) + "|" + String(kosteus) + "|" +
         String(savuarvo) + "|" + String(ovipalautus));
24 }
```

Listaus 5.3: Ote lähdekoodista, funktio mittaukseen ja tulosten palautukseen

Muut funktiot

Ohjelmisto sisältää myös kaksi muuta HTTPS-palveluun liittyvää funktiota, jotka ovat *handle_Root* ja *handle_404*. Nämä eivät ole mittauslaitteen toiminnan kannalta välttämättömiä, mutta niiden avulla on toteutettu käsittely polulle / ja niille poluille, joita ei ole olemassa. Puuttuville poluille palautetaan statuskoodiksi 404, RFC 7231-standardin mukaisesti [37]. Listauksessa 5.4 esitetään funktio *loop*. Tämä funktio on laitteen toiminnan näkökulmasta tärkeä, koska loopin sisällä suoritettava *server.handleClient()* -funktio käsittelee HTTPS-asiakkailta saapuvat pyynnöt.

```
1  // loop-funktio
2  // Loopin sisalla kasitellaan asiakkaan HTTPS-pyynnot
3
4  void loop(void) {
5    server.handleClient();
6  }
```

Listaus 5.4: Ote lähdekoodista, loop-funktio

6 Pohdinta ja jatkokehitys

Alaluvussa 6.1 pohditaan, kuinka hyvin mittauslaitteen suunnittelu ja toteutus onnistuivat. Onnistumisen arviointi perustuu alaluvussa 5.2 mainittujen mittauskohteiden ja datankäsittelyyn liittyvien toimintojen toimivuuden arviointiin. Alaluvussa 6.2 pohditaan, kuinka laitteistoa ja datankäsittelyä voitaisiin jatkokehittää. Jatkokehitystarpeiden arvioinnissa pohditaan, miten havaittuja ongelmia voidaan korjata sekä millä tavoin toimintaa voitaisiin muuttaa käyttäjän näkökulmasta paremmaksi. Lisäksi pohditaan, mitä asioita tulee huomioida kun olemassa olevaan laitteeseen kehitetään uusia ominaisuuksia.

6.1 Mittauslaitteen arviointi

Mittauslaitteeseen suunnitellut ja toteutetut ominaisuudet toimivat halutulla tavalla. Kehitetyn mittauslaitteen viimeisin iteraatio on ollut käytössä testiympäristössä yli neljän kuukauden ajan. Koko tämän ajan se on pystynyt toteuttamaan alaluvussa 5.2 mainitut mittaustarpeet. Tiedonsiirto ja datankäsittely ovat tapahtuneet luotettavasti, eikä tallennetun mittausdatan aikaleimojen perusteella ole havaittu puuttuvia mittaustuloksia. Yksittäisen mittauskerran siirtäminen mittauslaitteelta Zabbixiin tapahtuu lokien mukaan noin kahdessa sekunnissa. Myös Zabbixin tuottamat visuaaliset näkymät sekä sähköpostihälytykset ongelmatilanteissa toimivat halutulla tavalla. Seuraavissa kappaleissa esitellään tarkemmin eri kohteissa havaitut huomiot.

Mittaukset ja ympäristön havainnointi
Mittauslaite kykenee mittaamaan luotettavasti huoneilman lämpötilaa ja ilmankosteutta. Vertailukohteena on käytetty toista lämpö- ja ilmankosteusmittaria, jonka antamat lukemat ovat yhtenevät mittauslaitteen antamien lukemien kanssa. Myös magneettikytkimen avulla toteutetun räkkikaapin oven tilatiedon monitoroiminen onnistuu. Testauksen perusteella magneettikytkin toimii tarkasti, jolloin vähänkin raollaan oleva räkkikaapin ovi antaa hälytyksen. Huoneilman savupitoisuutta mittaavaa MQ-2 -sensoria ei voida pitää täysin luotettavana. Ensimmäinen MQ-2 sen-

sori rikkoontui noin kahden viikon käytön jälkeen. Rikkoontuneen sensorin tilalle asennettu uusi sensori on toiminut luotettavasti useiden kuukausien ajan, mutta sen antamat mittaustulokset ovat säännönmukaisesti poikenneet rikkoontuneen sensorin antamista lukemista. On myös huomioitava MQ-2 sensorin kyvykkyys mitata huoneilmasta palavia kaasuja. Sensori ei pysty tällöin tarkasti erottamaan, onko huoneilmassa savua vai muita normaalista huoneilmasta poikkeavia kaasuyhdisteitä. Tällöin on mahdollista, että monitorointijärjestelmään asetetut yksinkertaiset hälytysrajat eivät toimi halutulla tavalla, jolloin ylläpitäjälle aiheutuu virheellisiä hälytyksiä. Normaalissa konesaliympäristössä ei kuitenkaan ole mittaustuloksia vääristäviä tekijöitä ja testien perusteella sensori reagoi selkeästi huoneilman savupitoisuuden, joten MQ-2 sopii tyydyttävästi käyttötarkoitukseen. Tuotantokriittistä toimintaa ajatellen on kuitenkin todennäköistä, että markkinoilta löytyisi käyttötarkoitukseen paremmin soveltuvia vaihtoehtoja.

Tiedonsiirron toimivuus

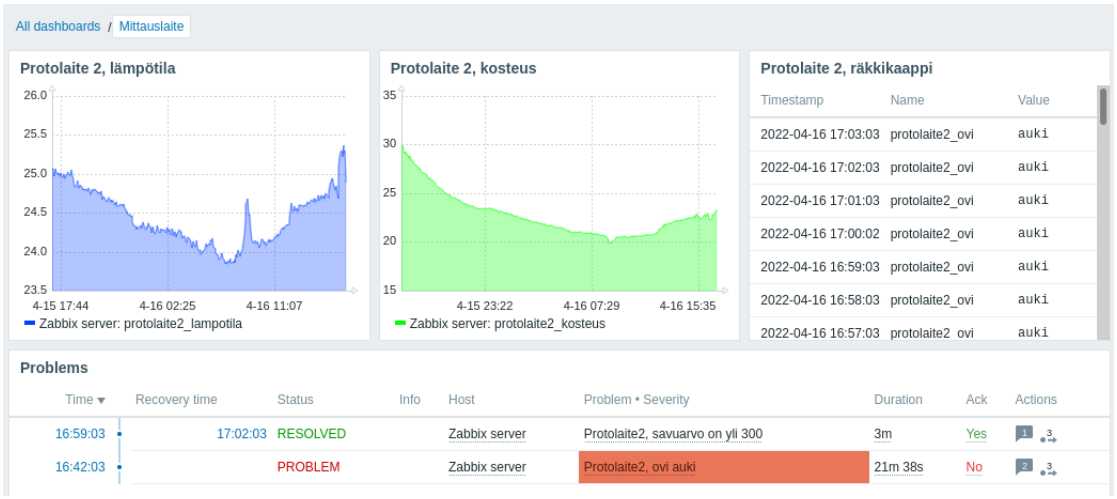
ESP32-järjestelmäpiiriin perustuva kehitysalusta toimii testausten perusteella luotettavasti. Järjestelmäpiirin sisältyvän WLAN-komponentin ja sen käyttämisen mahdollistavien ohjelmakirjaston toiminnassa ei ole havaittu puutteita tai ongelmia. Kokonaisuutta ajatellen WLAN mahdollistaa yksinkertaisimman tiedonsiirtotavan, koska useissa tapauksissa WLAN-verkko on jo toteutettu konesaliympäristöön tai vaihtoehtoisesti se voidaan toteuttaa kustannustehokkaasti. Tämän toteutusmallin myötä mittauslaite ja kohdepalvelin pystyvät kommunikoimaan TCP/IP -verkossa, jolloin tiedonsiirto laitteiden välillä onnistuu ilman erillisiä yhdyskäytäväratkaisuja.

HTTPS-protokollaan perustuva automatisoitu tiedonsiirto mittauslaitteelta monitorointijärjestelmään toimii vakaasti alaluvussa 5.4.4 esitetyn viimeisimmän iteraatiokierroksen muutoksen jälkeen, jolloin kaikki mittaustulokset haetaan yhdellä kyselyllä.

Datan visualisointi ja hälytysten toimivuus

Zabbix-monitorointijärjestelmän avulla datan visualisointi onnistuu halutulla tavalla. Kuvassa 6.1 esitetään esimerkkitoteutus ylläpitäjän näkymästä, jonka avulla saadaan nopeasti yleiskuva konesaliympäristön tilanteesta. Näkymästä ilmenee konesalin huoneilman lämpötila, ilmankosteus, räkkikaapin oven tilatieto sekä aktiiviset ja lähiaikoina korjaantuneet hälytykset. Ylläpitäjä pystyy valitsemaan, kuinka pitkältä ajalta lämpötilan ja kosteuden historiadataa halutaan tarkastella. Ongelma-

luettelosta havaitaan, että kuvankaappauksen ottohetkellä räkkikaapin ovi on ollut auki yli 21 minuutin ajan, eikä sitä ole vielä suljettu. Savusensori on mitannut huoneilmasta kohonnutta savupitoisuutta kolmen minuutin ajan, jonka jälkeen savupitoisuus on laskenut hälytysrajan alapuolelle ja ongelma on merkitty korjaantuneeksi. Näkymää voidaan muokata tarpeiden mukaisesti ja samassa näkymässä voidaan esittää useiden datalähteiden tuottamaa informaatiota.



Kuva 6.1: Zabbix, ylläpitäjän näkymä

Myös sähköpostihälytykset toimivat asianmukaisesti. Ongelmatilanteen alkaessa sähköposti lähetetään samanaikaisesti, kuin tilannetieto päivittyy ylläpitäjän näkymään. Vastaavasti tapahtuu myös ongelmatilanteen korjaantuessa.

Spiraalimallin soveltuvuus

Spiraalimalli havaittiin sopivaksi tuotekehitysmenetelmäksi. Useiden suunnitelmallisten iteraatiokierrosten ansiosta yhdellä kerralla tehtävät muutokset ovat hallittuja ja kohdistuvat kerrallaan yhteen toimintoon. Kehityskohteiden jaottelu iteraatiokierroksiin helpottaa kokonaistyömäärän arviointia. Myös tuotekehityksen aikana havaitut muutos- ja ominaisuuksien lisäystarpeet on mahdollista huomioida spiraalimallin avulla, koska toteutettavien ominaisuuksien priorisointia ja toteutusjärjestystä voidaan vaihtaa tarpeiden mukaan.

Spiraalimalli sopii myös isompiin tuotekehitysprojekteihin, joissa sovelluksen ja laitteiston kehitys jaetaan useammalle kehittäjälle. Tällöin korostuu kokonaisuuden tarkoituksenmukainen jakaminen itsenäisiin osa-alueisiin, jolloin osa-alueita voidaan kehittää samanaikaisesti ilman riippuvuuksia muihin osa-alueisiin.

6.2 Jatkokehitys

Mittauslaitteen jatkokehitystarpeita voidaan tarkastella useista näkökulmasta. Näkökulmia voivat olla esimerkiksi lisätoimintojen tarve, laitteen vakaan toiminnan turvaaminen sekä tietoturvallisuuden parantaminen.

Lisätoiminnot

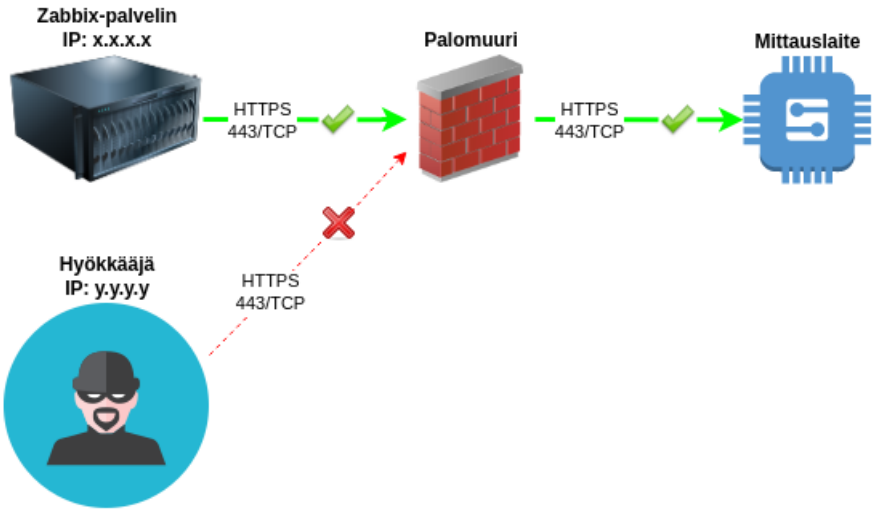
Mittauslaitteeseen voidaan toteuttaa lisätoimintoja esimerkiksi alaluvussa 2.2.3 mainittujen sensoreiden ja aktuaattoreiden avulla. Esimerkiksi kulunvalvontaa voidaan parantaa liiketunnistinsensorin avulla, jolloin monitorointijärjestelmään voidaan kerätä dataa konesalissa havaitusta liikkeestä. Liiketunnistimen avulla voidaan toteuttaa myös vartiointia siten, että monitorointijärjestelmä lähettää kaikesta virka-ajan jälkeen havaitusta liikkeestä sähköpostihälytyksen. Solenoidin avulla on mahdollista toteuttaa räkkikaapin oven sähköinen avausmekanismi, joka mahdollistaa tarkan lokituksen oven avauksista. Räkkikaapin oven avaamisen lokitusta voidaan kehittää lisäämällä oven avauspyyntöön tunnistautumisen, jolloin aikaleimojen lisäksi voidaan lokittaa oven avaamisen suorittanut henkilö.

Tämänhetkisessä ohjelmistoversiossa WLAN-verkkoon liittymiseen tarvittavat WLAN-verkon nimi ja salasana ovat määritetty kiinteästi ohjelmakoodiin. Jos verkon nimi tai salasana muuttuvat, mittauslaite ei enää pysty liittymään verkkoon ja ohjelmakoodia joudutaan muokkaamaan ongelman ratkaisemiseksi. Käyttäjäystävällinen näkökulma olisi mahdollistaa WLAN-määritysten muokkaaminen Bluetoothin avulla, jolloin WLAN-määritysten muuttuminen ei aiheuta tarpeita ohjelmakoodin muuttamiseen.

Laitteen vakaus ja tietoturvallisuus

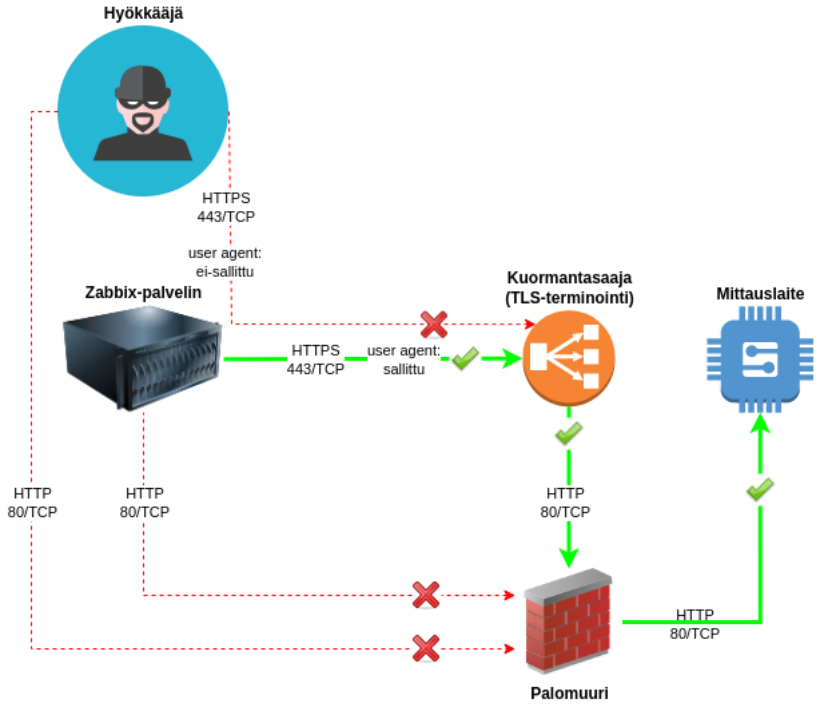
Alaluvussa 5.4.4 havaittiin ongelma, joka ilmenee useiden peräkkäisten HTTPS-kyselyiden jälkeen. Ongelman myötä laite ei kykene vastaamaan kyselyihin, vaan jumiutuu. Jumiutuminen poistuu vain laitteen uudelleenkäynnistyksellä. On todennäköistä, että ongelmaa saisi lievennettyä muuttamalla ohjelmistoa siten, että laite pystyisi käynnistymään itsestään uudelleen ongelmatilanteen myötä. Tämä ei kuitenkaan poista palvelunestohyökkäyksen mahdollisuutta, koska heikkotehoisen ESP32-alustan toimintaa on yksinkertaista häiritä toistuvien HTTPS-kyselyiden avulla. Tällöin laite ei pysty vastaamaan Zabbix-palvelimeltakaan tuleviin kutsuihin, jolloin monitorointia ei voida toteuttaa. Yksi tapa ratkaista ongelma on esitetty kuvassa 6.2. Tässä ratkaisumallissa mittauslaitteeseen tietoliikennettä rajataan palo-

muurin avulla. Palomuurisäännön avulla rajataan saapuvia pyyntöjä siten, että vain Zabbix-palvelimen IP-osoitteesta saapuvat pyynnöt sallitaan, jolloin ei-sallituista osoitteista saapuvat pyynnöt eivät mene mittauslaitteelle asti.



Kuva 6.2: Yhteyksien rajoittaminen palomuurin avulla

Toinen, monimutkaisempi tapa ratkaista ongelma on esitetty kuvassa 6.3. Tässä ratkaisumallissa mittauslaitteeseen kohdistuvaa kuormitusta vähennetään kuormantasaajan avulla. Kuormantasaajalla on kaksi roolia, jotka ovat TLS-salauksen terminointi ja HTTPS-pyynnössä olevan user-agent -otsikon tarkastaminen. TLS-salauksen terminointi tarkoittaa HTTPS-salauksen purkamista, jolloin kuormantasaajan ja mittauslaitteen välinen tietoliikenne tapahtuu salaamattomana. Salaamattoman HTTP-liikenteen käsittely vie vähemmän resursseja kuin salatun HTTPS-liikenteen käsittely. RFC7231-standardin mukaan user-agent -otsikon pitäisi sisältää tiedot käytettävästä asiakasohjelmasta, mutta merkkijono on vapaasti määritettävissä [37]. Kuormantasaaja mahdollistaa liikenteen sallimisen tai estämisen user-agent -otsikon perusteella, jolloin käytettäessä normaaleista poikkeavia merkkijonoja on mahdollista suodattaa ei-sallittujen lähteiden pyynnöt. Palomuurin tehtävänä on tässä ratkaisumallissa estää kaikki pyynnöt, jotka eivät tule kuormantasaajan kautta.



Kuva 6.3: Yhteyksien rajoittaminen palomuurin ja kuormantasaajan avulla

Jatkokehityksen arviointi

Jatkokehityskohteita pohtiessa on tärkeää arvioida, kuinka paljon muutoksia olemassa olevaan kokonaisuuteen tarvitsee tehdä uusien ominaisuuksien mahdollistamiseksi. Osa lisätoiminnoista voidaan toteuttaa suoraviivaisesti uuden toiminnon vaatimien sensoreiden ja aktuaattoreiden kytkemisellä ja ohjelmakoodin lisäyksillä. Esimerkkinä suoraviivaisesta uuden ominaisuuden lisäämisestä on aiemmin mainitun liiketunnistinsensorin käyttöönotto. Sen sijaan esimerkiksi täysin reaaliaikainen räkkikaapin ovenavauksen valvonta vaatisi sovellustasolla enemmän muutoksia. Kun nykyinen toimintamalli perustuu säännöllisin aikavälein HTTPS-palvelimelta tapahtuviin kyselyihin, reaaliaikainen malli vaatisi oven avaamisen yhteydessä tapahtuvan herätteen lähettämisen esimerkiksi HTTP POST-pyynnön kautta. Tällöin mittauslaitteeseen tulisi olemassa olevan HTTPS-palvelinroolin lisäksi HTTPS-asiakasrooli, eli mittauslaitteen alkuperäinen arkkitehtuuri muuttuisi. Myös vastaanottavaan järjestelmään tulee tehdä tarvittavat muutokset, jotta datan vastaanottaminen on mahdollista.

Jatkokehitystarpeita arvioidessa tulee arvioida myös alustan suorituskyvyn riittävyyttä muutosten jälkeen. Uudet sensorit vaativat usein ohjelmakirjastojen lisäyk-

siä, jolloin ohjelmiston kompleksisuus kasvaa lisäten resurssien tarvetta. Erityisesti heikkosuorituskykyisten IoT-alustojen kohdalla tämä voi aiheuttaa ongelmia, jos laitteen resurssien käyttö on ollut korkeaa jo ennen uusien ominaisuuksien lisäämistä. Tällöin voi tulla kyseeseen alustaratkaisun muuttaminen, kuten ESP32-alustan vaihtaminen Raspberry Pi -tietokoneeseen. Isojen muutosten tekeminen ei ole olemassa olevan järjestelmän jatkokehittämistä, vaan vaatii uuden kehitysprojektin.

7 Yhteenveto

Osana tätä tutkielmaa kehitettiin spiraalimallin mukaisesti IoT-laitteeksi luokiteltava mittauslaite, joka mahdollistaa konesaliolosuhteiden monitoroinnin. Mittauslaitteen tuottamaa dataa hyödynnetään Zabbix-monitorointisovelluksen avulla. Toteutettu kokonaisuus auttaa konesaliympäristön ylläpitäjää havaitsemaan epäsuotuisaksi muuttuneet olosuhteet nopeasti. Alkaneen ongelmatilanteen nopea havainnointi on avaintekijänä vahinkojen minimoimisessa.

IoT-käsitteen moniulotteisuuden vuoksi tutkielman teoriaosuudessa käsiteltiin esineiden Internetin aihepiiriin liittyviä teknologioita ja laitteistoja, joiden avulla voidaan toteuttaa IoT-laitteistojen tekniset mittaus- ja tiedonsiirto-ominaisuudet. Esiteltiin myös referenssimalleja, joihin useiden IoT-ratkaisujen toimintamallit perustuvat. IoT-laitteilla on potentiaalisia käyttömahdollisuuksia useissa käyttökohteissa, joissa ne mahdollistavat kokonaan uusia toimintoja tai auttavat saavuttamaan parempia tuloksia nykyisiin toimintamalleihin verrattuna. On kuitenkin huomioitava, että hyötyjen lisäksi IoT-laitteisiin liittyy uhkakuvia ja ongelmia, joihin on varauduttava.

Konesaliympäristön olosuhteet poikkeavat tavanomaisista toimisto- ja varastotiloista monin tavoin. Konesaleissa sijaitsee tietojenkäsittelyyn käytettäviä laitteistoja, jotka ovat suunniteltu käytettäväksi ympärivuorokautisesti. Konesaliympäristöä käsittelevässä luvussa esiteltiin konesaliympäristön erityispiirteitä ja mahdollisia ongelmakohteita, joiden asianmukaisella monitoroinnilla voidaan välttää ongelmatilanteiden eskaloituminen. Lisäksi esiteltiin ratkaisuja, joiden avulla monitorointia voidaan toteuttaa.

Tutkielman empiirisessä osassa toteutettiin mittauslaite spiraalimallin mukaisesti. Spiraalimalli on tuotekehitysmalli, joka soveltuu tämäntyyppiseen kehitystyöhön. Tuotekehitysmenetelmiä käsittelevässä luvussa esiteltiin myös muita tuotekehitysmenetelmiä, joita voidaan käyttää laitteisto- ja ohjelmistokehitystyössä. Eri malleissa on vahvuutensa ja heikkoutensa, jolloin käytettävän tuotekehitysmallin valinta on tehtävä tapauskohtaisesti.

Mittauslaitteen suunnitteluprosessin alussa määritettiin toiminnot, joihin valmiilla laitteella pitää olla kyvykkyys. Ominaisuusvaatimusten perusteella laittee-

seen valittiin tarvittavat komponentit, jotka mahdollistavat toimintojen toteuttamisen tarkoituksenmukaisella tavalla. Iteraatiokierrosten aikana laitteen kannalta merkityksellisimmät toiminnot toteutettiin ensin. Tällöin laitteen toimintamallia olisi voitu muuttaa ongelmatilanteissa siten, ettei tarpeetonta ohjelmointityötä olisi jouduttu tekemään. Mittauslaitteen valmistuttua se integroitiin Zabbix-monitorointisovellukseen ajastetusti suoritettavan Bash-komentojonon avulla, jolloin Zabbix vastaanottaa säännöllisesti vastaan uutta mittausdataa. Lisäksi Zabbixiin toteutettiin konesalin ylläpitäjälle visuaaliset näkymät, jotka helpottavat ja nopeuttavat mittauslaitteen tuottaman datan tulkintaa sekä toteutettiin hälytyssäännöt, jotka lähettävät konesalin ylläpitäjälle sähköpostiviestin mittaustulosten ylittäessä määritetyt rajaarvot.

Pohdinta ja jatkokehitys -osiossa käsiteltiin mittauslaitteen ominaisuuksien ja datankäsittelyn onnistumista ja toimivuutta toimintokohtaisesti. Mittauslaite onnistui kokonaisuudessaan hyvin ja se täyttää sille asetetut vaatimukset. Myös Zabbixilla toteutetut toiminnot toimivat asianmukaisesti. Pohdittiin myös, kuinka mittauslaitekokonaisuutta voitaisiin jatkokehittää. Tietoturvallisuuden ja laitteen vakaan toimivuuden turvaamisen näkökulmasta tärkein kehityskohde on pääsynrajoituksen toteuttaminen, jolloin ulkopuoliset eivät pääse käsiksi mittausdataan tai toteuttamaan palvelunestohyökkäystä. Lisätoimintojen näkökulmasta tärkeimmät kehityskohteet ovat liiketunnistusominaisuuden lisääminen sekä räkkikaapin ovimonitoroinnin jatkokehitys. Käyttömukavuuden parantamiseksi WLAN-verkkoasetusten määritysten muuttaminen on mahdollistettava Bluetoothin avulla.

Lähteet

[1] 3RD GENERATION PARTNERSHIP PROJECT (3GPP). 3GPP Release 15. URL https://portal.3gpp.org/desktopmodules/Release/ReleaseDetails.aspx?releaseId=190, viitattu 27.3.2022.

[2] 3RD GENERATION PARTNERSHIP PROJECT (3GPP). 3GPP Release 18. URL https://www.3gpp.org/release18, viitattu 27.3.2022.

[3] 3RD GENERATION PARTNERSHIP PROJECT (3GPP). About 3GPP. URL https://www.3gpp.org/about-3gpp, viitattu 27.3.2022.

[4] ABDELRAHMAN, R. B. M., MUSTAFA, A. B. A., JA OSMAN, A. A. A Comparison between IEEE 802.11 a, b, g, n and ac Standards. *IOSR Journal of Computer Engineering (IOSR-JEC) 17*, 5 (2015), 26–29.

[5] AGC NETWORKS LTD. Black Box Alertwerks sensors. URL https://www.blackbox.fi/_AppData/cms/Default%20pages/Solutions/AlertWerks/AlertWerks_Brochure_EN.pdf, viitattu 1.2.2022.

[6] ALSHAMRANI, A., JA BAHATTAB, A. A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. *International Journal of Computer Science Issues (IJCSI) 12*, 1 (2015), 106–111.

[7] ANASTASI, G., BANDELLONI, R., CONTI, M., DELMASTRO, F., GREGORI, E., JA MAINETTO, G. Experimenting an indoor bluetooth-based positioning service. Julkaisusarjassa *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* (Providence, RI, USA, Toukokuu 2003), IEEE, 480–483.

[8] ARDUCAM TECHNOLOGY CO., LTD. Arducam ESP32 UNO user guide. URL https://www.arducam.com/downloads/ESP32_UNO/ArduCAM_ESP32_UNO_DS.pdf, viitattu 26.3.2022.

[9] ARDUINO AG. Arduino UNO introduction. URL https://blog.arduino.cc/2010/09/24/dinner-is-ready/, viitattu 26.3.2022.

[10] ARDUINO AG. Arduino UNO R3 Product Reference manual. URL https://docs.arduino.cc/static/dd40bcbb5f5ac04b6315a92e4e45d0f0/A000066-datasheet.pdf, viitattu 6.2.2022.

[11] ARDUINO AG. Trademark and Copyright. URL https://www.arduino.cc/en/trademark, viitattu 23.1.2022.

[12] ARDUINO AG. What is Arduino? URL https://www.arduino.cc/en/Guide/Introduction, viitattu 23.1.2022.

[13] BINMASOUD, A., JA CHENG, Q. Design of an IoT-based Vehicle State Monitoring System Using Raspberry Pi. Julkaisusarjassa *2019 International Conference on Electrical Engineering Research & Practice (ICEERP)* (Sydney, NSW, Australia, Marraskuu 2019), IEEE, 1–6.

[14] BLUETOOTH SIG, INC. About us. URL https://www.bluetooth.com/about-us/, viitattu 18.12.2021.

[15] BLUETOOTH SIG, INC. Understanding Bluetooth Range. URL https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/, viitattu 18.12.2021.

[16] BLUETOOTH SIG, INC. *Bluetooth Core Specification 5.3*, 2021.

[17] BOEHM, B. W. A spiral model of software development and enhancement. *Computer 21*, 5 (1988), 61–72.

[18] BOSCH SENSORTEC GMBH. BME280 Technical Data. URL https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/#technical, viitattu 31.10.2021.

[19] CISCO SYSTEMS INC. *The Internet of Things Reference Model*. Tekninen raportti 06-04, Cisco Systems Inc, 2014.

[20] COLLIN, J., JA SAARELAINEN, A. *Teollinen internet*. Talentum Oyj, Helsinki, Suomi, 2016.

[21] CONGER, K., FAUSSET, R., JA KOVALESKI, S. F. San Francisco bans facial recognition technology. *The New York Times 14* (2019).

[22] CONNECTIVITY STANDARDS ALLIANCE. CSA Members. URL https://zigbeealliance.org/members/, viitattu 1.1.2022.

[23] CONNECTIVITY STANDARDS ALLIANCE. Zigbee FAQ. URL https://zigbeealliance.org/zigbee-faq/, viitattu 16.1.2022.

[24] DACHYAR, M., ZAGLOEL, T. Y. M., JA SARAGIH, L. R. Knowledge growth and development: internet of things (IoT) research, 2006–2018. *Heliyon 5*, 8 (2019), e02264.

[25] DELL. *Dell EMC PowerEdge R7515 Technical Guide*. Tekninen raportti 12/21, Dell Technologies Inc., 2021.

[26] DIGITA OY. IoT:lla tasaista asumismukavuutta VVO-konsernin kohteissa Espoossa. URL https://www.digita.fi/ajankohtaista/iotlla-tasaista-asumismukavuutta-vvo-konsernin-kohteissa-espoossa/, viitattu 4.12.2021.

[27] DIGITA OY. IoT:n kartta. URL https://www.digita.fi/iotn-kartta/, viitattu 1.1.2022.

[28] DIGITA OY. VVO-konsernista Digitan valtakunnallisen IoT-verkon ensimmäinen asiakas. URL https://www.digita.fi/ajankohtaista/vvo-konsernista-digitan-valtakunnallisen-iot-verkon-ensimmainen-asiakas/, viitattu 4.12.2021.

[29] DRAGINO TECHNOLOGY CO., LTD. Dragino LoRa Shield. URL https://wiki.dragino.com/index.php?title=Lora_Shield, viitattu 26.3.2022.

[30] DÖNMEZ, T. C. M., JA NIGUSSIE, E. Security of lorawan v1. 1 in backward compatibility scenarios. *Procedia computer science 134* (2018), 51–58.

[31] ELECTRONIC INDUSTIES ALLIANCE (EIA). *EIA-310-D*, 1992.

[32] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. ESP32 Devkits. URL https://www.espressif.com/en/products/devkits, viitattu 30.10.2021.

[33] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. ESP32-WROOM-32D Datasheet. URL https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf, viitattu 6.2.2022.

[34] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. Espressif milestones. URL https://www.espressif.com/en/company/about-us/milestones, viitattu 26.3.2022.

[35] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. Espressif Technical Documents. URL https://www.espressif.com/en/support/documents/technical-documents, viitattu 6.2.2022.

[36] EVANS, D. *The Internet of Things - How the Next Evolution of the Internet Is Changing Everything*. Tekninen raportti 04/11, Cisco Internet Business Solutions Group (IBSG), 2011.

[37] FIELDING, R., JA RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, Kesäkuu 2014.

[38] FRIEDHELM LOH GROUP. Rittal CMC III Monitoring system. URL https://www.rittal.com/imf/none/5_1923/, viitattu 1.2.2022.

[39] FUJIFILM CORPORATION. LTO Ultrium Technical Data. URL https://www.fujifilm.com/us/en/business/data-storage/data-storage-media/lto-ultrium-6/specifications, viitattu 28.10.2021.

[40] FUTURE US, INC. GPU Benchmarks and Hierarchy 2021: Graphics Cards Ranked. URL https://www.tomshardware.com/reviews/gpu-hierarchy,4388.html, viitattu 21.11.2021.

[41] GARTNER, INC. IoT Units Installed Base by Category (Millions of Units). URL https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016, viitattu 4.12.2021.

[42] GENG, H. *Data Center Handbook*. John Wiley & Sons, Inc., Palo Alto, CA, USA, 2014.

[43] GHOSH, A., MAEDER, A., BAKER, M., JA CHANDRAMOULI, D. 5G Evolution: A View on 5G Cellular Technology Beyond 3GPP Release 15. *IEEE Access 7* (2019), 127639–127651.

[44] GOZALVEZ, J. Samsung Electronics Sets 5G Speed Record at 7.5 Gb/s [Mobile Radio]. *IEEE Vehicular Technology Magazine 10*, 1 (2015), 12–16.

[45] HAARTSEN, J. The Bluetooth radio system. *IEEE Personal Communications 7*, 1 (2000), 28–36.

[46] HATAKKA, S., VALKEINEN, H., JA HUURINAINEN, V. *Sähkölaitteistoista aiheutuneet tulipalot ja palovaarat Suomessa - esiselvitys*. Tekninen raportti 1/2014, Turvallisuus- ja kemikaalivirasto, 2014.

[47] HEWLETT PACKARD ENTERPRISE. *HPE ProLiant DL380 Gen10 Server - Specifications*. Tekninen raportti 02/22, Hewlett Packard Enterprise Company, 2022.

[48] IEEE STANDARD 802.11. *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, 1997.

[49] IEEE STANDARD 802.11A. *IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band*, 1999.

[50] IEEE STANDARD 802.11AC. *IEEE Standard for Information technology–Telecommunications and information exchange between systems—Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz.*, 2013.

[51] IEEE STANDARD 802.11AH. *IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation*, 2017.

[52] IEEE STANDARD 802.11AX. *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN*, 2021.

[53] IEEE STANDARD 802.11B. *IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control*

(MAC) and Physical Layer (PHY) specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band, 2000.

[54] IEEE STANDARD 802.11G. *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher Data Rate Extension in the 2.4 GHz Band*, 2003.

[55] IEEE STANDARD 802.11N. *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, 2009.

[56] ILOQ OY. iLOQ NFC, Frequently asked questions. URL `https://www.iloq.com/nfc/faq`, viitattu 2.2.2022.

[57] ILOQ OY. iLOQ S50 esite. URL `https://www.iloq.com/wp-content/uploads/2021/11/iloq-s50-2021-eng.pdf`, viitattu 2.2.2022.

[58] INTEL CORPORATION. Thermal Management for Intel Xeon Processors. URL `https://www.intel.com/content/www/us/en/support/articles/000006710/processors/intel-xeon-processors.html`, viitattu 28.10.2021.

[59] JUSAK, J., PRATIKNO, H., JA PUTRA, V. H. Internet of Medical Things for cardiac monitoring: Paving the way to 5G mobile networks. Julkaisusarjassa *2016 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)* (Surabaya, Indonesia, Joulukuu 2016), IEEE, 75–79.

[60] KATAOKA, H., DUOLIKUN, D., ENOKIDO, T., JA TAKIZAWA, M. Power Consumption and Computation Models of a Server with a Multi-core CPU and Experiments. Julkaisusarjassa *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops* (Gwangju, Korea (South), Maaliskuu 2015), IEEE, 217–222.

[61] KIM, T.-H., RAMOS, C., JA MOHAMMED, S. Smart city and IoT. *Future Generation Computer Systems 76* (2017), 159–162.

[62] KOLHE, L., MORE, S., AHER, K., JA THAKARE, A. Server Room Access Control Using (IOT). *IRE Journals 1*, 11 (2018), 47–51.

[63] KOLIAS, C., KAMBOURAKIS, G., STAVROU, A., JA VOAS, J. DDoS in the IoT: Mirai and other botnets. *Computer 50*, 7 (2017), 80–84.

[64] KORHONEN, H., LEPPÄNEN, H., MATTILA, H. T., JA PEKKOLA, V. Otsonoinnin turvallinen käyttö. *Ympäristö ja Terveys-lehti 53*, 1 (2022), 40–45.

[65] KUMAR, M. S., CHANDRA, T. R., KUMAR, D. P., JA MANIKANDAN, M. S. Monitoring moisture of soil using low cost homemade Soil moisture sensor and Arduino UNO. Julkaisusarjassa *2016 3rd international conference on advanced computing and communication systems (ICACCS)* (Coimbatore, India, Tammikuu 2016), IEEE, 1–4.

[66] LECCESE, F. Remote-control system of high efficiency and intelligent street lighting using a ZigBee network of devices and sensors. *IEEE Transactions on Power Delivery 28*, 1 (2012), 21–28.

[67] LIIKENNE- JA VIESTINTÄVIRASTO TRAFICOM. Matkaviestinverkkojen taajuudet ja luvanhaltijat. URL `https://www.traficom.fi/fi/viestinta/viestintaverkot/matkaviestinverkkojen-taajuudet-ja-luvanhaltijat`, viitattu 27.3.2022.

[68] LORA ALLIANCE. About LoRa Alliance. URL `https://lora-alliance.org/about-lora-alliance/`, viitattu 19.12.2021.

[69] LORA ALLIANCE. *LoRaWAN 1.1 Specification*, 2017.

[70] LU, T., LU, X., REMES, M., JA VILJANEN, M. Investigation of air management and energy performance in a data center in Finland: Case study. *Energy and Buildings 43*, 12 (2011), 3360–3372.

[71] MA, Y., RAO, J., HU, W., MENG, X., HAN, X., ZHANG, Y., CHAI, Y., JA LIU, C. An efficient index for massive IOT data in cloud environment. Julkaisusarjassa *Proceedings of the 21st ACM international conference on Information and knowledge management* (New York, USA, Lokakuu 2012), Association for Computing Machinery, 2129–2133.

[72] MARKS, H. Server Installation Considerations. Kirjassa *Server Management*, G. Held, Ed. Auerbach Publications, 2000, ch. 8, ss. 107–114.

[73] MITTAL, Y., VARSHNEY, A., AGGARWAL, P., MATANI, K., JA MITTAL, V. K. Fingerprint biometric based Access Control and Classroom Attendance Management System. Julkaisusarjassa *2015 Annual IEEE India Conference (INDICON)* (New Delhi, India, Joulukuu 2015), IEEE, 1–6.

[74] MOTOCHI, V., BARASA, S., OWOCHE, P., JA WABWOBA, F. The role of virtualization towards green computing and environmental sustainability. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 6*, 6 (2017), 851–858.

[75] MUHAMMED, A. S., JA UCUZ, D. Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users' Perspectives. Julkaisusarjassa *2020 8th International Symposium on Digital Forensics and Security (ISDFS)* (Beirut, Lebanon, Kesäkuu 2020), IEEE, 1–4.

[76] NOKIA OYJ. 5G spectrum bands explained. URL `https://www.nokia.com/networks/insights/spectrum-bands-5g-world/`, viitattu 1.4.2022.

[77] NOKIA OYJ. Nokia achieves 5G speed world record with Turk Telekom. URL `https://www.nokia.com/about-us/news/releases/2021/03/22/nokia-achieves-5g-speed-world-record-with-turk-telekom/`, viitattu 9.4.2022.

[78] OOSTERLINCK, D., BENOIT, D. F., BAECKE, P., JA VAN DE WEGHE, N. Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits. *Applied geography 78* (2017), 55–65.

[79] OXFORD UNIVERSITY PRESS. *The Oxford English Dictionary*. Oxford University Press, Oxford, United Kingdom, 2021.

[80] PATEL, C. AWS IoT solutions design practices. URL `https://www.iot-now.com/2019/06/12/96502-aws-iot-solutions-design-practices/`, viitattu 5.2.2022.

[81] POORNACHANDRAN, P., SREERAM, R., KRISHNAN, M. R., PAL, S., SANKAR, A. U. P., JA ASHOK, A. Internet of Vulnerable Things (IoVT): Detecting Vulnerable SOHO Routers. Julkaisusarjassa *2015 International Conference on Information Technology (ICIT)* (Bhubaneswar, India, Joulukuu 2015), IEEE, 119–123.

[82] RAI, P., JA REHMAN, M. ESP32 Based Smart Surveillance System. Julkaisusarjassa *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (Sukkur, Pakistan, Tammikuu 2019), IEEE, 1–3.

[83] RASPBERRY PI LTD. Raspberry Pi Pico Tech Specs. URL https://www.raspberrypi.com/products/raspberry-pi-pico/specifications/, viitattu 26.3.2022.

[84] RASPBERRY PI LTD. Raspberry Pi Processors. URL https://www.raspberrypi.com/documentation/computers/processors.html, viitattu 26.3.2022.

[85] RASPBERRY PI LTD. Raspberry Pi Schematics and Mechanical Drawings. URL https://www.raspberrypi.com/documentation/computers/raspberry-pi.html, viitattu 6.2.2022.

[86] RASTOGI, V. Software Development Life Cycle Models-Comparison, Consequences. *International Journal of Computer Science and Information Technologies 6*, 1 (2015), 168–172.

[87] RODRIGUES, J. J., SEGUNDO, D. B. D. R., JUNQUEIRA, H. A., SABINO, M. H., PRINCE, R. M., AL-MUHTADI, J., JA DE ALBUQUERQUE, V. H. C. Enabling technologies for the internet of health things. *IEEE Access 6* (2018), 13129–13141.

[88] SANASTOKESKUS RY. Määritelmä, Esineiden Internet. URL http://www.tsk.fi/tsk/termitalkoot/hakemistot-267.html?page=get_id&id=ID335&vocabulary_code=TSKTT, viitattu 27.11.2021.

[89] SEHRAWAT, D., JA GILL, N. S. Smart sensors: Analysis of different types of IoT sensors. Julkaisusarjassa *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)* (Tirunelveli, India, Huhtikuu 2019), IEEE, 523–528.

[90] SHENZHEN KEYES ROBOT CO. LTD. Keyestudio KS0052 Technical Data. URL https://www.mantech.co.za/datasheets/products/KS-SENSOR_MODULES-1_KEYES.pdf, viitattu 14.2.2022.

[91] SIVANATHAN, A., GHARAKHEILI, H. H., JA SIVARAMAN, V. Can we classify an iot device using tcp port scan? Julkaisusarjassa *2018 IEEE International*

Conference on Information and Automation for Sustainability (ICIAfS) (Colombo, Sri Lanka, Joulukuu 2018), IEEE, 1–4.

[92] SUOMEN KUNTALIITTO RY. Kustannusrakenne. URL https://www.kuntaliitto.fi/tietotuotteet-ja-palvelut/analyysit-ja-tietoaineistot/kustannusrakenne, viitattu 4.12.2021.

[93] TAIVALSAARI, A., JA MIKKONEN, T. On the development of IoT systems. Julkaisusarjassa *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)* (Barcelona, Spain, Huhtikuu 2018), IEEE, 13–19.

[94] TERVEYDEN JA HYVINVOINNIN LAITOS. Koronavilkku, usein kysyttyä. URL https://koronavilkku.fi/ukk/, viitattu 19.12.2021.

[95] TEXAS INSTRUMENTS INCORPORATED. *What's New in Zigbee 3.0*. Tekninen raportti SWRA615A, Texas Instruments Incorporated, 2019.

[96] THE MATHWORKS, INC. ThingSpeak Github. URL https://github.com/iobridge/thingspeak, viitattu 2.4.2022.

[97] THE MATHWORKS, INC. ThingSpeak Home License. URL https://thingspeak.com/prices/thingspeak_home, viitattu 2.4.2022.

[98] THE MATHWORKS, INC. ThingSpeak Overview. URL https://se.mathworks.com/products/thingspeak/system-requirements.html, viitattu 2.4.2022.

[99] THE MATHWORKS, INC. ThingSpeak Product description. URL https://se.mathworks.com/help/thingspeak/product-description.html, viitattu 2.4.2022.

[100] THE MATHWORKS, INC. ThingSpeak Standard License. URL https://thingspeak.com/prices/thingspeak_standard, viitattu 2.4.2022.

[101] THE THINGS INDUSTRIES B.V. LoRaWAN Frequency Plans by Country. URL https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country/, viitattu 1.1.2022.

[102] THE THINGS INDUSTRIES B.V. LoRaWAN Limitations. URL https://www.thethingsnetwork.org/docs/lorawan/limitations/, viitattu 1.1.2022.

[103] VISHAY INTERTECHNOLOGY INC. TEMT6000 Ambient Light Sensor Technical Data. URL https://www.sparkfun.com/datasheets/Sensors/Imaging/TEMT6000.pdf, viitattu 14.2.2022.

[104] VLADISHEV, A. Open Source Enterprise Monitoring with Zabbix. URL http://www.netways.de/uploads/media/Alexei_Vladishev_Open_Source_Monitoring_with_Zabbix.pdf, viitattu 21.11.2021.

[105] VOJKOVIC, G., MILENKOVIC, M., JA KATULIC, T. IoT and Smart Home Data Breach Risks from the Perspective of Data Protection and Information Security Law. *Business Systems Research: International journal of the Society for Advancing Innovation and Research in Economy 11*, 3 (2020), 167–185.

[106] WAN, F., SWENSON, D., HILLSTROM, M., POMMERENKE, D., JA STAYER, C. The effect of humidity on static electricity induced reliability issues of ICT equipment in data centers – Motivation and setup of the study. *ASHRAE Transactions 119*, 2 (2013), 341–357.

[107] WI-FI ALLIANCE. Wi-Fi Certification. URL https://www.wi-fi.org/certification, viitattu 11.12.2021.

[108] WIEGERS, K., JA BEATTY, J. *Software requirements*. Microsoft Press, Redmond, WA, USA, 2013.

[109] ZABBIX LLC. About Zabbix LLC. URL https://www.zabbix.com/about, viitattu 21.11.2021.

[110] ZABBIX LLC. Zabbix Agent. URL https://www.zabbix.com/zabbix_agent, viitattu 10.4.2022.

[111] ZABBIX LLC. Zabbix License. URL https://www.zabbix.com/license, viitattu 21.11.2021.

[112] ZABBIX LLC. Zabbix overview. URL https://www.zabbix.com/documentation/current/en/manual/introduction/overview, viitattu 2.4.2022.

[113] ZABBIX LLC. Zabbix Proxy. URL https://www.zabbix.com/documentation/current/en/manual/concepts/proxy, viitattu 10.4.2022.

[114] ZABBIX LLC. Zabbix Requirements. URL https://www.zabbix.com/documentation/current/en/manual/installation/requirements, viitattu 2.4.2022.

[115] ZABBIX LLC. Zabbix Sender. URL https://www.zabbix.com/documentation/current/en/manual/concepts/sender, viitattu 10.4.2022.

[116] ZABBIX LLC. Zabbix Web interface. URL https://www.zabbix.com/documentation/current/en/manual/installation/frontend, viitattu 2.4.2022.

[117] ZHENGZHOU WINSEN ELECTRONICS TECHNOLOGY CO., LTD. MQ-2 Flammable Gas Sensor Manual. URL https://www.winsen-sensor.com/d/files/MQ-2.pdf, viitattu 31.10.2021.

[118] ZIGBEE ALLIANCE. Technical Presentation: ZigBee, The Standard for the IoT. URL https://zigbeealliance.org/wp-content/uploads/2019/12/Zigbee-Technical-2019.pptx, viitattu 1.1.2022.

[119] ZIGBEE STANDARDS ORGANIZATION. *ZigBee Specification Version 1.0 (053474r06)*, 2005.

[120] ZOHARI, M. H., BALA, V., JA ABD GHAFAR, A. S. Server monitoring based on IoT using ThingSpeak. *Journal of Electrical Power and Electronic Systems 1*, 2 (2019).

A Mittauslaitteen lähdekoodi

```
1  // Kirjasto WLAN varten
2  #include <WiFi.h>
3  // Kirjasto HTTPS-palvelua varten
4  #include <ESPWebServerSecure.hpp>
5
6  // Tiedostot, jotka sisaltavat tiedot sertifikaatista ja avaimesta
7  #include "serti.h"
8  #include "avain.h"
9
10 // Wlan-verkon SSID ja salasana
11 const char* ssid = "VERKON_SSID_TAHAN";
12 const char* salasana = "WLAN_SALASANA_TAHAN";
13
14 // Maaritetaan HTTPS-palvelu kayttamaan TCP-porttia 443
15 ESPWebServerSecure server(443);
16
17 // Kirjasto anturille BME280
18 #include <Adafruit_BME280.h>
19
20 // Asetetaan BME280-anturi kayttamaan I2C-vaylaa
21 Adafruit_BME280 bme;
22
23 // MQ-2-sensorin analoginen ulostulo on kytketty pinniin gpio34
24 const int mq2pinni = 34;
25
26 // Magneettikytkin on kytketty pinniin gpio35
27 const int ovipinni = 35;
28
29 // ovitila = muuttuja, johon tallennetaan ovipinnin tilatieto (HIGH tai LOW)
30 // ovipalautus = muuttuja, johon tallennetaan oven tilatieto ('auki' tai 'kiinni')
31 int ovitila;
32 String ovipalautus;
33
34 void setup(void) {
35   // Asetetaan sarjaportin siirtonopeudeksi 9600 bittia sekunnissa
36   Serial.begin(9600);
37   delay(500);
38
39   // Otetaan kayttoon magneettikytkimen (ovipinnin) sisainen ylosvetovastus
40   pinMode(ovipinni, INPUT_PULLUP);
41
42   // BME280: Aloitetaan I2C-liikenne
43   bme.begin(0x76);
44
45   Serial.print("Yhdistetaan ");
46   Serial.println(ssid);
```

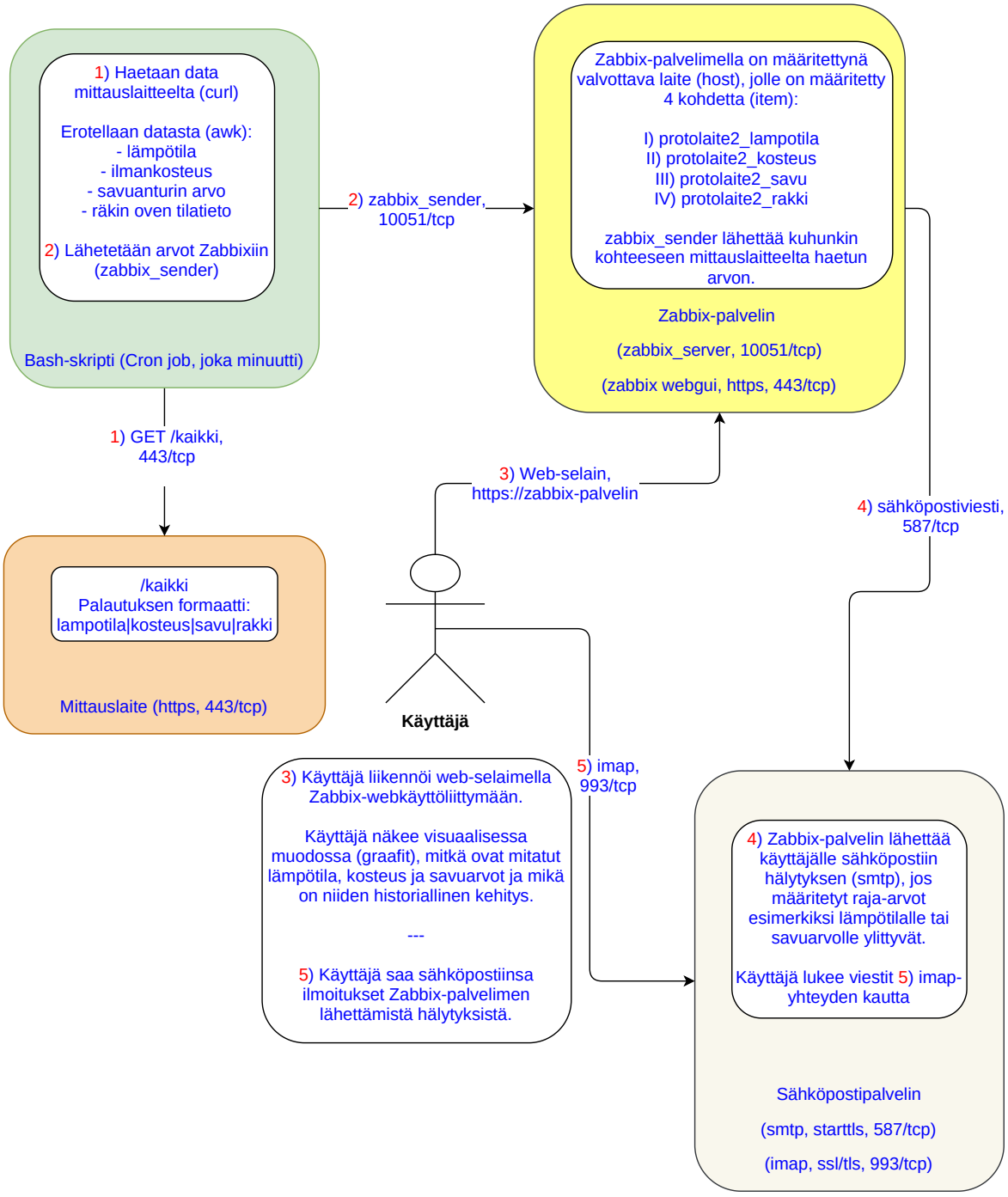
```
47
48    // Yhdista WLAN-verkkoon
49    WiFi.begin(ssid, salasana);
50
51    // Odotetaan, etta WLAN-yhteys on muodostunut ja tulostetaan IP-osoite.
52    while (WiFi.status() != WL_CONNECTED) {
53    delay(1000);
54    Serial.print(".");
55    }
56    Serial.println("");
57    Serial.print("Yhdistetty ");
58    Serial.println(ssid);
59    Serial.print("IP: ");  Serial.println(WiFi.localIP());
60
61
62    // Asetetaan HTTPS-sertifikaatti
63    // 'avain' ja 'avain_pituus' loytyy tiedostosta avain.h
64    // 'serti' ja 'serti_pituus' loytyy tiedostosta serti.h
65    server.setServerKeyAndCert(
66      avain,          // privaattiavain DER-muodossa, byte array
67      avain_pituus,   // pituus: privaattiavain DER-muodossa, byte array
68      serti,          // sertifikaatti DER-muodossa, byte array
69      serti_pituus    // pituus: sertifikaatti DER-muodossa, byte array
70    );
71
72    // Maaritetaan polut (/ ja /kaikki) ja niiden toiminnot (handle_Root ja handle_Kaikki),
          joita voidaan kayttaa HTTPS-kutsuissa.
73    server.on("/", handle_Root);
74    server.on("/kaikki", handle_Kaikki);
75
76    // Jos kysytaan polkua, jota ei ole olemassa, suoritetaan toiminto (handle_404)
77    server.onNotFound(handle_404);
78
79    // Kaynnistetaan HTTPS-palvelu
80    server.begin();
81    Serial.println("HTTPS palvelin kaynnissa.");
82  }
83
84
85  // Palauttaa http-requestille / laitteen "nimen"
86  void handle_Root() {
87    server.send(200, "text/html", "protolaite2");
88    Serial.println("protolaite2");
89  }
90
91  // Palauttaa http-requestille /kaikki
92  // kaikki arvot muodossa lampotila|kosteus|savu|ovi
93  void handle_Kaikki() {
94    // Luetaan 'lampotila' muuttujaan lampotila BME280-sensorilta
95    float lampotila = bme.readTemperature();
96    // Luetaan 'kosteus' muuttujaan kosteus BME280-sensorilta
97    float kosteus = bme.readHumidity();
98    // Luetaan 'savuarvo' muuttujaan mq2pinni:sta arvo
```

```
99    int savuarvo = analogRead(mq2pinni);
100   // Luetaan 'ovitila' muuttujaan ovipinnin tila
101   ovitila = digitalRead(ovipinni);
102
103   // Jos ovipinni on HIGH, ovi on kiinni. Jos LOW, ovi on auki.
104   // Asetetaan 'ovipalautus' muuttujaan joko 'kiinni' tai 'auki'
105   if (ovitila == HIGH) {
106     ovipalautus = "kiinni";
107   } else {
108     ovipalautus = "auki";
109   }
110
111   server.send(200, "text/html", String(lampotila) + "|" + String(kosteus) + "|" + String(
          savuarvo) + "|" + String(ovipalautus));
112   Serial.println("lampotila|kosteus|savu|ovi: " + String(lampotila) + "|" + String(kosteus)
           + "|" + String(savuarvo) + "|" + String(ovipalautus));
113 }
114
115 // Jos pyydettya sivua ei loydy, palautetaan '404' ja http status code 404
116 void handle_404(){
117   server.send(404, "text/html", "404");
118   Serial.println("err_404");
119 }
120
121 // Loopin sisalla kasitellaan asiakkaan HTTPS-pyynnot
122 void loop(void) {
123   server.handleClient();
124 }
```

B Tiedonsiirto mittauslaitteelta Zabbixiin (Bash)

```
#!/bin/bash
# J. Juntunen 10/2021
# Skripti, joka hakee datan mittauslaitteelta ja lahettaa
# eteenpain Zabbixiin. Ajastus crontab:ssa, ajetaan kerran minuutissa.

# Laitteen (dns-)nimi
LAITE="protolaite2"

# Oletusarvot lampotilalle, kosteudelle ja savulle
# Jos curl epaonnistuu tietojen hakemisessa, nama lahetetaan Zabbixiin
LAMPOTILA=0
KOSTEUS=0
SAVU=0
OVI="ERROR"

# 1a) Haetaan tiedot mittalaitteesta curl:n avulla. Parametreina:
# -s (silent)
# -k (sallii "ei-turvallisen" https-yhteyden)
# -m 15 (odottaa palvelimelta vastausta korkeintaan 15 sekuntia)
# Data saapuu muodossa lampotila|kosteus|savu|ovi
# Esimerkki datasta: 28.01|28.38|374|kiinni
DATA=`curl -s -k -m 15 https://$LAITE/kaikki`

# 1b) Otetaan talteen curlin exit code
CURLOUT=$?

# 2) Jos CURLOUT on 0, curl on hakenut tiedot onnistuneesti.
# Erotellaan datasta lampotila, kosteus, savu-arvo ja oven tilatieto
if [[ "$CURLOUT" -eq 0 ]]; then
LAMPOTILA=`echo $DATA | awk -F'|' '{print $1}'`
KOSTEUS=`echo $DATA | awk -F'|' '{print $2}'`
SAVU=`echo $DATA | awk -F'|' '{print $3}'`
OVI=`echo $DATA | awk -F'|' '{print $4}'`
fi

# 3) Lahetetaan tiedot zabbixiin.
# Zabbixiin on luotu "Zabbix server" -hostille 4 itemia:
# <laite>_lampotila, <laite>_kosteus, <laite>_savu ja <laite>_ovi
PALVELIN="Zabbix server"
zabbix_sender -z 127.0.0.1 -p 10051 -s "$PALVELIN" -k "$LAITE"_lampotila -o "$LAMPOTILA"
zabbix_sender -z 127.0.0.1 -p 10051 -s "$PALVELIN" -k "$LAITE"_kosteus -o "$KOSTEUS"
zabbix_sender -z 127.0.0.1 -p 10051 -s "$PALVELIN" -k "$LAITE"_savu -o "$SAVU"
zabbix_sender -z 127.0.0.1 -p 10051 -s "$PALVELIN" -k "$LAITE"_ovi -o "$OVI"
```

C Järjestelmän arkkitehtuurikuvaus



D Mittauslaitteen kytkentäkaavio