

Joona Prehti

# ETÄLUETTAVAN LÄMPÖTILAN MITTAUS- JA TILASTOINTIJÄRJESTELMÄN OPTIMOINTI

Informaatioteknologian ja viestinnän tiedekunta

Kandidaatintyö

Lokakuu 2020

# TIIVISTELMÄ

Joona Prehti: Etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän optimointi  
Kandidaatintyö  
Tampereen yliopisto  
Tietotekniikka  
Lokakuu 2020

---

Etäluettava lämpötilan mittaus- ja tilastointijärjestelmä on järjestelmä, joka mittaa ja dokumentoi jotain tiettyä arvoa. Etäluettavuus mahdollistaa pääsyn internetyhteydellä näihin arvoihin ja arvojen historiatietoon mistä tahansa. Tässä työssä vastataan kysymykseen ”miten optimoida etäluettavaa lämpötilan mittaus- ja tilastointijärjestelmää”. Järjestelmää optimoidaan tietoturvallisuuden ja toimintavarmuuden (reliability) näkökulmasta. Toimintavarmuudella tarkoitetaan järjestelmän kykyä suorittaa vaadittu toiminta tietyissä olosuhteissa ja tietyinä ajanhetkenä. Tietoturvalta tarkoitetaan prosesseja ja työkaluja, joita käytetään sensitiivisen datan suojaamiseen tarkastelulta, muuttamiselta ja tuhoamiselta ilman tarvittavia oikeuksia.

Työssä tarkastellaan millainen on etäluettavan mittaus- ja tilastointijärjestelmän tyypillinen arkkitehtuuri, jotta kandidaatintyössä toteutettu järjestelmä olisi helpompi ymmärtää. Lisäksi tarkastellaan millaiset laitteet, käyttöjärjestelmät ja web-sovelluskehikset sopisivat järjestelmän toteuttamiseen. Työn tuloksena näytetään yksi mahdollinen toteutus järjestelmälle, joka on optimoitu toimintavarmuuden ja tietoturvallisuuden näkökulmasta. Työssä toteutetun järjestelmän pohjana käytettiin Raspberry Pi 3 -mikrotietokonetta, Raspberry Pi OS -käyttöjärjestelmää, Node.js-web-sovelluskehystä, sekä perinteisiä web-teknologioita, kuten HTTP ja CSS.

Avainsanat: Optimointi, lämpötilan mittaus- ja tilastointijärjestelmä, etäluettava, toimintavarmuus, tietoturvallisuus, avoin lähdekoodi, Raspberry Pi 3, ohjelmallinen lämpötilanmittaus

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

# SISÄLLYSLUETTELO

1	Johdanto . . . . .	1
2	Etäluettavan mittaus- ja tilastointijärjestelmän arkkitehtuuri . . . . .	2
2.1	Datanprosessointiyksikkö pilvessä . . . . .	2
2.2	Datanprosessointiyksikkö paikallisessa ympäristössä . . . . .	4
3	Toteutettavan järjestelmän vaatimukset . . . . .	6
3.1	Toiminnalliset vaatimukset . . . . .	6
3.2	Toimintavarmuus . . . . .	6
3.2.1	Virransaannin varmistamisen ongelmat . . . . .	6
3.2.2	Verkkoyhteyden katkeaminen . . . . .	8
3.2.3	Tallennusmuistin korruptoituminen . . . . .	8
3.2.4	Oma verkkotunnus ja staattinen IP-osoite . . . . .	10
3.3	Tietoturvallisuus . . . . .	11
3.3.1	Järjestelmän vahvistaminen ja piilottaminen . . . . .	11
3.3.2	Web-palveluihin kohdistuvia hyökkäyksiä . . . . .	12
4	Järjestelmän toteutusvaihtoehdot . . . . .	15
4.1	Laitteiston valinta . . . . .	15
4.2	Käyttöjärjestelmän valinta . . . . .	17
4.3	Web-sovelluskehityksen valinta . . . . .	17
5	Toteutettu järjestelmä . . . . .	19
5.1	Järjestelmän kuvaus ja toiminnallisten vaatimusten täytyminen . . . . .	19
5.2	Toimintavarmuuden ja tietoturvallisuuden huomioiminen . . . . .	22
6	Yhteenveto . . . . .	25
	Lähteet . . . . .	26

## LYHENTEET JA MERKINNÄT

IoT Järjestelmiä, jotka perustuvat teknisten laitteiden suorittamaan automaattiseen tiedonsiirtoon sekä kyseisten laitteiden etäseurantaan ja -ohjaukseen internet-verkon kautta.

UPS Uninterruptible Power Supply

# 1 JOHDANTO

Yksityiset kuluttajat ja organisaatiot saattavat tarvita erilaisten asioiden tai paikkojen lämpötilatietoa etänä monista eri syistä. Esimerkiksi kuluttaja voi haluta pitää silmällä kesämökkkinsä tai muun kiinteistön lämpötilaa, jos lämpötilan liiallinen laskeminen tai nouseminen voi aiheuttaa vahinkoa kiinteistölle. Lämpötilatietoa voidaan tarvita useissa muissakin tilanteissa, esimerkiksi on tutkittu miten toteuttaa reaaliaikainen lämpötilan mittaus- ja tilastointijärjestelmä ampiaispesiin. Lämpötiladatan saaminen mahdollistaa ampiaisyhdyksuntien tarkemman hoitamisen. [32] Automaattisia lämpötilan mittaus- ja tilastointijärjestelmiä on toteutettu paljon myös terveysteknologian piirissä, koska ruumiinlämpö on yksi tärkeimmistä ihmiseen liittyvistä suureista. Esimerkiksi Mansor et al. [36] ovat julkaisussaan tutkineet, miten lääkärit kykenevät etäyhteyden välityksellä lukemaan potilaiden ruumiinlämmön. Tällainen teknologia voi vähentää sairaalakuluja ja turhaa sairaalassa odottelua, jos yhä enemmän lääkärikäynneistä kyetään tekemään etänä. [35] Lisäksi on kehitetty etäluettavia IoT-järjestelmiä mittaamaan urbaanien ympäristöjen ilmanlaatua ja lämpötilaa [48]. Nämä esimerkit ovat kuitenkin vain pintaraapaisu kaikkiin mahdollisiin käyttökohteisiin nähden.

Tässä työssä tutkitaan, miten optimoida etäluettavaa lämpötilan mittaus- ja tilastointijärjestelmää. Järjestelmää optimoidaan tietoturvallisuuden ja toimintavarmuuden (reliability) näkökulmasta ja toteutuksessa käytetään ainoastaan avoimen lähdekoodin komponentteja. Toteutetun järjestelmän kautta lämpötilatietoja tulee pystyä tarkastelemaan sekä reaaliaikaisesti että menneisyydessä. Työssä myös toteutetaan vaatimukset täyttävä järjestelmä, jonka käyttökohde on jääkaapin ja pakastimen lämpötilan seuraaminen. Toteutettavasta järjestelmästä tulee kuitenkin yleiskäyttöinen ja sen tulisi soveltua myös muiden kohteiden lämpötilan mittaamiseen ja tilastointiin.

Toisessa luvussa tarkastellaan aiempaan tutkimukseen nojautuen, mitä etäluettavalla lämpötilan mittaus- ja tilastointijärjestelmällä tarkalleen tarkoitetaan ja millainen arkkitehtuuri niissä tyypillisesti on. Kolmannessa luvussa tarkennetaan järjestelmän toiminnallisia sekä ei-toiminnallisia vaatimuksia, toimintavarmuutta ja tietoturvallisuutta. Neljännessä luvussa perehdytään siihen kuinka hyvin erilaiset laitteet, käyttöjärjestelmät ja sovel-luskehukset sopivat vaatimukset täyttävän järjestelmän toteuttamiseen. Viidennessä luvussa käydään läpi aiempien lukujen teoriaan nojautuen sitä, miten vaatimukset täyttävä järjestelmä toteutetaan ja mihin tuloksiin tutkimuksessa päästiin. Lopuksi kuudennessa luvussa yhteenveto, joka kokoaa kandidaatintyön.

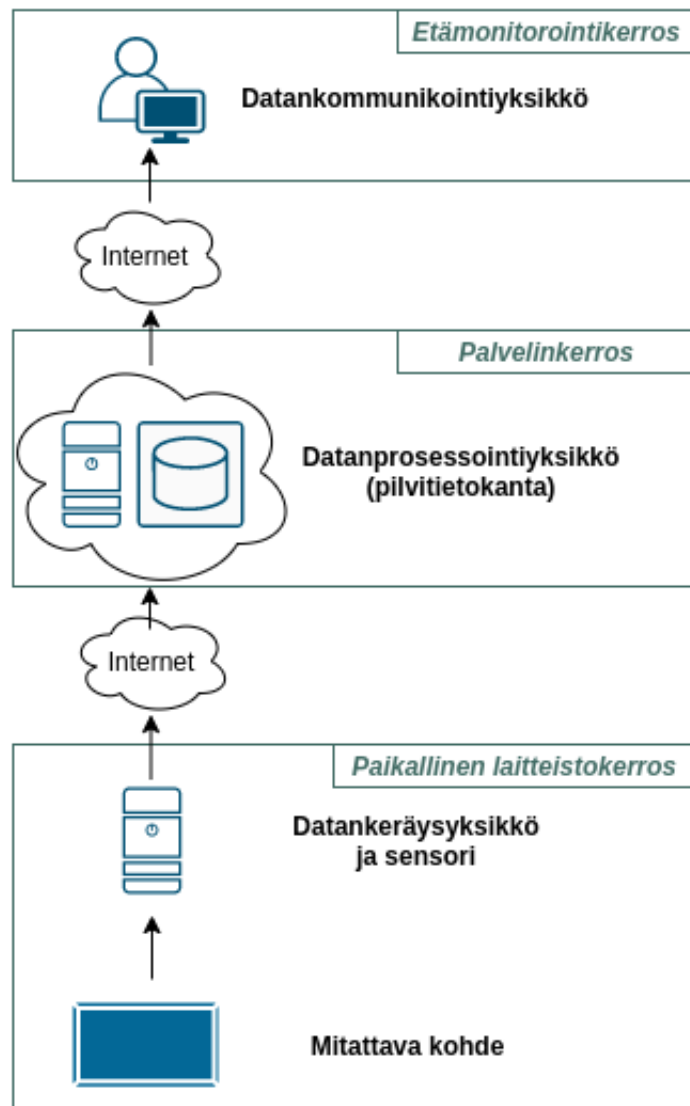
## 2 ETÄLUETTAVAN MITTAUS- JA TILASTOINTIJÄRJESTELMÄN ARKKITEHTUURI

Tässä luvussa tarkastellaan, mikä etäluettava mittaus- ja tilastointijärjestelmä tarkalleen on ja millainen sen tyypillinen arkkitehtuuri on. Etäluettava lämpötilan mittaus- ja tilastointijärjestelmä on laaja käsite. Etäluettavalla järjestelmällä tarkoitetaan järjestelmää, jonka arvoja kyetään lukemaan internetyhteyden välityksellä mistä vain [35]. Etäluettavuus siis mahdollistaa arvojen seuraamisen myös sellaisissa tilanteissa, joissa arvoja tarvitseva henkilö ei itse pääse paikalle niitä lukemaan.

Mittaus- ja tilastointijärjestelmä (monitoring system) taas on automaattinen järjestelmä, joka samanaikaisesti ja jatkuvasti mittaa ja dokumentoi yhtä tai useampaa fyysistä parametria, kuten lämpötilaa tai suhteellista kosteutta [62]. Tässä tutkielmassa keskitytään lämpötilan mittaamiseen ja muiden arvojen mittaamista käsitellään vain lyhyesti. Mittaamisella tarkoitetaan tapahtumaa, jossa parametri, kuten lämpötila, luetaan järjestelmään siten, että siihen päästään ohjelmallisesti käsiksi. Dokumentointi taas tarkoittaa arvon tallentamista siten, että arvoon pääsee käsiksi myös tulevaisuudessa ja silloinkin jos järjestelmä käynnistetään uudelleen. Dokumentoinnin toteuttamiseksi tieto tulee tallentaa johonkin pysyvään tietovarastoon, kuten relaatiotietokantaan tai tiedostoon. Pysyvä tietovarasto on tietovarasto, jossa tieto säilyy pidempään, kuin tietovarastoa käyttävän ohjelman suoritus. [8]

### 2.1 Datanprosessointiyksikkö pilvessä

Etäluettava lämpötilan mittaus- ja tilastointijärjestelmä kyetään toteuttamaan monella erilaisella arkkitehtuurilla. Mansor et al. [36] esittävät tutkimuksessaan tyypillisen arkkitehtuurin kyseisen kaltaisille järjestelmille. Järjestelmän arkkitehtuuri koostuu kolmesta osaluueesta: datankeräisyksiköstä (data acquirement unit), datanprosessointiyksiköstä (database system) ja datankommunikointiyksiköstä (data communication unit). [36] Kuva 1 on arkkitehtuurikaavio kyseisestä järjestelmästä.



*Kuva 1. Arkkitehtuurikaavio järjestelmästä, jossa datanprosessointiyksikkö sijaitsee pilvessä [36].*

Datankeräisyysyksikkö koostuu lämpötilasensorista ja laitteesta, joka kykenee kommunikoi-  
maan internetyhteydellä datanprosessointiyksikön kanssa. Datankeräisyysyksikkö ja läm-  
pötilasensori siis pitää asettaa paikkaan, jonka lämpötila halutaan etälukea. [36]

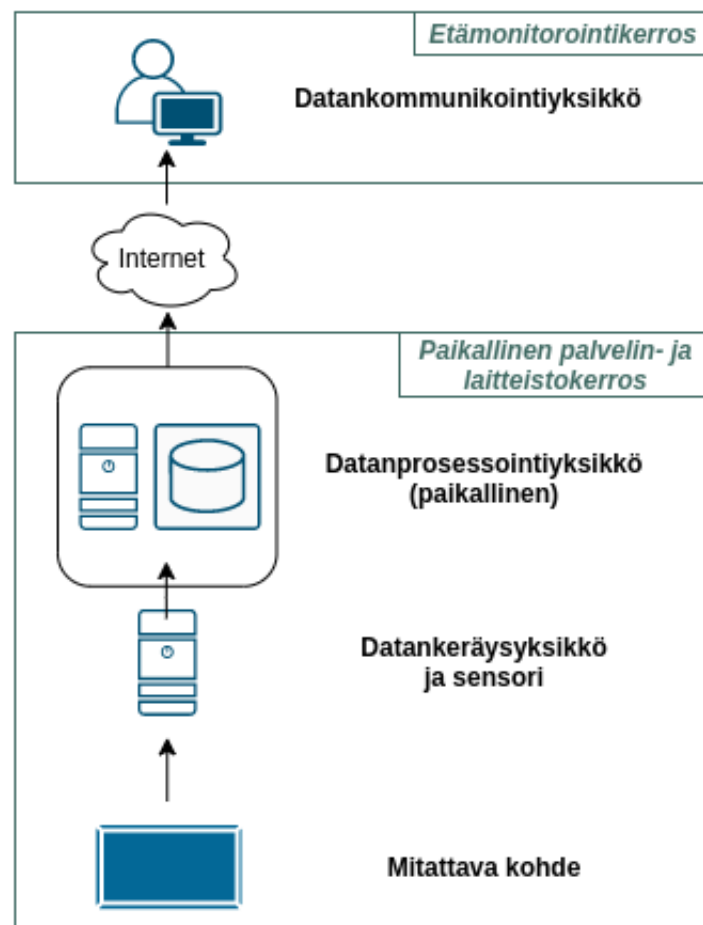
Datanprosessointiyksikkö on pilvessä sijaitseva yksikkö, johon datankeräisyysyksikkö kyke-  
nee tallentamaan lämpötilatietoa pysyvästi ja josta datankommunikaatiyksikkö kykenee  
sitä lukemaan. Yhtenä vaihtoehtona toteuttaa datanprosessointiyksikkö on toteuttaa se  
pilvessä sijaitsevana tietokantana, johon mitattavaa ja tilastoitavaa dataa tallennetaan.  
[36] [32, s. 4]

Datankommunikaatiyksikkö on järjestelmä, jonka kautta järjestelmän käyttäjä kykenee  
etälukemaan tilastoitua dataa. Se voidaan toteuttaa esimerkiksi tietokoneella ja web-  
sivulla, joiden avulla datankeräisyysyksikön tilastoima lämpötiladata kyetään näyttämään

käyttäjälle. Datankommunikointiyksikkö voidaan toteuttaa myös muilla tavoin, kuten työpöytäsovelluksella. Yksikön vaatimuksena on vain valmius lukea lämpötiladataa datanprosessointiyksiköstä ja näyttää sitä käyttäjälle. Data voidaan näyttää käyttäjälle käyttäen erilaisia visualisointitoimintoja, kuten graafeja. Tietoja pääsee tarkastelemaan esimerkiksi sisäänkirjautumalla järjestelmän ylläpitäjän asettamilla tunnuksilla [36].

## 2.2 Datanprosessointiyksikkö paikallisessa ympäristössä

Etäluettava lämpötilan mittaus- ja tilastointijärjestelmä voidaan toteuttaa luvussa 2.1 kuvatulla pilvessä sijaitsevalla datanprosessointiyksiköllä. Fang et al. [16] esittävät hieman erilaisen tavan suunnitella ja toteuttaa etäluettava mittaus- ja tilastointijärjestelmä (remote monitoring system) [16]. Julkaisussa esitetty arkkitehtuuri on muuten sama kuin luvussa 2.1, mutta tässä datankeräisyysyksikköä kutsutaan paikalliseksi laitteistokerrokseksi (Locale Equipments Layer), datanprosessointiyksikköä kutsutaan sulautetuksi palvelinkerrokseksi (Embedded Web Server Layer) ja datankommunikointiyksikköä kutsutaan web-pohjaiseksi etämonitorointikerrokseksi (Web-based Remote Monitoring Layer). Toinen mainittava ero on, että sulautettu palvelinkerros ja paikallinen laitteistokerros sijaitsevat fyysisesti samassa ympäristössä mittauksen kohteen kanssa. Kuva 2 on arkkitehtuurikaavio kyseisestä järjestelmästä. Arkkitehtuurin etuna on, että siinä ei tarvitse luoda tietokantaa ja palvelinta pilveen mitattavien arvojen tallentamiseksi ja jakamiseksi, vaan arvot voidaan tallentaa ja jakaa sulautetussa palvelinkerroksessa paikallisesti. [16] Tämä luonnollisesti pienentää kuluja, koska toteutuksessa ei tarvitse kolmannen osapuolen maksullisia pilvipalveluita. Arkkitehtuurin eduksi voidaan sanoa myös se, että dataa ei tarvitse antaa kolmannen osapuolen vastuulle, kuten pilvipalvelutarjoajille.



*Kuva 2. Arkkitehtuurikaavio järjestelmästä, jossa datanprosessointiyksikkö sijaitsee paikallisessa ympäristössä mittauslaitteiston kanssa [16].*

Etäluettavien mittaus- ja tilastointijärjestelmien arkkitehtuureista on olemassa lukuisia muitakin tutkimuksia, mutta kaikki niistä edustavat kutakuinkin jompaakumpaa aiemmin esitellyistä arkkitehtuureista. Esimerkiksi Zhang et al. [64] esittävät järjestelmälle samantyyppistä arkkitehtuuria kuin Fang et al. [16] omassa julkaisussaan. Lisäksi Kviesis ja Zaccagnini [32] julkaisemassa tutkimuksessa esitetään erilaisia arkkitehtuureja yleiskäyttöisille etäluettaville mittaus- ja tilastointijärjestelmille. Tämänkin tutkimuksen esittelemistä arkkitehtuureista löytyy vastaavat arkkitehtuurit yllä kuvatuille kahdelle arkkitehtuurille.

## **3 TOTEUTETTAVAN JÄRJESTELMÄN VAATIMUKSET**

Tässä luvussa tarkennetaan järjestelmän toiminnallisia ja ei-toiminnallisia vaatimuksia. Järjestelmällä on kaksi ei-toiminnallista vaatimusta, tietoturvallisuus ja toimintavarmuus (reliability).

### **3.1 Toiminnalliset vaatimukset**

Tutkimuksessa halutaan selvittää, miten optimoida etäluettavaa lämpötilan mittaus- ja tilastointijärjestelmää toimintavarmuuden ja tietoturvallisuuden näkökulmista. Työssä myös näytetään yksi toteutusvaihtoehto järjestelmälle, joka on optimoitu näiden vaatimusten näkökulmasta. Tässä luvussa tarkastellaan toteutettavan järjestelmän toiminnallisia vaatimuksia. Työssä toteutettavan järjestelmän tarkoitus ja käyttökohde on mitata ja tilastoida asuinrakennuksessa sijaitsevan pakastimen lämpötilaa. Järjestelmän tulee myös mahdollistaa tietoturallinen datan tallentaminen ja tarkastelu etänä. Datasta pitää pystyä näkemään tämänhetkinen lämpötila ja lämpötilan historiadataa. Lisäksi järjestelmän tulee olla yleiskäyttöinen, niin että sitä pystyy käyttämään mahdollisimman pienillä muutoksilla myös muiden kohteiden lämpötilan mittaamiseen ja tilastointiin. Järjestelmän toiminnalliset vaatimukset saavutetaan toteuttamalla järjestelmä mukaillen jompaa kumpaa luvussa 2 kuvatuista arkkitehtuureista. Perustelut arkkitehtuurin valinnalle kerrotaan luvussa 5.

### **3.2 Toimintavarmuus**

Järjestelmän toimintavarmuudella (reliability) tarkoitetaan järjestelmän tai komponentin kykyä suorittaa vaadittu toiminta tietyissä olosuhteissa ja tietyinä ajanhetkenä [27]. Tässä luvussa tarkastellaan toimintavarmuutta yleisellä tasolla. Lisäksi tarkastellaan, mitä pitää ottaa huomioon, jotta etäluettavalle lämpötilan mittaus- ja tilastointijärjestelmälle saadaan mahdollisimman hyvä toimintavarmuus.

#### **3.2.1 Virransaannin varmistamisen ongelmat**

Etäluettava lämpötilan mittaus- ja tilastointijärjestelmä koostuu yhdestä tai useammasta sähkölaitteesta, kuten luvussa 2 on kuvattu. Täten järjestelmän sähkölaitteiden virransaannin varmistaminen ja siihen liittyvien ongelmien huomioiminen on tärkeää, koska

järjestelmästä halutaan mahdollisimman toimintavarma. Tässä luvussa perehdytään millaisia riskejä virtalähde voi aiheuttaa elektronisille laitteille ja miten näitä riskejä pystytään minimoimaan.

Järjestelmän toimintavarmuus heikkenee, jos se sammuu virran katketessa. Tällöin järjestelmään ei saada yhteyttä eikä järjestelmä pysty jatkamaan lämpötilan tilastointia. Tyypillinen syy virran katkeamiselle on sähkökatkos, joka yleensä johtuu palveluntarjoajasta ja sähköinfrastruktuurista. Syynä voi esimerkiksi olla vahingoittunut sähkölinja. Tänä päivänä äkillinen virran katkeaminen harvoin aiheuttaa sähkölaitteille mitään ongelmia. Tietokoneiden tapauksessa suurin ongelma on lähinnä käsiteltävän datan menettäminen ja tietysti tietokoneen toimimattomuus sähkökatkoksen aikana. [54]

Lisäksi ali- sekä ylijännite saattavat rikkoa laitteen tai saada sen toimimaan väärällä tavalla. Virtapiikki (power surge) tarkoittaa jännitteen äkkinäistä kasvua ja johtuu muiden samaa virtalähdettä käyttävien sähkölaitteiden sähkökulutuksen vaihtelusta. Paljon sähköä vaativien sähkölaitteiden käynnistäminen ja sammuttaminen aiheuttavat muille laitteille jännitteen äkkinäisiä vaihteluja. Virtapiikin vakavuus riippuu muutosten suuruudesta ja ajoituksesta. [22] [54]

Jännitepiikki (voltage spike) on kuin virtapiikki, mutta huomattavasti vakavampi. Jännitepiikit aiheuttavat suurimman riskin tietokoneille ja elektroniikalle, koska niiden aikana johdinten jännitteet voivat kasvaa siinä määrin suuriksi että eristeet ja virtapiiristö (circuitry) menevät rikki. [22] [54]

Kaikki edellä mainitut virransaantiin liittyvät riskit etäluettavalle mittaus- ja tilastointijärjestelmälle kyetään ratkomaan käyttämällä keskeytymätöntä virransyöttöjärjestelmää (uninterruptible power supply). Keskeytymätön virransyöttöjärjestelmä on järjestelmä, joka kytketään sähkölaitteen ja virtalähteen eli vaikka pistorasian väliin. Tällöin sähkölaitteelle syötettävä virta annetaan tämän UPS-järjestelmän läpi. Keskeytymätön virransyöttö takaa sen, että järjestelmä saa aina virtaa oikealla jännitteellä. UPS-järjestelmän akku myös varmistaa, että järjestelmästä ei katkea virta vaikka sähköt lähtisivät ja samalla UPS-järjestelmästä lähtisi sähköt. [2] [32] UPS-järjestelmistä on malleja joissa on C13 IEC liittimiä [6] ja malleja joissa on normaaleja pistorasioita [5]. UPS-järjestelmä takaa sen, että mittaus- ja tilastointijärjestelmä jatkaa toimintaansa, vaikka sähköt katkeaisivatkin. Tämä parantaa järjestelmän toimintavarmuutta.

UPS-järjestelmän lisäksi luvun alussa esiteltyjä riskejä pystytään minimoimaan myös akkupankkien ja akkukäyttöisten sähkölaitteiden avulla. Akkupankeilla ja akuilla pystytään varmistamaan, että sähkölaitteesta ei lähde heti sähköt vaikka virtalähteestä katkeaisikin sähköt. [54] Lisäksi akkupankki tasaa järjestelmän saamaa virtaa ja ne ovat huomattavasti halvempi vaihtoehto UPS-järjestelmiin verrattuna. On myös hyvä mainita, että helppo tapa suojata sähkölaite ylijännitteeltä on käyttää ylijännitesuojaa [54].

### 3.2.2 Verkkoyhteyden katkeaminen

Luvussa 2 esiteltyihin arkkitehtuureihin nojautuen voidaan sanoa, että toimiva verkkoyhteys on välttämättömyys etäluettavan mittaus- ja tilastointijärjestelmän toiminnassa, koska etälukeminen tapahtuu verkkoyhteyttä käyttäen. Verkkoyhteyden katketessa järjestelmään ei luonnollisesti saada yhteyttä, jolloin mitattuja ja tilastoituja arvoja ei päästä enää lukemaan. Tällöin järjestelmän toimintavarmuus heikkenee. Verkkoyhteyden katkeaminen siis kiistatta aiheuttaa suuren ongelman etälukemiseen. Tässä luvussa käsitellään yleisimpiä syitä verkkoyhteyden katkeamiseen ja miten ongelma voitaisiin ratkoa.

Yleinen syy verkkoyhteyden katkeamiseen on, että reitittimestä katkeaa virta sähkökatkoksen takia. Virran katkeaminen reitittimestä on kuitenkin helppo korjata käyttämällä UPS-järjestelmää. UPS-järjestelmä on selitetty tarkemmin luvussa 3.1.1. Samalla UPS-järjestelmä suojaa reititintä ali- ja ylijännite tilanteissa.

Toinen yleinen syy verkkoyhteyden katkeamiseen on operaattorin ongelmat runkoverkossa ja mobiiliverkossa. Ainoa tapa suojautua tällaiselta ongelmalta on laittaa järjestelmään varalle verkkoyhteys joltain toiselta operaattorilta. Tällöin jos ensisijaisen operaattorin tarjoamassa verkkoyhteydessä on jotain ongelmaa, voidaan nopeasti ja automaattisesti vaihtaa toisen operaattorin verkkoon. Tällöin järjestelmän verkkoyhteys palautuu ja siihen päästään etäyhteydellä käsiksi. Lisäksi toimintavarmuutta lisää se, jos toinen verkko toimii eri tekniikalla. Ensisijainen verkko voisi esimerkiksi toimia valokuidulla ja toissijainen mobiiliverkolla. Linux-käyttöjärjestelmissä pystyy asettamaan tietokoneen eri verkkoyhteyksille prioriteettejä ifmetric -nimisen ohjelman avulla. Jos ensisijainen verkkoyhteys ei ole saatavilla, ohjelma vaihtaa automaattisesti toissijaiseen ja järjestelmän toiminta palautuu [28]. Myös Windows-käyttöjärjestelmissä on mahdollista asettaa verkkoyhteyksiin automaattimen yhdistäminen prioriteetilla. Tällöin jos ensisijainen verkkoyhteys ei ole saatavilla, vaihdetaan automaattisesti toiseen. [20] [21] Samanlaisen toiminnallisuuden voi saavuttaa myös liittämällä järjestelmä 4G-reitittimeen, johon on mahdollista saada useamman kuin yhden operaattorin verkkoyhteys. Jos ensisijainen verkkoyhteys katkeaa, reititin automaattisesti vaihtaa toissijaiseen verkkoyhteyteen ilman, että reitittimeen liitetyn laitteen tarvitsee tehdä mitään toimenpiteitä. [47] Vastuu verkkoyhteyden vaihtamisesta siis annetaan reitittimelle.

### 3.2.3 Tallennusmuistin korruptoituminen

Luvun 2 johtopäätösten myötä voidaan pitää datan tallentamista välttämättömänä toimenpiteenä toteutettaessa etäluettavaa mittaus- ja tilastointijärjestelmää. Lämpötilan historia-datan tallentaminen on järjestelmälle asetettujen tavoitteiden toteutumisen kannalta tärkeää. Ainoastaan tallentamisella arvo saadaan selville myös tulevaisuudessa ja voidaan näyttää käyttäjälle. Jos tallennusmuisti, johon mittaus- ja tilastointijärjestelmä tallentaa dataa hajoaa, kaikki data menetetään. Tämä aiheuttaa ilmeisen ongelman järjestelmälle, koska järjestelmältä vaaditaan hyvää toimintavarmuutta. Muistin hajotessa tai korruptoi-

tuessa järjestelmä ei kykene suorittamaan sen yhtä tärkeimmistä tehtävistä, eli näyttämään käyttäjällä lämpötilan historiadataa. Tämä taas johtaa toimintavarmuuden heikkenemiseen. Tässä luvussa tarkastellaan datan tallennuksen aiheuttamien riskien minimoimista mittaus- ja tilastointijärjestelmälle.

Datan häviäminen tallennusmuistin hajoamisen vuoksi voidaan estää ottamalla datasta säännöllisesti varmuuskopioita. Järjestelmää toteutettaessa luvussa 2.2 kuvatulla paikallisella prosessointiyksiköllä datan varmuuskopiointi voidaan toteuttaa asettamalla data automaattisesti synkronisoivan kansion sisään. Data voi olla mitä vain tekstitiedostoista tietokantatiedostoihin. Tällöin aina kun datan sisältö muuttuu, uusi versio datasta synkronisoituu automaattisesti pilveen. Tämän toteutuksen etuna on sen helppokäyttöisyys, koska se ei vaadi muita toimenpiteitä kuin synkronisointisovelluksen asentamisen ja datan tallennuskohteen asettamisen synkronisointikansioon. Kyseinen varmuuskopiointijärjestelmä voidaan toteuttaa esimerkiksi ownCloud-palvelimella ja ownCloud-työpöytäsovelluksella, jonka pystyy asentamaan Windows- ja Linux-käyttöjärjestelmiin. [57] Toteutuksessa on kuitenkin yksi suuri heikkous: jos data korruptoituu, sitten myös korruptoitunut versio datasta kopioidaan pilveen. Kuvattu varmuuskopiointimenetelmä siis toimii vain tilanteessa, jossa paikallinen muisti hajoaa kokonaan.

Fang et al. esittävät paremman tavan toteuttaa varmuuskopiointijärjestelmä luvun 2.2 kuvaamaan arkkitehtuuriin. [16, s. 2] Fang et al. kuvailevat arkkitehtuurissaan Kaksitasoisen tallennus-strategiamallin (Two-tier Storage Strategy Design). Tässä mallissa järjestelmään liitetään ulkoinen tietokantapalvelin, johon paikallinen palvelin tallentaa reaaliaikaista dataa. Nyt siis paikallinen palvelin tallentaa oman paikallisen tietovaraston lisäksi kaiken datan myös pilvessä sijaitsevaan tietokantaan. Tämä varmistaa sen, että jos paikallinen datan tallennus pettää, data on silti turvassa pilvessä tietokannassa. Tämä menetelmä suojaa myös datan korruptoitumiselta, koska järjestelmässä ei kopioida tiedostoja, vaan sama data tallennetaan samaan aikaan sekä paikallisesti että pilveen.

Luvun 2.2 arkkitehtuurin mukaiselle järjestelmälle on olemassa vielä yksi sopiva vaihtoehto varmuuskopioinnille. Järjestelmään voidaan ottaa käyttöön RAID (Redundant Array of Independent Disks). RAID on virtualisointiteknologia, jonka avulla voidaan yhdistää yksi tai useampia fyysisiä kovalevyjä samalla saavuttaen datalle redundanttiutta, tehokkuutta tai molempia. RAID-teknologiasta on useita versioita ja esimerkiksi RAID 1 toimii niin, että kaikki data tallennetaan automaattisesti samanlaisena kahdelle tai useammalle muistilevyille. Tällöin jos yksi levyistä hajoaa se ei vielä tarkoita datan ja järjestelmän menettämistä, koska kaikki sama data löytyy myös muilta levyiltä. RAID-järjestelmän avulla siis kyetään parantamaan etäluettavan mittaus- ja tilastointijärjestelmän toimintavarmuutta, koska dataa ei todennäköisesti menetetä. [34]

Tarkastellaan vielä lopuksi, miten luvun 2.1 arkkitehtuurissa datan varmuuskopiointi voisi tapahtua. Luvun 2.1 esittelemä arkkitehtuuri tarjoaa luonnostaan hyvän turvan datalle. Kyseisessä arkkitehtuurissa datanprosessointiyksikkö sijaitsee jo valmiiksi pilvessä. Jos pilvi on jonkun kolmannen osapuolen ylläpitämä, niin varmuuskopiot ovat silloin tämän kolmannen osapuolen vastuulla. Esimerkiksi Amazon AWS ja Microsoft Azure tarjoavat

pilvitietokantapalvelua, johon saa asetettua automaattisen varmuuskopioinnin haluamallaan aikavälillä. Palvelulla pystyy palauttamaan tietokannan mihin tahansa aiempaan tilaan. [3] [7] Yhteenvetona pilvipalvelut helpottavat käyttäjää varmuuskopioinnin järjestämisessä, koska se tapahtuu kokonaan pilvessä ja ei ole käyttäjän vastuulla.

### 3.2.4 Oma verkkotunnus ja staattinen IP-osoite

Järjestelmän toimintavarmuuden parantamiseksi on kannattavaa asettaa sille oma verkkotunnus jonkin DDNS-palvelun kautta [64]. Suomessa ilmaista DDNS-palvelua tarjotaan verkkosivulla [www.dy.fi](http://www.dy.fi) [15]. Lisäksi on syytä asettaa järjestelmälle staattinen IP-osoite reitittimen DHCP-palvelimeen. Tässä luvussa tarkastellaan yleisesti, miten ja miksi nämä toimenpiteet kannattaa tehdä.

Useimmissa reitittimissä on dynaaminen ulkoinen IP-osoite. Tämä tarkoittaa, että IP-osoite voi vaihtua operaattorin toimesta esim. tilanteessa, jossa reititin on ollut sammukissa useamman tunnin tai jos operaattori tekee muutoksia järjestelmässään. Tällöin reitittimelle saatetaan antaa uusi IP-osoite, ja vanha IP-osoite ei enää siis toimikaan. Järjestelmän näkökulmasta ulkoisen IP:n vaihtuminen aiheuttaa ongelman järjestelmään yhdistämisessä, koska jos verkon IP-vaihtuu, verkkoon, ja täten myöskään järjestelmään, ei saada enää yhteyttä. Tämä johtuu siitä, että sisäverkon ulkopuolelta on mahdoton sanoa mikä reitittimen uusi IP-osoite on. [64] Ongelma voidaan ratkoa käyttämällä kolmansien osapuolien DDNS (Dynamic Domain Name Server) palveluita, joiden avulla reitittimen IP-osoitteelle voidaan antaa oma verkkotunnus (domain name). Verkkotunnuksen asettamisella järjestelmään voidaan yhdistää aina saman verkkotunnuksen avulla, joka on helpompi muistaa kuin ajoittain vaihtuva IP-osoite. [64]

DDNS-palvelun käyttäminen ei tosin vielä ratkaise reitittimen IP:n ajoittaisesta vaihtumisesta aiheutuvaa ongelmaa. Vaikka reitittimen ulkoiselle IP-osoitteelle annettaisiinkin oma verkkotunnus DDNS-palvelussa, reitittimen IP-osoite voi silti muuttua. Tällöin DDNS-palvelun luoma verkkotunnus viittaisi yhä vanhaan IP-osoitteeseen, jolloin yhdistäminen ei onnistu tai yhteydenotto menisi jonnekin toiselle palvelimelle, jos IP-osoite on ehditty antaa jo jonkun toisen palvelimen käyttöön. Ongelma voidaan ratkaista asettamalla reitittimen sisäverkkoon suorittumaan sovellus, joka käy DDNS-palvelussa päivittämässä IP-osoitteen jos se vaihtuu. Palvelimen tulee tietyn väliajoin tarkastaa onko IP-osoite vaihtunut. Tällöin järjestelmän toimintavarmuus paranee huomattavasti, koska järjestelmään saadaan yhteys hyvin nopeasti uudestaan, vaikka reitittimen ulkoinen IP-osoite vaihtuu. Tämä toimintatapa tosin vaatii sen, että DDNS-palvelussa on jonkin rajapinta, jonka kautta IP-osoitteen päivittäminen kytetään tekemään ohjelmallisesti. [64]

Ulkkoisen IP-osoitteen lisäksi järjestelmän IP-osoite reitittimen DHCP-palvelimessa tulee asettaa staattiseksi. Näin varmistetaan, että järjestelmän IP-osoite ei muutu reitittimen alaisessa verkossa, jos vaikka järjestelmä sammuu useiksi tunneiksi. Muuten järjestelmälle annettu IP-osoite voi vaihtua ja tällöin siihen ei saada enää ulkopuolelta yhteyttä, koska ulkoa ei tiedetä, mikä järjestelmän uusi IP-osoite on. [25]

Edellä mainitut IP-osoitteiden vaihdokset aiheuttavat ilmeisen ongelman etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän toimintavarmuudelle. IP-osoitteiden vaihtuessa järjestelmään ei enää saada yhteyttä, jolloin järjestelmä ei pysty suorittamaan vaadittua toimintaansa. Käytännössä se tarkoittaa että järjestelmän toimintavarmuus heikenee.

### 3.3 Tietoturvallisuus

Tietoturva tarkoittaa prosesseja ja työkaluja, joita käytetään sensitiivisen datan suojaamiseen tarkastelulta, muuttamiselta ja tuhoamiselta ilman tarvittavia oikeuksia [61]. Tietoturva on tänä päivänä ehkä tärkeämmässä roolissa kuin koskaan ennen. Tietoturvan tärkeys korostuu etenkin web-palveluissa. Yleisessä tapauksessa web-palvelun julkisiin rajapintoihin pääsee käsiksi aivan kuka vain. Siksi onkin äärimmäisen tärkeää, että verkossa vapaasti saatavilla olevat rajapinnat suojataan, jotta niiden kautta ei kyetä aiheuttamaan vahinkoa palvelulle tai sen käyttäjille. Nykyään tietoturvaan panostetaan suuret määrät resursseja, koska panostamatta jättäminen saattaa johtaa suuriin taloudellisiin tappioihin ja organisaation maineen heikkenemiseen.

Luvun 2 mukaan on ilmeistä, että etäluettava lämpötilan mittaus- ja tilastointijärjestelmä luo julkiseen verkkoon rajapinnan, jota kautta dataan pääsee käsiksi. Tällöin järjestelmään voi kohdistua samoja tietoturvariskejä kuin mihin tahansa web-palveluun. Siksi onkin tärkeää tarkastella näitä tietoturvariskejä, koska toteutettavalta järjestelmältä vaaditaan korkeaa tietoturvallisuutta. Luvussa 3.3.1 tarkastellaan, miten järjestelmää vahvistetaan ja piilotetaan siten, että siihen hyökkäämisestä tulisi mahdollisimman vaikeaa. Luvussa 3.3.2 käydään läpi yleisellä tasolla millaisia erilaisia tietoturvariskejä internetiin liitettyyn web-palveluun voi kohdistua ja, miten näiltä voidaan suojautua. Verkkosivulla "owasp.org" on saatavilla lukuisia ilmaisia tietoturvaa käsitteleviä artikkeleita, dokumentteja ja työkaluja. Sivulla pidetään yllä listaa kymmenestä tällä hetkellä kaikkein vaarallisimmasta web-palveluihin kohdistuvasta tietoturvariskistä [49]. Tässä luvussa käsitellään niistä injektio, cross-site scripting ja rikkinäinen autentikaatio.

#### 3.3.1 Järjestelmän vahvistaminen ja piilottaminen

Eräs tehokas tekniikka vähentää web-palveluihin kohdistuvia tietoturvariskejä on pyrkiä vahvistamaan ja piilottamaan ne tahoilta, joiden ei tarvitse käyttää järjestelmää (hardening). Palvelun vahvistaminen on prosessi, jossa pyritään turvaamaan systeemi vähentämällä mahdolliset haavoittuvuudet minimiinsä. Lyhyesti voidaan sanoa, että mitä enemmän erilaisia toimintoja palvelulla on, sitä enemmän hyökkääjillä on mahdollisia reittejä käytettäväänään ja sitä tietoturvattomampi järjestelmästä tulee. Toisin sanoen palvelimelta tulee sulkea kaikki palvelut, joita siinä ei tarvita. [38]

Etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän käyttäjä tai ylläpitäjä voi haluta ottaa järjestelmään SSH-yhteyden ylläpidollisten toimintojen tekemiseksi. Tätä var-

ten palvelimessa pitää käynnistää SSH-palvelin. Samalla avautuu uusi reitti mahdollisille hyökkääjille hyökätä palveluun, koska nyt SSH-palvelimeen voidaan yhdistää mistä tahansa IP-osoitteesta maailmassa. Internetissä on tuhansittain botteja, joiden tarkoitus on käydä läpi eri IP-osoitteita ja "kolkutella" niiden porttia 22 samalla kokeillen yleisimpiä tunnus-salasana-pareja. SSH-palvelin tyypillisesti käyttää juuri tätä porttia 22. Onkin siis erittäin tärkeää, että palvelimessa ei käytetä oletussalasanvoja tai muita liian heikkoja salasanvoja. Muuten palvelimeen voidaan helposti tunkeutua. [60] [38]

Myös HTTP- ja HTTPS-palvelimia pystytään vahvistamaan erilaisilla tekniikoilla. Eräs hyvä tekniikka on hylätä kaikki yhteydenotot palvelimelle, joiden otsikkotiedoissa (headers) olevan otsikkotiedon internetlaite (host) kentän arvo on IP-osoite, eikä palvelimen verkkotunnus DDNS-palvelussa. Oman verkkotunnuksen asettaminen DDNS-palveluun on esitelty luvussa 3.2.4. Teknisesti tämä tarkoittaa, että jos käyttäjä yhdistää palvelimeen kirjoittamalla palvelimen IP-osoitteen selaimen hakukenttään, niin kutsun internetlaitekentän arvoksi tulee tämä IP-osoite. Kun palvelin ottaa tämän kutsun vastaan, se voi hylätä pyynnön suoraan. Pyyntö oltaisiin hyväksytty, jos käyttäjä olisi yhdistänyt palveluun sen verkkotunnuksella. Hylkäämällä kaikki yhteydenotot, jotka otetaan IP-osoitteen kautta auttaa palvelua pysymään paremmin piilossa, koska IPv4-osoitteita on maailmassa pieni määrä. Bottien on helppo skannata IPv4 osoitteita läpi, koska melkein kaikki niistä ovat jossakin käytössä. Verkkotunnuksien skannaaminen on huomattavasti tuloksettomampaa, koska verkkotunnus voi olla pitkäkin merkkijono ja erilaisia mahdollisuuksia on paljon enemmän.

### 3.3.2 Web-palveluihin kohdistuvia hyökkäyksiä

Tarkastellaan ensin kaikkein yleisintä web-palveluihin kohdistuvaa hyökkäystä injektioita [49]. Injektio on web-palveluun tehtävä hyökkäys, jossa hyökkääjä yrittää saada web-palveluun suorittamaan omaa vahingollista koodiaan. Perinteinen tapa suorittaa injektio on syöttää sivun erilaisiin syötekenttiin SQL-komentoja. Kyseessä voi olla esimerkiksi kenttä, johon voisi kirjoittaa kommentin jotain kuvaa varten. Kun hyökkääjä tallentaa kommentin, vahingollinen kommentti siirtyy palvelimelle ja mahdollisesti tietokantaan. Jos hyökkääjä onkin kommentin tekstin sijasta syöttänyt SQL-komennon, se ajetaan samalla kun kommentti tallennetaan tietokantatauluun. Haitallinen koodi voisi sisältää esimerkiksi SQL-komennon "DROP DATABASE;", joka siis tuhoaisi koko tietokannan. Tietokanta ei tiedä, että kyseessä on haitallista koodia, jos web-palvelun ja tietokannan kehittäjä on unohtanut tehdä toimia injektioilta suojautumiseksi.[19, s. 1]

Vakavuudestaan huolimatta SQL-injektioilta on helppo suojautua, jos on perustietämystä käytetystä tietokannasta ja web-kehyksestä. Waltzer toteaa, että keskeisin menetelmä hyökkäyksien torjumiseksi on kaiken syöttötiedon huolellinen suodattaminen [58, s. 29]. Toisin sanoen web-palvelu ei voi luottaa mihinkään syötteisiin mitä sen käyttäjä sille syöttää. Kaikki käyttäjän syöttämät syötteet tulee validoida ja sanitoida, jotta missään tilanteessa syötteiden koodi ei joudu ajettavaksi. Tyyppien tarkistus on myös helppo tapa

estää SQL-injektio, esim. jos johonkin kenttään voi ainoastaan syöttää kokonaislukuja, niin kaikki muut syötteet voidaan hylätä [19, s. 6]. Tyypin tarkastaminen ei tosin auta jos syötteenä on merkkijono.

Merkkijonojen turvallinen validointi tapahtuu parametrisoitujen hakujen avulla. Parametrisoitu haku tarkoittaa sitä, että tietokannalle annetaan erillisellä listalla kaikki hakuun tarvittava käyttäjän antama data [12]. Listan alkioita kutsutaan hakuparametreiksi. Lisäksi tietokannalle annetaan merkkijono, joka sisältää varsinaisen haun, johon on merkattu tietokannalle ominaisella syntaksilla, mikä parametri menee mihinkin kohtaan. Tietokannan tehtäväksi jää yhdistää parametrit oikeisiin kohtiin haku-merkkijonoon ja ajaa haku. SQL-injektiota ei voi nyt tapahtua, koska tietokannalla on tieto siitä, mikä on käyttäjän antamaa syötettä. Näin ollen tietokanta tietää kohdella hakuparametreja normaaleina merkkeinä eikä SQL-komentoina. [12]

Injektioilta suojautuminen on tärkeää etäluettavissa lämpötilan mittaus- ja tilastointijärjestelmissä, jos järjestelmässä on minkäänlaisia syötekenttiä ja jos siinä on tietokanta. Syötekenttiä voisivat olla vaikka kentät joiden kautta pystyy muokkamaan käyttäjätietoja. Tällaisia kenttiä on esitetty lisättävän datankommunikointiyksikköön ainakin luvussa 2.1 esitetyssä järjestelmässä. Syötekenttien lisäksi luvun 2.1 järjestelmässä on tietokanta, joka tarkoittaa että SQL-injektion riski on kuvatussa järjestelmässä mahdollinen. [36, s. 3] SQL-injektioilta suojautuminen on relevanttia myös Zhang et al. esittämässä järjestelmässä, koska myös siinä on syötekenttiä ja tietokanta [64, s. 3].

Seuravaaksi tarkastellaan toista erittäin yleistä web-palveluihin kohdistuvaa hyökkäystä Cross-site scripting (XSS). Cross-site scripting (XSS) on jokseenkin samantapainen hyökkäys kuin injektio. XSS-hyökkäyksessä hyökkääjä lähettää palvelimelle esimerkiksi kommentissa JavaScript-koodia script-tagin sisällä. Tällöin vaarallinen koodi tallentuu tietokantaan. Myöhemmin kun joku toinen käyttäjä haluaisi lukea kyseisen kommentin, hän menisi selaimellaan HTML-sivulle, jolta kommentin voi lukea. Selain palauttaisi hänelle HTML-sivun. Tämän HTML-sivun sisällä olisi hyökkääjän vaarallinen JavaScript koodi kommentin sisällä, joka käynnistyy sivun latauduttua. Vaarallinen JavaScript koodi voisi toimia monella eri tavalla ja eräs tyypillinen tapa olisi lähettää uhrin keksi tai tunnukset hyökkääjän palvelimelle [58, s. 16]. Keksien saatuaan hyökkääjä pystyy esittämään uhria ja tekemään kaikkea mitä uhrikin pystyy kirjautuneena tekemään.

XSS-hyökkäykseltä suojautuminen tapahtuu hyvin samalla tavalla kuin SQL-injektioiltakin suojautuminen. Kaikki käyttäjän antama syöte tulee tarkistaa niin että sen mukana ei pääse JavaScript-koodia tietokantaan eikä script-tageja. XSS-hyökkäykseltä voi myös suojautua olemalla tallentamatta järjestelmään mitään käyttäjän antamaa syötettä. Näin missään tilanteessa käyttäjien mahdollisesti syöttämät vahingolliset JavaScript-koodit eivät voi päätyä palvelun sivuille muiden käyttäjien selaimien ajettaviksi. [4]

Cross-site scripting hyökkäykseltä suojautuminen on tärkeää etäluettavissa lämpötilan mittaus- ja tilastointijärjestelmissä, jos palvelussa on useita tunnuksia. Lisäksi XSS-hyökkäyksen riskin syntyminen vaatii, että palvelussa on syötekenttiä, joiden sisältö voi

joutua näytettäväksi jollakin palvelun sivulla muille käyttäjille. Hyökkääjä tarvitsee syötekenttiä, jotta hän saa järjestelmään sisään omaa haitallista koodiaan. Lisäksi jos järjestelmässä on useita käyttäjiä, kuten Mansor et al. esittämässä järjestelmässä [36], niin yhdenkin käyttäjän menettäessä tunnuksensa hyökkääjälle, niin hyökkääjä pystyy cross-site scripting hyökkäyksellä saamaan myös muita tunnuksia. Hyökkääjä voi saada ensimmäiset tunnukset palveluun esimerkiksi sähköpostihuijauksella.

Lopuksi tarkastellaan rikkinäistä autentikaatiota, joka on myös yleinen tietoturvariski web-palveluissa. Autentikaation turvallisuuden varmistaminen on tärkeää etäluettavissa mittaus- ja tilastointijärjestelmissä, koska useimmissa tällaisissa järjestelmissä on jonkinlainen autentikaatiomekanismi, kuten Mansor et al. julkaisemassa tutkimuksessa [36, s. 3], sekä Zhang et al. tekemässä tutkimuksessa [64, s. 3]. Usein autentikaatio tapahtuu tunnus-salasana-parilla. Rikkinäisellä autentikaatiolla viitataan verkkopalvelun vikoihin, jotka mahdollistavat hyökkääjän varastaa muiden käyttäjien salasanoja, avaimia tai sessioavaimia (session token). Tällaisten haavoittuvuuksien avulla hyökkääjä kykenee varastamaan toisen käyttäjän identiteetin verkkosivuun nähden. Salasanaperustaisessa autentikaatiossa eräs suurimmista ongelmista on, että käyttäjät valitsevat syystä tai toisesta liian heikon salasanan [24, s. 4]. Muistettavuus on yleisin syy valita liian heikko salana. Käyttäjä saattaa valita salasanan omasta elämästään tai valita saman salasanan kuin mitä käyttää muissakin palveluissa [24, s. 4]. Jos yksikin näistä palveluista joutuu tietomurron kohteeksi, hyökkääjän on helppo automatisoidusti alkaa testaamaan missä kaikissa muissa palveluissa on samoja tunnus-salasana-pareja.

Verkkopalvelun kehittäjät voivat pakottaa käyttäjän laittamaan vahvan salasanan asettamalla sille vähimmäismitan ja tiettyjä rajoituksia salasanan entropian suhteen. Ei ole kuitenkaan mitään varmaa tapaa estää käyttäjää käyttämästä samaa salasanaa kuin hänellä on muissa palveluissa. Voidaan kuitenkin antaa käyttäjälle mahdollisuus kirjautua kertakirjautumisen (single sign-on) avulla, joka vapauttaa käyttäjän useiden salasanojen muistamisesta.[24, s. 4]

Verkkopalvelun näkökulmasta rikkinäisen autentikaation riskiä voidaan vähentää käyttämällä kirjautumiseen ja sessionhallintaan valmiita laajalti käytettyjä moduuleja, koska niiden toteuttajat ovat perehtyneet kirjautumiseen liittyvään tietoturvasuuteen. Jos ohjelmoija haluaisi itse toteuttaa kaikki kirjautumiseen liittyvän toiminnallisuuden tietoturvallisesti, vaatisi se erittäin paljon aikaa ja resursseja. Siltikin jäisi suuri riski että palveluun jää tietoturva-aukkoja. Tietoturvaa lisää myös se, jos ohjelma on avoimen lähdekoodin. Avoimen lähdekoodin ohjelmat ovat turvallisempia, koska koodia on useimmissa tapauksissa tarkastellut useammat ihmiset kuin vain kehittäjät. Tällöin tietoturva-aukot löytyvät helpommin.

Turvallinen autentikaatio on käytännössä pakollinen, jos halutaan että etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän dataan pääsee käsiksi vain sallitut henkilöt. Turvallinen autentikaatio on ainoa järkevä tapa varmistaa tietoturvasuus.

## 4 JÄRJESTELMÄN TOTEUTUSVAIHTOEHDOT

Automaattinen lämpötilan mittaus- ja tilastointijärjestelmä voidaan toteuttaa lukuisilla eri laitteilla, käyttöjärjestelmillä ja sovelluskehysillä. Tässä luvussa tarkastellaan tarkemmin mitkä laitteet, käyttöjärjestelmät ja sovelluskehukset sopisivat etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän pohjaksi. Lisäksi tarkastellaan myös laitteita, käyttöjärjestelmiä ja sovelluskehysia, jotka eivät olisi sopineet tähän tarkoitukseen.

### 4.1 Laitteiston valinta

Luvussa 2 kuvattujen arkkitehtuurien nojalla voidaan todeta, että etäluettava mittaus- ja tilastointijärjestelmä tarvitsee jonkinlaisen fyysisen laitteiston toimiakseen. Esimerkiksi luvun 2 kuvaaman datankeräisyksikön tulee olla fyysinen laite ja sen tulee sijaita sensorin kanssa samassa ympäristössä mitattavan kohteen kanssa. Lisäksi luvussa 2.2 kuvatussa arkkitehtuurissa datanprosessointiyksikkö tarvitsee laitteen pohjaksi, joka kykenee lukemaan dataa datankeräisyksiköltä ja jakamaan sitä kommunikointiyksikölle. Tässä luvussa tarkastellaan millaisia laitteita järjestelmän pohjana voisi käyttää.

Tärkein vaatimus laitteistolle on, että siihen saa ohjelmiston suorittamaan ja että laitteessa on internetyhteys. Laitteeseen pitää myöskin saada liitettyä lämpötilasensori, jota pystyy lukemaan ohjelmallisesti. Lisäksi järjestelmän ja laitteen tulee kyetä luomaan julkiseen verkkoon rajapinta, jonka kautta lämpötiladataan päästään käsiksi tietoturvasyistä autentiikkaatiota käyttäen. On olemassa hyvin paljon laitteita, jotka täyttävät nämä vaatimukset. Tarkastellaan seuraavaksi niistä sopivimpia ja yleisimpiä.

Tarkastellaan ensin voiko järjestelmän toteuttaa käyttäen pelkkää Arduino mikro-ohjainta. Arduinon pystyy liittämään paljon erilaisia sensoreita, lämpötilasensori mukaan lukien. Lämpötilasensorin pystyy esimerkiksi liittämään I2C-väylään. Lämpötiladatan mittaaminen ei siis ainakaan muodostuisi ongelmaksi. [10] Lisäksi Arduino on helppo liittää internetiin, koska siitä on malleja, joissa on USB-portteja joihin voi asettaa PHPoC Shield WiFi-sovittimen [32]. Arduino Uno ja Mega ovat esimerkkejä tällaisista malleista.[41] PHPoC Shield WiFi-sovittimessa on sisäänrakennettu Web Serial Plotter -web applikaatio, joka luo käyttäjän selaimen ja adapterin välille WebSocket yhteyden. WiFi-sovitin lukee Arduinon sensorien dataa serial portin kautta ja lähettää datan WebSocket yhteyden läpi selaimelle. Selaimessa data otetaan vastaan ja näytetään visualisoituina graafeina. Selaimen käyttöliittymä ja graafit on toteutettu perinteisillä verkkotekniikoilla, kuten HTML, JavaScript ja CSS. Web Serial Plotter -web applikaation koodit ovat helposti saatavilla

Arduino IDE:n Examples osiossa. Arduinolla siis olisi melko suoraviivaista mitata ja tallentaa erilaisia suureita, kuten lämpötilaa ja päästä siihen käsiksi selaimella verkon yli. [41] Arduino toteutuksen ongelmaksi kuitenkin tulisi se, että Arduinossa ei ole mitään valmista tai helppoa tapaa tehdä HTTP-pohjaista tunnus-salasana-parilla toimivaa autentikaatiota. Valmiita autentikaation mahdollistavia kirjastoja ei tällä hetkellä ole olemassa Arduinolle [33]. Autentikaatio-toiminnallisuuden voisi silti kirjoittaa itse, mutta se olisi erittäin työlästä, koska Arduinon käyttämä ohjelmointikieli on vain kokoelma C/C++-funktioita [17]. C/C++-kielet eivät tunnetusti taivu kovinkaan hyvin web-ohjelmointiin. Valmiin autentikaatiokirjaston puuttuminen on liian suuri heikkous, koska tietoturva ja sen vaatima autentikaatio on yksi järjestelmän vaatimuksista. Muutenkin rajoitettu ohjelmointikielten valikoima vaikeuttaa järjestelmän muokkaamista ja laajentamista tulevaisuudessa. Täten Arduino ei soveltuisi kovin hyvin etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän toteuttamiseen.

Tarkastellaan seuraavaksi voiko vaatimukset täyttävän järjestelmän toteuttaa henkilökohtaisella tietokoneella (PC) tai palvelimella. Tälle ei pitäisi olla mitään estettä, koska henkilökohtaisissa tietokoneissa sekä, että palvelimissa on lähes poikkeuksetta mahdollisuudet päästä internetiin käsiksi. Lisäksi ne tukevat lähes kaikkia käyttöjärjestelmiä. Tietokone-pohjaisen järjestelmän etuna olisi myös tehokkuus ja lyhempi ohjelmiston kehittämisäika kehittyneiden kehitystyökalujen avulla [16]. Lämpötilan mittaaminen olisi mahdollista esimerkiksi USB-porttiin liitettävällä lämpötila-anturilla, jonka pääsee lukemaan ohjelmallisesti [53] [65]. Järjestelmän voisi siis toteuttaa henkilökohtaisella tietokoneella tai palvelimella. Nämä ovat kuitenkin verrattain hintavia ratkaisuja ja tietokone tai palvelin on muutenkin turhan järeä ratkaisu tämän kaltaisen ongelman ratkaisemiseen [16].

Tarkastellaan vielä lopuksi miten Raspberry Pi:n kaltaiset mikrotietokoneet soveltuisivat etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän toteuttamiseen. Mikrotietokoneiden etuna on niiden verrattain pieni hinta ja koko, sekä monipuoliset liitännät. Esimerkiksi lähes kaikissa Raspberry Pi mikrotietokoneissa on rivillinen GPIO (general-purpose input/output) -pinnejä, joihin pystyy liittämään lähes minkälaisia sensoreita tahansa, mukaanlukien lämpötilasensoreita [18]. Mikrotietokoneiden eduksi voidaan lukea myös se, että ne ovat tietokoneita, täten niihin voi asentaa lähes minkä tahansa käyttöjärjestelmän ja niihin on saatavilla kehittyneitä kehitystyökaluja, jotka lyhentävät kehittämisäikää [16]. Mainintana vielä, että useimmissa Raspberry Pi malleissa on Ethernet -portti tai WiFi, jotka mahdollistavat laitteelle internetyhteyden. Raspberry Pi siis mahdollistaa helposti mittaus- ja tilastointijärjestelmän suorittamisen ja HTTP -rajapinnan luomisen, joten se sopii todella hyvin toteutettavan järjestelmän pohjaksi. Mikrotietokoneiden ongelmaksi voidaan lukea niiden vähäinen keskusmuistin määrä. Esimerkiksi todella suositussa Raspberry Pi 3 Model B mikrotietokoneessa on vain 1 GB keskusmuistia, joka on tyypilliseen henkilökohtaiseen tietokoneeseen tai palvelimeen verrattuna todella vähän [44]. Kuitenkin kyseisessä laitteessa riittää keskusmuistia helposti yhden palvelimen käynnissä pitämiseen. Toisia mikrotietokoneita joilla vaatimukset täyttävän järjestelmän voisi toteuttaa on Orange Pi Prime [39], Banana Pi M3 [9] ja Rock64 [46].

## 4.2 Käyttöjärjestelmän valinta

Vaatimukset täyttävän järjestelmän toteuttamisessa voisi käyttää monia erilaisia käyttöjärjestelmiä. Etäluettavan mittaus- ja tilastointijärjestelmän pohjaksi kannattaa ottaa jokin käyttöjärjestelmä, koska ne tarjoavat suuren määrän valmiita toimintoja sekä työkaluja. Käyttöjärjestelmien avulla on esimerkiksi mahdollista suorittaa ohjelmia, hallita tiedostojärjestelmää ja saada internetyhteys. Näitä kaikkia toimintoja tarvitsee toteutettaessa etäluettavaa mittaus- ja tilastointijärjestelmää. Täten on perusteltua, että järjestelmän pohjaksi otetaan jokin käyttöjärjestelmä. Käyttöjärjestelmä on otettu järjestelmän pohjaksi mm. Zhang et al. tekemässä tutkimuksessa [64, s. 2].

Ensiksi tarkastellaan Linux-pohjaisia käyttöjärjestelmiä. Linux-pohjaisten käyttöjärjestelmien etuna on, että ne ovat kevyitä ja antavat hyvän pohjan suorittaa melkein pä millä tahansa kielellä ohjelmoituja ohjelmia. Näihin lukeutuu esimerkiksi Java, C++, C#, Python ja JavaScript/Node.js. Linux on myös erittäin kevyt. Yksi suosituin Linux-jakelu Ubuntu Server tarvitsee toimiakseen keskusmuistia vain 1 GB verran ja muistia 2.5 GB [55], joten sen saa toimimaan useimmilla tietokoneilla, palvelimilla sekä mikrotietokoneilla. Linuxin etu on myös se, että se on avoimen lähdekoodin käyttöjärjestelmä, joten sen asentaminen ja muokkaaminen on mahdollista ja ilmaista.

Linuxin lisäksi järjestelmän voisi toteuttaa Windowsilla. Myös Windows-käyttöjärjestelmään perustuvilla käyttöjärjestelmillä pystyy suorittamaan kaikilla yleisimmillä ohjelmointikielillä luotuja ohjelmistoja. Näihin lukeutuu esimerkiksi Java, C++, C#, Python ja JavaScript/Node.js. Windows 10 32-bit version minimivaatimus keskusmuistille on 1 GB ja muistille 16 GB [63]. Windows 10 kuitenkin toimii melko hitaasti jos sillä on käytössään vain 1 GB muistia. Jos järjestelmän haluaa toteuttaa käyttäen mikrotietokoneita, niin Windows 10 käyttöjärjestelmästä olisikin syytä käyttää IoT Core versiota, joka on suunniteltu pieniresurssisille laitteille, kuten mikrotietokoneille. IoT Core tarvitsee ilman näyttötukea toimiakseen vain 256 MB keskusmuistia ja näyttötuen kanssa 512 MB tai 768 MB jos prosessorin arkkitehtuuri on 64 bittiä. Muistia IoT Core tarvitsee vähintään 2 GB. [37] Windows 10 IoT Core:n etuna on sen ilmaisuus ja sen oleminen avoimen lähdekoodin käyttöjärjestelmä, toisin kuin henkilökohtaisille tietokoneille suunnattu Windows 10. IoT Core:lle saa suorittamaan kaikkia samoja ohjelmia kuin mitä Windows 10 käyttöjärjestelmällekin saa, joka mahdollistaa paljon eri vaihtoehtoja järjestelmän ohjelmointikielen ja sovelluskehityksen valinnassa.

## 4.3 Web-sovelluskehityksen valinta

Tässä luvussa perehdytään siihen, mitä web-sovelluskehitykset (web framework) ovat ja mitä hyötyä niistä voisi olla vaatimukset täyttävän etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän toteuttamisessa. Tässä luvussa rajataan tarkastelusta pois Arduinon kaltaiset mikro-ohjaimet, koska niille ei pysty ohjelmoimaan ohjelmia käyttäen web-sovelluskehityksiä. Luvussa 4.1 todetun mukaan mikro-ohjaimilla pystyy kehittämään oh-

jelmia tyypillisesti vain jonkin oman rajatun ohjelmointikielen avulla.

Mozilla Firefox -selaimen kehittäjien ylläpitämässä dokumentaatioissa web-sovelluskehys määritellään seuraavasti: web-sovelluskehukset ovat sovelluskehiksiä, jotka tekevät helpommaksi web-sovelluksien ylläpidon ja skaalaamisen. Web-sovelluskehukset antavat työkaluja ja kirjastoja jotka yksinkertaistavat tyypillisiä web-sivujen kehittämiseen liittyviä tehtäviä, mukaan lukien liikenteen reitittämistä oikeille käsittelijöille, tietokantojen kanssa vuorovaikuttamista, sessio- ja käyttäjäautentikaatiota, ulostulojen formatoimista sekä tietoturvaa ulkoisia hyökkäyksiä vastaan. [13] Tästä johtuen web-sovelluksen kehittäjän ei tarvitse itse käyttää aikaansa lukuisten suurien toiminnallisuuksien toteuttamiseen. Suurin osa ei-triviaaleista web-aplikaatioissa tarvittavista toiminnoista on valmiiksi toteutettu kirjastoina. Edellä mainittujen etujen vuoksi etäluettavan lämpötilan mittaus- ja tilastointijärjestelmän toteuttamisessa on syytä käyttää apuna web-sovelluskehiksiä. Luvussa käydään läpi web-sovelluskehiksiä, jotka sopisivat järjestelmän toteuttamiseen.

Tarkastellaan kuinka Java-pohjaiset web-sovelluskehukset soveltuisivat järjestelmän toteuttamiseen. Lämpötilan mittaaminen voisi ainakin onnistua mikrotietokoneiden kanssa, koska luvussa 4.1 todetun mukaan niissä usein on I2C-väylä johon pystyy liittämään lämpötilasensoreita. Javalla pystyy lukemaan dataa I2C-väylästä melko suoraviivaisesti [1]. I2C-väylästä on syytä mainita myös se, että sen arvo pystytään lukemaan Linux-käyttöjärjestelmissä tiedostojärjestelmän tiedostosta. Toisin sanottuna I2C-väylältä pystytään lukemaan dataa millä tahansa ohjelmointikielillä, joka pystyy lukemaan tiedostoja. [30] Myös USB-porttiin liitettävän lämpötilasensorin lukeminen voisi onnistua Javalla, koska esimerkiksi Yocto-Temperature -nimistä sensoria pystyy lukemaan Javalla [65] [66]. Javalle on myös toteutettu lukuisia web-sovelluskehiksiä, joissa on valmiina monet web-ohjelmointiin liittyvät perustoiminnot. Esimerkiksi Spring web-sovelluskehyksessä on Spring security -kehys, jolla voi hoitaa käyttäjäautentikaatiota [51]. Spring tarjoaa kaikki muutkin perinteiset web-palveluiden tarvitsevat toiminnot kuten palvelimen ja datankäsittelyn [50].

Myös JavaScript-ohjelmointikieli ja Node.js-web-sovelluskehys voisivat sopia järjestelmän toteuttamiseen. Tämän pitäisi onnistua, koska Node.js-web-sovelluskehysten NPM-pakettienhallintamanagerissa on kirjastoja I2C-väylän lukemiseen [26]. Lisäksi myös JavaScript-ohjelmointikielillä pystyy lukemaan Yocto-Temperature-lämpötilasensoria [66]. Node.js-web-sovelluskehyksellä on myös helppoa tehdä yksinkertainen kirjautumissivu ja käyttäjäautentikaatio [11] [59]. Järjestelmän voisi toteuttaa myös Django-nimisellä Python-pohjaisella web-sovelluskehyksellä. Django tarjoaa samat järjestelmän tarvitsemat toiminnot kuin Node.js [56].

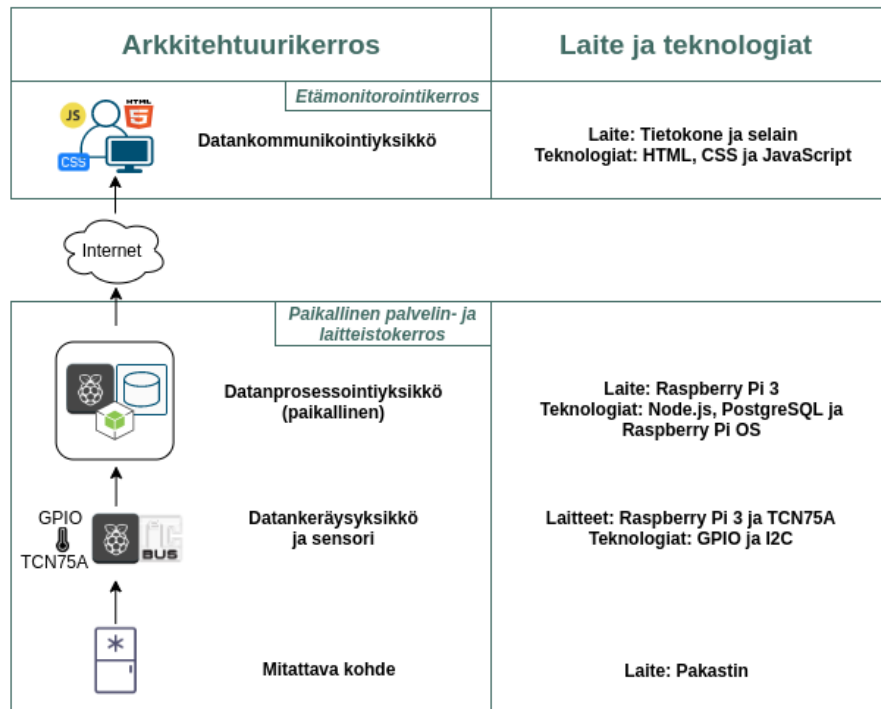
Aiempien lisäksi PHP-ohjelmointikieli sopisi vaatimukset täyttävän järjestelmän toteuttamiseen. Myöskään PHP-ohjelmointikielen käyttämiselle ole estettä, koska sillä on suoraviivaista tehdä palvelin, jossa on käyttäjäautentikaatio sekä kirjautumissivu [23]. Myös PHP-ohjelmointikielillä pystytään lukemaan Yocto-Temperature -sensoria [66] ja lukemaan tiedostoja tiedostojärjestelmästä, jolloin I2C-väylän lukeminen onnistuu myös [40].

## 5 TOTEUTETTU JÄRJESTELMÄ

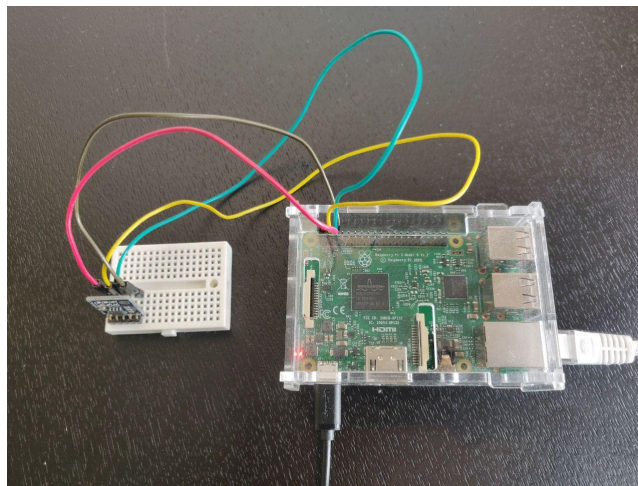
Yhdistetään luvussa 2 esiteltyä tutkimusta ja luvussa 4 tehtyä esitutkimusta ja esitelään yksi mahdollinen toteutusvaihtoehto etäluettavalle, toimintavarmalle ja tietoturvaliselle lämpötilan mittaus- ja tilastointijärjestelmälle. Luvussa arvioidaan myös kuinka luvussa 3 esitettyihin toiminnallisiin ja ei-toiminnallisiin vaatimuksiin päästiin. Koko järjestelmä toteutettiin avoimen lähdekoodin sovelluskomponentteja käyttäen. Järjestelmän Node.js sovelluksen koodeja voi tarkastella GitHub-verkkosivulla [43].

### 5.1 Järjestelmän kuvaus ja toiminnallisten vaatimusten täytyminen

Tutkimuksessa tavoitteena oli suunnitella ja toteuttaa etäluettava, toimintavarma ja tieturvallinen lämpötilan mittaus- ja tilastointijärjestelmä. Tutkimuksen järjestelmä toteutettiin mukailen luvussa 2.2 esiteltyä arkkitehtuuria, jossa datanprosessointiyksikkö sijaitsee paikallisessa ympäristössä datankeräysyksikön kanssa. Kuva 3 on kaaviokuva toteutetusta järjestelmästä. Kuvan arkkitehtuuri on muuten sama kuin kuvassa 2, mutta nyt kuvan oikealla laidalla on määritetty, millä laitteella ja teknologioilla kyseinen kerros on toteutettu. Paikallinen arkkitehtuuri valittiin, koska se ei vaadi pilvipalveluita datanprosessointiyksikön toteuttamiseen, kuten luvussa 2.1 esitelty arkkitehtuuri vaatisi. Tietokannan sisältävän pilvipalvelun pystyttäminen itsenäisesti olisi ollut liian aikaa vievää ja kolmannen osapuolen pilvipalvelut olisivat maksaneet liikaa suhteutettuna työn laajuuteen.



Kuva 3. Arkkitehtuurikaavio tutkimuksessa toteutetusta järjestelmästä. Kuva mukailee kuvaa 2, mutta siihen on lisätty oikeaan sarakkeeseen tiedot kerroksien laitteista ja teknologioista.



Kuva 4. Kuva toteutetusta järjestelmästä, jossa Raspberry Pi 3 -mikrotietokone ja TCN75A lämpötilasensori.

Toteutetun järjestelmän ainoa eroavaisuus luvun 2.2 arkkitehtuuriin on, että datanprosessointiyksikkö ja datankeräisyysyksikkö ovat sama laite. Kuvasta 3 on lisäksi syytä huomata, että siihen merkityt kaksi Raspberry Pi 3 -mikrotietokonetta esittävät samaa laitetta. Datanprosessointiyksikkö- ja datankeräisyysyksikkökerroksia ei yhdistetty kuvaan 3, vaikka ne sijaitsevatkin samassa laitteessa, koska näin kuvaa on helpompi verrata kuvaan 2. Lisäksi kerroksien toteutusteknologiat tulevat näin selkeämmin ilmi.

Järjestelmän laitteeksi valittiin Raspberry Pi 3 -mikrotietokone, koska se on markkinoilla helpoiten saatavilla oleva mikrotietokone. Lisäksi kuten luvun 4.1 esitutkimuksesta käy ilmi, Raspberry Pi 3 kykenee hoitamaan samaan aikaan sekä datanprosessointiyksikön että datankeräisyksikön tehtävät. Toteutettava järjestelmä on sen verran yksinkertainen, että datankeräisyksikön ei tarvitse olla erillinen. Jos toteutettavassa järjestelmässä olisi useampia erilaisia sensoreita, datankeräisyksikkö olisi kannattanut toteuttaa erillisenä laitteena. Tämä olisi onnistunut esimerkiksi liittämällä Raspberry Pi 3 -mikrotietokoneeseen Arduino-mikro-ohjain. Arduino-pohjaisessa datankeräisyksikössä olisi ollut se etu, että siinä on ADC (analog-to-digital conversion)-siru, joka muuttaa analogiset signaalit digitaalisiksi. Toisin sanoen tällöin olisi voinut käyttää myös analogisia sensoreita. [31]

Datan kerääminen suoritetaan käyttäen Raspberry Pi 3 -mikrotietokoneen GPIO-pinneihin liitettävää digitaalista TCN75A-lämpötilasensoria. Kyseinen sensori valittiin, koska se on helppo liittää neljällä johdolla Raspberry Pi 3 -mikrotietokoneeseen [52]. Sensorin lukeminen onnistuu millä tahansa ohjelmointikielellä koodatulla ohjelmalla, jossa on kyky lukea tiedostoja, kuten on mainittu luvussa 4.3. Tämä johtuu siitä, että käytettäessä Linux-pohjaisia käyttöjärjestelmiä I2C-väylä kyetään lukemaan tiedostosta. Lisäksi sensori on edullinen ja niitä saa ostettua myös Suomesta [14].

Järjestelmän käyttöjärjestelmäksi valikoitui Linux- ja Debian-pohjainen Raspberry Pi OS versiolla 10. Raspberry Pi OS valittiin, koska se on suositeltu käyttöjärjestelmä Raspberry Pi 3 -mikrotietokoneelle. Se on ilmainen ja optimoitu valmiiksi kyseiselle mikrotietokoneelle [45]. Kuten luvussa 4.2 todetaan, Linux-pohjaisuuden etuna on käyttöjärjestelmän avoimuus, pieni resurssientarve ja kyky suorittaa lähes kaikilla kielillä ohjelmituista ohjelmia. Pieni resurssientarve korostuu, koska järjestelmän pohjana on verrattain pienillä resursseilla varustettu Raspberry Pi 3.

Luvussa 2 esitelty datanprosessointiyksikkö on yksikkö, jonka tulee huolehtia datan tilastoinnista eli tallentamisesta. Datanprosessointiyksikön tehtäviin kuuluu myös tarjota tätä tilastoitua dataa tarvittaessa datankommunikointiyksikölle. datanprosessointiyksikön pohjaksi valittiin Raspberry Pi 3 ja Raspberry Pi OS, kuten aiemmin jo mainittiin. Tämän lisäksi datanprosessointiyksikköön kehitettiin ja laitettiin Node.js-palvelin, sekä PostgreSQL-tietokanta. Node.js valittiin, koska luvun 4.3 esitutkimuksen mukaan sille löytyy valmiita kirjastoja I2C-väylän lukemiseen. Node.js-web-sovelluskehityksellä on myös helppoa tehdä yksinkertainen kirjautumissivu. Valintaan vaikutti myös se, että Node.js-web-sovelluskehityksestä oli aiempaa kokemusta tutkimuksen tekijällä. Tietokannaksi valikoitui relaatiotietokanta PostgreSQL, koska myös siitä oli aiempaa kokemusta. PostgreSQL-tietokannan etuna on myös se että se on yksi suosituimmista avoimen lähdekoodin tietokannoista.

Järjestelmällä oli useita toiminnallisia vaatimuksia. Näistä tärkeimpänä oli kyky tarkastella mitattua ja tilastoitua lämpötilaa etänä. Tämä vaatimus toteutettiin luomalla Node.js-palvelimeen toiminnallisuus hakea ja näyttää käyttäjälle selaimessa JSON-formaatissa viimeiset 10 lämpötilamittausta. JSON-datan pääsee näkemään menemällä kirjautumi-

sen jälkeen selaimella polkuun "/tenlasttemps". Tämän lisäksi järjestelmälle asetettiin vaatimus, että sen tulee olla yleiskäyttöinen, niin että sitä pystyy käyttämään mahdollisimman pienillä muutoksilla myös muiden kohteiden mittaamiseen ja tilastointiin. Tämän pitäisi olla mahdollista, koska Raspberry Pi 3 -mikrotietokoneen GPIO-pinneihin pystyy liittämään myös toisenlaisia sensoreita. Lisäksi järjestelmä kyetään asentamaan mihin tahansa ympäristöön, jossa on internetyhteys ja verkkovirta, sekä normaalit huonelämpötilan vaihtelut.

## 5.2 Toimintavarmuuden ja tietoturvallisuuden huomioiminen

Tässä aliluvussa käsitellään millaisilla toimenpiteillä toteutetun järjestelmän ei-toiminnalliset vaatimukset toteutettiin. Ei-toiminnallisia vaatimuksia oli kaksi, toimintavarmuus ja tietoturvallisuus. Ensin käsitellään toimintavarmuuteen liittyvät ratkaisut ja sitten tietoturvallisuuteen liittyvät ratkaisut.

Seuraavissa kappaleissa käydään läpi miten luvussa 3.2 esitellyt etäluettaviin lämpötilan mittaus- ja tilastointijärjestelmiin liittyvät toimintavarmuusongelmat ratkottiin toteutetussa järjestelmässä. Luvussa 3.2.4 suositeltiin toimintavarmuuden parantamiseksi hankkimaan järjestelmän käyttämälle internetyhteydelle oma verkkotunnus jonkin DDNS-palvelun kautta. Järjestelmälle varattiin "dy.fi" DDNS-palvelun kautta verkkotunnus "prehe.dy.fi". Tämä vapauttaa käyttäjän IP-osoitteiden muistamiselta. Tämän lisäksi luvussa 3.2.4 suositellaan, että reitittimen sisäverkkoon laitetaan web-palvelin, joka käy DDNS-palvelussa päivittämässä IP-osoitteen, jos se vaihtuu. Tämä toteutettiin laittamalla samaan Raspberry Pi 3 -mikrotietokoneeseen suorittamaan toinen Node.js-palvelin, jonka tehtävä on käydä "dy.fi"-palvelun rajapinnassa päivittämässä "prehe.dy.fi"-verkkotunnuksen osoittama IP-osoite, jos se vaihtuu. Päivittäminen tapahtuu tekemällä Basic-autentikointia käyttävä kutsu osoitteeseen "https://www.dy.fi/nic/update?hostname=prehe.dy.fi". Kutsun URL-parametreissa annetaan päivitettävä verkkotunnus. Tunnukset annetaan kutsussa otsikkotietojen mukana. Oman verkkotunnuksen asettamisen lisäksi luvussa 3.2 suositeltiin asettamaan järjestelmälle staattinen IP-osoite sen käyttämän reitittimen luomassa DHCP-palvelimessa. Tämä tehtiin menemällä järjestelmän käyttämän TP-LINK-reitittimen hallinnointisivulle ja asettaen sieltä Raspberry Pi 3 -mikrotietokoneelle annettu IP-osoite staattiseksi.

Luvussa 3.2.1 todettiin, että eräs tehokas tapa suojata etäluettavaa lämpötilan mittaus- ja tilastointijärjestelmää virransaannin ongelmilta on asettaa järjestelmä saamaan virtansa UPS-järjestelmän läpi. UPS-järjestelmät ovat kuitenkin melko hintavia, joten työn budjetin takia UPS-järjestelmä korvattiin akkupankilla. Luvussa 3.2.1 todettiin että akkupankeilla kyetään saavuttamaan samanlaista toiminnallisuutta kuin UPS-järjestelmilläkin. Raspberry Pi 3 tarvitsee toimiakseen melko vähän voltteja (5.1V) ja ampeereita (1.2A) [42]. Useimmat akkupankit pystyvät täyttämään nämä vaatimukset helposti. Lisäksi akkupankilta vaaditaan että se pystyy samaan aikaan sekä syöttämään että vastaanottamaan virtaa. Muuten järjestelmä aina sammuisi siksi aikaa kun akkupankki latautuu. Järjestel-

mään kytkettiin “STREETZ”-merkkinen akkupankki, jossa on 16000 milliampeeria. Nyt vaikka akkupankista katkeaa virta, niin Raspberry Pi 3-mikrotietokoneesta ei katkea. Akkupankki siis varmistaa että järjestelmän toimintavarmuus ei heikkene sähkökatkoksien takia.

Luvussa 3.2.3 mainitaan, että tallennusmuistin korruptoitumisella ja hajoamisella on suuri heikentävä vaikutus järjestelmän toimintavarmuudelle. Tallennusmuistin hajotessa järjestelmä ei kykene suorittamaan yhtä sen tärkeimmistä toiminnallisista vaatimuksista eli näyttämään käyttäjälle etänä lämpötilan historiadataa, koska hajoamisen yhteydessä kaikki data menetetään. Edellä kuvattu ongelma ratkottiin luvussa 3.2.3 esitellyllä varmuuskopiointi vaihtoehdolla, jossa Raspberry Pi 3 -mikrotietokoneeseen asennetaan jokin automaattinen kansioiden synkronisointiohjelma. Tähän tarkoitukseen valittiin ownCloud työpöytäsovellus [29]. Kyseinen työpöytäsovellus mahdollistaa synkronisoitavien kansioiden asettamisen, jolloin kansiot varmuuskopioidaan automaattisesti ownCloud-palvelimelle aina kun niiden sisältö muuttuu. PostgreSQL-tietokannan tietokantatiedostot sisältävä kansio asetettiin ownCloud-työpöytäsovelluksen varmuuskopioitavaksi, jolloin järjestelmän data varmuuskopioituu automaattisesti pilveen. Nyt jos järjestelmän tallennusmuisti hajoaa ja paikallinen data menetetään, datat saadaan ladattua manuaalisesti pilvestä uudestaan järjestelmään. Järjestelmän toiminto historiadatan näyttämisestä ei siis vaarannu.

Verkkoyhteyden katkeaminen järjestelmästä aiheuttaa toimintavarmuuden heikentymistä ja tätä pystyttäisiin estämään hankkimalla järjestelmälle varalle toinen verkkoyhteys. Toimintavarmuus luvussa suositellaan että järjestelmää varten voisi hankkia toisen reitittimen, jossa olisi jonkun toisen operaattorin internetyhteys. Tällöin jos ensisijaiseen verkkoon tulee ongelmia, niin järjestelmä vaihtaisi automaattisesti varalla olevaan verkkoon. Kyseinen toiminnallisuus olisi siis vaatinut uuden reitittimen ja uuden kuukausisopimuksellisen internetyhteyden. Edellä mainitusta syystä vara verkkoyhteys jätettiin toteuttamatta järjestelmästä työn pienen budjetin vuoksi.

Seuraavissa kappaleissa tarkastellaan, miten luvussa 3.3 esitellyt etäluettaviin lämpötilan mittaus- ja tilastointijärjestelmiin liittyvät tietoturvaongelmat ratkottiin toteutetussa järjestelmässä. Luvussa 3.3.1 tarkasteltiin millä toimilla järjestelmää kyetään vahvistamaan ja piilottamaan sellaisilta tahoilta, joiden ei haluta pääsevän sisälle palveluun. Kyseisessä luvussa suositellaan sulkemaan palvelusta kaikki sellaiset toiminnot, joita palvelu ei tarvitse toiminnassaan. Näin minimoidaan mahdollisia hyökkäysreittejä, joita hyökkääjät voisivat hyödyntää. Tämä saavutettiin asentamalla Raspberry Pi OS -käyttöjärjestelmästä puhdas uusi asennus, jossa oletuksena käyttämättömät palvelut on suljettuina. Lisäksi järjestelmän käyttämä Raspberry Pi 3 -mikrotietokone rajoitettiin pelkästään tämän tutkimuksen käyttöön, jolloin mahdolliset hyökkäysreitit jäävät vain niihin palveluihin, joita järjestelmä tarvitsee. Nämä palvelut ovat 80 portissa toimiva HTTP, 443 portissa toimiva HTTPS ja 22 portissa toimiva SSH-palvelin. HTTP- ja HTTPS-portteja tarvitaan, koska niiden kautta järjestelmän käyttöliittymän pääsee näkemään selaimen kautta. SSH-palvelinta taas tarvitaan, jos joskus tulee tarvetta ottaa järjestelmään yhteys pääkäyttäjän oikeuksilla eri-

laisten hallinnollisten toimenpiteiden suorittamiseksi. Raspberry Pi -mikrotietokoneen salasanaaksi valittiin pitkä 23 merkin salasanana, koska luvun 3.3.1 mukaan internetissä on tuhansittain botteja, jotka "kolkuttelevat" eri IP-osoitteiden porttia 22 samalla kokeillen niihin erilaisia tunnus-salasana-pareja. Täältä suojautumiseksi onkin syytä valita erittäin vahva salasana.

Seuraavaksi tarkastellaan, miten järjestelmässä suojaututtiin web-palveluihin kohdistuvilta yleisimmiltä riskeiltä. Kyseisessä luvussa todettiin että jos palvelussa on yksikin syötekenttä ja tietokanta, niin SQL-injektioilta suojautuminen on otettava huomioon. Palvelussa on tietokanta ja kaksi syötekenttää kirjautumissivulla, joten SQL-injektio tulee ottaa huomioon. Toteutetussa järjestelmässä suojaututtiin SQL-injektioilta parametrisoitujen hakujen avulla käyttäen "node-postgres"-npm-pakettia. Tämän npm-paketin kautta tietokantaa kytetään käyttämään niin, että sille annetaan käyttäjän antama syöte erillisenä muuttujana, jonka nimi on hakuparametrit. Näin tietokanta saa tiedon, että mikä osa hakulauseesta on käyttäjän syötettä ja mihin siis tulee suhtautua varauksella. Näin järjestelmä saatiin suojattua SQL-injektioiden varalta.

Aiemmin mainittiin, että Cross-site scripting (XSS) on eräs pahimmista tietoturvariskeistä web-palvelulle. Järjestelmä suojattiin XSS-hyökkäyksiltä pitämällä se niin yksinkertaisena, että palvelussa ei ole ainuttakaan kenttää jonka syöte tallennettaisiin järjestelmään. Kuten edellä olevassa kappaleessa mainittiin, palvelun käyttöliittymässä on vain 2 syötekenttää ja kumpaankaankaan kenttään syötettyjä syötteitä ei tallenneta missään vaiheessa. Täten XSS-hyökkäyksen riskiä ei voi syntyä, koska käyttäjillä ei ole mitään tapaa saada järjestelmään sisään omaa vahingollista JavaScript-koodiaan.

Luvun 3.3.2 lopussa tarkasteltiin rikkinäistä autentikaatiota, joka on eräs yleisimmistä web-palvelujen tietoturvariskeistä. Mainittu tietoturvariski tulee ottaa huomioon toteutetussa järjestelmässä, koska siihen toteutettiin tunnus-salasana-parilla toimiva käyttäjäautentikaatio. Rikkinäiseltä autentikaatiolta suojautumiseksi autentikaatio toteutettiin käyttäen suosittuja npm-paketteja. Tunnus ja salasanan tiiviste lisättiin tietokantaan manuaalisesti, koska järjestelmän käyttöliittymässä ei ole rekisteröitymissivua. Käyttäjän kirjautuessa järjestelmän kirjautumissivulla "login.html" ja painaessa "Login"-nappia, syöteyt tunnus ja salasana lähetetään Node.js-palvelimelle tarkastettavaksi. Node.js-palvelin hakee tietokannasta olemassaolevan tunnuksen ja tiivisteiden ja vertaa sitä käyttäjän syötämiin vastaaviin. Tiivisteiden luominen ja vertaaminen tehdään erittäin suositulla npm-paketilla "bcrypt". Jos ne täsmäävät, niin käyttäjän todetaan olevan tunnistautunut ja hänelle luodaan ja lähetetään JSON Web Token. Käyttäjän selain tallentaa JSON Web Tokenin ja jatkossa jokaisen kutsun yhteydessä lähettää sen kutsun mukana, jolloin Node.js-palvelimella kytetään tunnistamaan kutsun tekijä ja hyväksymään tunnistautumista vaativat kutsut. JSON Web Token -toiminnallisuus toteutettiin käyttäen suosittua npm-pakettia "jsonwebtoken". Näin järjestelmään saatiin toteutetuksi tietoturvallinen autentikaatio.

## 6 YHTEENVETO

Tässä tutkimuksessa pohdittiin kysymystä “miten toteuttaa etäluettava lämpötilan mittaus- ja tilastointijärjestelmä toimintavarmasti ja tietoturvallisesti avoimen lähdekoodin komponenteista”. Tutkimuksessa toteutettu järjestelmä sopii käyttäjille, jotka tarvitsevat järjestelmälleen erityisen suurta toimintavarmuutta ja tietoturvallisuutta. Järjestelmän toteuttamista varten luvussa 2 perehdyttiin tyypillisiin etäluettavien mittaus- ja tilastointijärjestelmien arkkitehtuureihin, koska niistä tietäminen helpottaa ymmärtämään toimia, joilla järjestelmää optimoitiin asetettujen vaatimusten näkökulmasta. Lisäksi luvussa 2.2 esitelty arkkitehtuuri otettiin toteutetun järjestelmän arkkitehtuurin pohjaksi, joten siitä tietäminen oli tärkeässä roolissa toteutuksen ymmärtämiseksi. Luvussa 3 käsiteltiin millaisia toimintavarmuuteen ja tietoturvallisuuteen liittyviä ongelmia etäluettavilla mittaus- ja tilastointijärjestelmillä tyypillisesti on. Luvussa 4 käytiin läpi erilaisia vaihtoehtoja laitteiston, käyttöjärjestelmien ja web-sovelluskehysten näkökulmasta järjestelmän toteuttamiseksi. Lopuksi luvussa 5 esiteltiin eräs toteutusvaihtoehto vaatimukset täyttävälle järjestelmälle.

Tutkimuksessa saavutettiin johdannossa asetetut tavoitteet hyvin, koska löydettiin lukuisia erilaisia tapoja optimoida etäluettavan mittaus- ja tilastointijärjestelmän toimintavarmuutta ja tietoturvallisuutta. Järjestelmästä tuli kuitenkin melko yksinkertainen ja tulevaisuudessa sitä voisi kehittää vielä esimerkiksi visualisoidulla lämpötilan historiadatalla, sekä luomalla järjestelmään hallintapaneelin. Hallintapaneelista pystyisi lisäämään järjestelmään uusia käyttäjiä ja muuttamaan mittausväliä. Järjestelmään voisi myös lisätä mahdollisuuden tarkastella lämpötilaa tietyinä päivinä tai tietyllä aikavälillä. Järjestelmän ohjelmistot myös suunniteltiin pitäen mielessä mahdollisuus lisätä järjestelmään uusia erilaisia sensoreita, jos tulevaisuudessa tulee tarvetta laajentaa järjestelmää.

## LÄHTEET

- [1] 6 Working with the I2C Bus, URL: <https://docs.oracle.com/javame/8.0/me-dev-guide/i2c.htm> (viitattu 10. 08. 2020).
- [2] M. Aamir, K. A. Kalwar ja S. Mekhilef, Uninterruptible power supply (UPS) system, *Renewable and sustainable energy reviews* 58 (2016), 1395–1410, URL: <http://www.sciencedirect.com/science/article/pii/S1364032115017189> (viitattu 03. 09. 2020).
- [3] Amazon RDS Backup and Restore, URL: <https://aws.amazon.com/rds/features/backup/> (viitattu 12. 09. 2020).
- [4] R. Anderson, Prevent Cross-Site Scripting (XSS) in ASP.NET Core (2018), URL: <https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-3.1> (viitattu 27. 09. 2020).
- [5] APC Back-UPS BX 1400U-GR 1400VA - UPS toimistoon, URL: <https://www.verkkokauppa.com/fi/product/65199> (viitattu 09. 09. 2020).
- [6] APC Back-UPS Pro 550 UPS, URL: <https://www.verkkokauppa.com/fi/product/21564> (viitattu 09. 09. 2020).
- [7] Azure Backup, URL: <https://azure.microsoft.com/en-us/services/backup/#overview> (viitattu 12. 09. 2020).
- [8] S. Balzer, Contracted persistent object programming, *PhD Workshop, ECOOP*, vol. 12, Citeseer, 2005.
- [9] Banana Pi, URL: <http://www.banana-pi.org/> (viitattu 25. 08. 2020).
- [10] K. Baraka, M. Ghobril, S. Malek, R. Kanj ja A. Kayssi, Low Cost Arduino/Android-Based Energy-Efficient Home Automation System with Smart Task Scheduling, *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, 2013, 296–301.
- [11] B. Batbold, Handling Authentication and Authorization with Node (2019), URL: <https://medium.com/quick-code/handling-authentication-and-authorization-with-node-7f9548fedde8> (viitattu 12. 09. 2020).
- [12] B. Carlson, Queries, Parameterized query, URL: <https://node-postgres.com/features/queries> (viitattu 03. 01. 2020).
- [13] M. contributors, Server-side web frameworks (2020), URL: [https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\\_steps/Web\\_frameworks](https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks) (viitattu 30. 08. 2020).
- [14] *DIGITAL TEMPERATURE SENSOR WITH TCN75A*, URL: [http://robomaa.fi/digitaalinen-lmptila-anturi-tn75a?manufacturer\\_id=15](http://robomaa.fi/digitaalinen-lmptila-anturi-tn75a?manufacturer_id=15) (viitattu 07. 09. 2020).
- [15] DY.fi, URL: <https://www.dy.fi/> (viitattu 18. 06. 2020).

- [16] H. Fang ja K. Fang, The Design of Remote Embedded Monitoring System Based on Internet, *2010 International Conference on Measuring Technology and Mechatronics Automation*, vol. 3, 2010, 852–854.
- [17] Frequently Asked Questions, Arduino Software, URL: <https://www.arduino.cc/en/main/FAQ> (viitattu 31.08.2020).
- [18] GPIO Documentation, URL: <https://www.raspberrypi.org/documentation/usage/gpio/> (viitattu 25.08.2020).
- [19] W. G. Halfond, J. Viegas, A. Orso et al., A classification of SQL-injection attacks and countermeasures, *Proceedings of the IEEE international symposium on secure software engineering*, vol. 1, 2006, 13–15, URL: <https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf> (viitattu 05.01.2020).
- [20] How to change network connection priority in Windows 10, URL: <http://ecross.mvps.org/howto/change-network-connection-priority-in-windows-10.htm> (viitattu 01.09.2020).
- [21] How to change Wi-Fi network connections priority order on Windows 10, URL: <https://www.windowscentral.com/how-change-wi-fi-network-connections-priority-order-windows-10> (viitattu 01.09.2020).
- [22] HOWSTUFFWORKS.COM, How are voltage surges and spikes different?, URL: <https://science.howstuffworks.com/voltage-surges-spikes-different.htm> (viitattu 15.07.2020).
- [23] HTTP authentication with PHP, URL: <https://www.php.net/manual/en/features.http-auth.php#features.http-auth> (viitattu 14.09.2020).
- [24] D. Huluka ja O. Popov, Root cause analysis of session management and broken authentication vulnerabilities, eng, *World Congress on Internet Security (WorldCIS-2012)*, IEEE, 2012, 82–86, ISBN: 9781467311083.
- [25] F. Hutcheson, How to Set Up a Static IP Address from Your Router (2011), URL: <http://blog.dlink.com/mastering-static-ip-addresses-2/> (viitattu 18.06.2020).
- [26] i2c-bus, URL: <https://www.npmjs.com/package/i2c-bus> (viitattu 12.09.2020).
- [27] IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, *IEEE Std 610* (1991), 1–217.
- [28] ifmetric, URL: <http://manpages.ubuntu.com/manpages/xenial/man8/ifmetric.8.html> (viitattu 01.09.2020).
- [29] Install ownCloud Desktop Client on Debian 10 Buster, URL: <https://kifarunix.com/install-owncloud-desktop-client-on-debian-10-buster/> (viitattu 27.09.2020).
- [30] Interfacing with I2C Devices, URL: [https://elinux.org/Interfacing\\_with\\_I2C\\_Devices](https://elinux.org/Interfacing_with_I2C_Devices) (viitattu 12.09.2020).
- [31] M. Koshti, S. Ganorkar ja L. Chiari, IoT Based Health Monitoring System by Using Raspberry Pi and ECG Signal, *International Journal of Innovative Research in Science, Engineering and Technology* 5.5 (2016), 8977–8985.
- [32] A. Kviesis ja A. Zacepins, System architectures for real-time bee colony temperature monitoring, *Procedia Computer Science* 43 (2015), 86–94.

- [33] Libraries, URL: <https://www.arduino.cc/reference/en/libraries/> (viitattu 10.07.2020).
- [34] D. B. Little, *Digital data integrity : the evolution from passive protection to active management*, eng, Chichester, England ; URL: [https://andor.tuni.fi/permalink/358FIN\\_TAMPO/1j3mh4m/alma9910639185805973](https://andor.tuni.fi/permalink/358FIN_TAMPO/1j3mh4m/alma9910639185805973) (viitattu 12.09.2020).
- [35] H. Mansor, M. H. A. Shukor, S. S. Meskam, N. Q. A. M. Rusli ja N. S. Zamery, Body temperature measurement for remote health monitoring system, *2013 IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, 2013, 1–5.
- [36] H. Mansor, M. H. A. Shukor, S. S. Meskam, N. Q. A. M. Rusli ja N. S. Zamery, Body temperature measurement for remote health monitoring system, *2013 IEEE International conference on smart instrumentation, measurement and applications (ICSIMA)*, IEEE, 2013, 1–5.
- [37] Minimum hardware requirements, 2017, URL: <https://docs.microsoft.com/en-us/windows-hardware/design/minimum/minimum-hardware-requirements-overview> (viitattu 20.08.2020).
- [38] W. J. Noonan, *Hardening network infrastructure*, eng, New York, 2004, URL: [https://andor.tuni.fi/permalink/358FIN\\_TAMPO/1j3mh4m/alma9911114544505973](https://andor.tuni.fi/permalink/358FIN_TAMPO/1j3mh4m/alma9911114544505973) (viitattu 14.09.2020).
- [39] Orange Pi, URL: <http://www.orangepi.org/OrangePiPrime/> (viitattu 25.08.2020).
- [40] PHP Readfile, URL: <https://www.php.net/manual/en/function.readfile.php> (viitattu 14.09.2020).
- [41] `phpocman`, Arduino - Web Serial Plotter, URL: <https://www.hackster.io/phpoc-man/arduino-web-serial-plotter-f2c751> (viitattu 01.08.2020).
- [42] Power Supply, URL: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md> (viitattu 24.09.2020).
- [43] J. Prehti, Jondelele / KandiNodejsProjekti (2020), URL: <https://github.com/Jondelele/KandiNodejsProjekti> (viitattu 11.10.2020).
- [44] Raspberry Pi 3 Model B, URL: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/> (viitattu 18.08.2020).
- [45] Raspberry Pi OS, URL: <https://www.raspberrypi.org/documentation/raspbian> (viitattu 14.09.2020).
- [46] Rock 64, URL: <https://www.pine64.org/devices/single-board-computers/rock64/> (viitattu 25.08.2020).
- [47] RUT240, URL: <https://teltonika-networks.com/product/rut240/> (viitattu 12.09.2020).
- [48] R. Shete ja S. Agrawal, IoT based urban climate monitoring using Raspberry Pi, *2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, 2008–2012.
- [49] N. Smithline, A. Stock ja T. Gigler, OWASP Top Ten, English (2019), URL: <https://owasp.org/www-project-top-ten/> (viitattu 25.02.2020).

- [50] Spring, URL: <https://spring.io/> (viitattu 03.09.2020).
- [51] Spring Security, URL: <https://spring.io/projects/spring-security> (viitattu 03.09.2020).
- [52] TCN75A Datasheet, URL: [https://www.microbot.it/documents/mr003-001\\_datasheet.pdf](https://www.microbot.it/documents/mr003-001_datasheet.pdf) (viitattu 07.09.2020).
- [53] TEMPer Gold "Original"USB Temperature Sensor, URL: <https://thepihut.com/products/temper-gold-original-usb-temperature-sensor> (viitattu 19.08.2020).
- [54] The Effects of Power Outages on Computers Electronics (2018), URL: <https://pennyelectric.com/blog/the-effects-of-power-outages-on-computers-electronics/> (viitattu 15.07.2020).
- [55] Ubuntu Server Guide, 2020, URL: <https://assets.ubuntu.com/v1/d71537c3-ubuntu-server-guide.pdf> (viitattu 22.08.2020).
- [56] User authentication in Django, URL: <https://docs.djangoproject.com/en/3.1/topics/auth/#user-authentication-in-django> (viitattu 12.09.2020).
- [57] Using the Synchronization Client, URL: <https://doc.owncloud.org/desktop/2.0/navigating.html#using-the-synchronization-client> (viitattu 12.09.2020).
- [58] J. Waltzer, Tietoturvallinen WWW-Ohjelmointi, Suomi (2012), URL: [https://www.theseus.fi/bitstream/handle/10024/41564/Waltzer\\_Jaakko.pdf?sequence=1&isAllowed=y](https://www.theseus.fi/bitstream/handle/10024/41564/Waltzer_Jaakko.pdf?sequence=1&isAllowed=y) (viitattu 25.02.2020).
- [59] J. Watmore, NodeJS - Basic Authentication Tutorial with Example API (2018), URL: <https://jasonwatmore.com/post/2018/09/24/nodejs-basic-authentication-tutorial-with-example-api> (viitattu 12.09.2020).
- [60] Web server security – Part 1: Basic hardening, URL: <https://infosec-handbook.eu/blog/wss1-basic-hardening/> (viitattu 14.09.2020).
- [61] What is Information Security?, URL: <https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html> (viitattu 29.08.2020).
- [62] S. WHO headquarters Geneva, Temperature and humidity monitoring systems for fixed storage areas, URL: <https://www.gmp-compliance.org/guidelines/gmp-guideline/who-supplement-6-temperature-and-humidity-monitoring-systems-for-fixed-storage-areas> (viitattu 05.08.2020).
- [63] Windows 10 system requirements, 2019, URL: <https://support.microsoft.com/en-us/help/4028142/windows-10-system-requirements> (viitattu 20.08.2020).
- [64] Ximin Zhang, Junding Sun ja Lihua Zhou, Development of an Internet Home Automation System using Java and Dynamic DNS Service, *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, 2005, 537–539.
- [65] Yocto-Temperature, URL: <http://www.yoctopuce.com/EN/products/usb-environmental-sensors/yocto-temperature> (viitattu 10.09.2020).
- [66] Yocto-Temperature User's guide (2020), URL: <http://www.yoctopuce.com/projects/yoctotemp/TMPSENS1.usermanual-EN.pdf>.