RASPBERRY PI -LAITTEEN ENERGIANKULUTUKSEN MINIMOINTI OHJELMALLISEN KONFIGURAATION AVULLA

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tietotekniikan kandidaatintutkielma

2024

Jaakko Pyrhönen

Tarkastaja: Tutkijaopettaja Jouni Ikonen

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT insinööritieteiden tiedekunta

Tietotekniikka

Jaakko Pyrhönen

Raspberry Pi -laitteen energiankulutuksen minimointi ohjelmallisen konfiguraation avulla

Tietotekniikan kandidaatintyö

2024

43 sivua, 5 kuvaa ja 4 taulukkoa

Tarkastaja: Tutkijaopettaja Jouni Ikonen

Avainsanat: IoT, esineiden Internet, Raspberry Pi, teho, energia, minimointi

Monet IoT-laitteet toimivat akkujen varassa sellaisissa ympäristöissä, joissa akkujen säännöllinen lataaminen ei ole käytännöllistä. Vastaavien laitteiden toiminta-ajan maksimointi on siis toivottavaa. Raspberry Pi on suosittu yhden piirilevyn tietokone, jolla voidaan sanoa olevan monta IoT-sovelluksiin soveltuvan laitteen tyypillistä piirrettä. Tämä työ tutkii millaisia käytännössä sovellettavia menetelmiä tai periaatteita akateeminen kirjallisuus käsittelee Raspberry Pi -mallien 3B+, 4B tai 5 energiankulutuksen minimoimiseksi ja millaisia vaikutuksia näitä menetelmiä tai periaatteita soveltamalla voidaan havaita mittauskokeiden yhteydessä. Tulosten perusteella tiettyjä laiteominaisuuksia pois kytkemällä Raspberry Pi -mallien tyhjäkäyntitilassa käyttämän tehon keskiarvoa voidaan laskea 66.4 %, 25.6 %, ja 11.2 % malleilla 3B+, 4B ja 5. Nostamalla prosessorin kellotaajuutta matalimmasta mahdollisesta taajuudesta korkeimpaan mahdolliseen taajuuteen voidaan havaita muita asetuksia muokkaamatta tyhjäkäyntitilassa 8.3 %, 27.2 % ja 36.8 % kasvua käytetyn tehon keskiarvossa malleilla 3B+, 4B ja 5. Mittauskokeiden lisäksi työ esittelee viitekehyksen energiankulutuksen mittaamiseksi.

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Software Engineering

Jaakko Pyrhönen

Minimizing the power consumption of Raspberry Pi via programmatic configuration

Bachelor’s thesis

2024

43 pages, 5 figures, and 4 tables

Examiner: Associate professor Jouni Ikonen

Keywords: IoT, Internet of Things, Raspberry Pi, power, energy, minimizing

Many IoT devices rely on batteries and operate in environments where regular charging or swapping of batteries is impractical. Therefore, maximizing the operational time of such devices is desirable. Raspberry Pi is a popular single-board computer which can be said to possess many of the qualities that a typical device used for IoT applications exhibits. This work studies what kind of applicable methods or principles can be found in academic literature for minimizing the power consumption of the Raspberry Pi models 3B+, 4B, and 5, and what kind of effects regarding power consumption can be measured when performing experiments utilizing these methods or principles. The results of this work show that by disaling certain device features the idle state mean power consumption of the models 3B+, 4B, and 5 can be lowered by 66.4 %, 25.6 %, and 11.2 %, respectively. By increasing the processor clock frequency from the lowest possible frequency to the highest possible frequency, without modification of other settings, increases of 8.3 %, 27.2 % and 36.8 % can be observed in the mean power consumption of the idle state for the models 3B+, 4B, and 5, respectively. In addition, this work presents a framework for measuring energy consumption.

KIITOKSET/ ACKNOWLEDGEMENTS

Kiitos työn ohjaajalle Jouni Ikoselle arvokkaista neuvoista ja palautteesta.

SYMBOLI- JA LYHENNELUETTELO

Lyhenteet

u	hetkellinen jännite, V
U	tehollinen jännite, V
C	kapasitanssi, F
E	Energia, J
f	taajuus, Hz
i	hetkellinen virta, A
p	hetkellinen teho, W
P	keskimääräinen teho, W
Q	sähkövaraus, Ah
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DVFS	Dynamic Voltage and Frequency Scaling
GPU	Graphics Processing Unit
I/O	Input/Output
IP	Internet Protocol
ICT	Information and Communication Technology
IoT	Internet of Things
RFCOMM	Radio Frequency Communication
RPi	Raspberry Pi
RTC	Real-Time Clock
SDRAM	Synchronous Dynamic Random Access Memory

SSH Secure Shell

Sisällysluettelo

Tiivistelmä

Abstract

Kiitokset

Symboli- ja lyhenneluettelo

1 Johdanto....11

1.1 Tausta....11

1.2 Tutkimuskysymykset....13

1.3 Työn rakenne....13

2 Ratkaisumenetelmät ja kirjallisuuskatsaus....15

2.1 Ratkaisumenetelmät....15

2.2 Kirjallisuuskatsaus....16

3 Energiankulutuksen minimoimisen keinoja....19

3.1 Dynaaminen jännitteen ja kellotaajuuden säätely....20

3.2 Laiteominaisuuksien käytöstä poistaminen....22

3.3 Ohjelmistojen konfigurointi....22

3.4 Käyttöjaksottaminen ja lepotilan hyödyntäminen....23

3.5 Verkkoliikenteen optimointi....24

4 Järjestelmä energiankulutuksen mittaamiseksi....25

4.1 Vaatimukset....25

4.2 Mittausjärjestelmän komponentit....26

5 Kokeet ja tulokset....29

5.1 Mittarin Bluetooth-yhteyden vaikutus energiankulutukseen....30

5.2 SSH-yhteyden ja verkonkäytön vaikutus....31

5.3 Laiteominaisuuksien käytöstä poistaminen....34

5.4 Kellotaajuuden säätely....38

6 Johtopäätökset....40

Lähteet...41

Kuvaluettelo

Kuva 1: Kirjallisuuskatsausprosessi ja lähteet, joissa käsitellään energiankulutuksen minimoimisen keinoja

Kuva 2: Mittausjärjestelmä

Kuva 3: Eri aikavälein SSH-yhteyden kautta lähetettyjen viestien vaikutus RPi:n käyttämään tehoon

Kuva 4: Laiteominaisuuksien käyttöönoton vaikutus tehoon

Kuva 5: Tyhjäkäyntitilan tehonkäyttö suhteessa prosessorin kellotaajuuteen, kun WiFi ja Bluetooth ovat käytössä ja pois käytöstä

Taulukkoluettelo

Taulukko 1: Lähteissä käsiteltyjä energiankulutukseen vaikuttavia tekijöitä kategorisoituna viiteen eri luokkaan

Taulukko 2: UM25C-mittarin jännite- ja virtamittausominaisuudet

Taulukko 3: Raspberry Pi -mallien tiedot

Taulukko 4: Laiteominaisuuksien käytöstä poistamisen menetelmiä

1 Johdanto

1.1 Tausta

Sähköenergian kulutus maailmalla kasvaa edelleen vuosi vuodelta. International Energy Agency (IEA) (2023) raportoi sähköenergian globaalin kysynnän kasvaneen vuonna 2022 lähes 2 %:lla, ja vuotuisen kysynnän kasvun voidaan IEA:n mukaan odottaa nousevan noin 3.2 %:iin vuosien 2024–2025 välillä. Globaali kysynnän kasvu on merkittävää siitä huolimatta, että Euroopan unionin alueella kysyntä laski IEA:n mukaan 3.5 %:lla vuonna 2022. Globaalista kasvavasta sähköenergian kulutuksesta merkittävä osa kuuluu ICT-sektorille (Information and Communication Technology), jossa mm. esineiden Internet, eli englanniksi Internet of Things (IoT), on nopeassa kasvussa oleva ala. Tästä eteenpäin esineiden Internettiin viitataan lyhenteellä IoT.

IoT voidaan määritellä joukoksi älykkäitä tunnistettavissa olevia objekteja, jotka kykenevät kommunikoimaan keskenään muodostaen verkkoja toisiinsa kytkettyjä asioita tai kommunikoimaan loppukäyttäjien tai muiden verkkoon kytkettyjen entiteettien kanssa (Miorandi et al., 2012). IoT Analytics –raportin (Satyajit, 2023) mukaan vuonna 2022 IoT-verkkoihin yhdistettyjen laitteiden määrä maailmalla kasvoi 14.3 miljardiin, mikä tarkoittaa 18 %:n kasvua vuoden 2022 alkuun verrattuna, ja vuonna 2023 yhteyksien määrän voidaan odottaa kasvaneen noin 16 %:lla.

IoT-laitteet ovat usein edullisia ja helposti saatavilla olevia laitteita. Eräs näitä piirteitä edustava laite on Raspberry Pi (RPi), joka on suurta suosiota kerännyt yhden piirilevyn tietokone, ja joka soveltuu monenlaisiin IoT-sovelluksiin, kuten älykkäisiin maatalous-, terveydenhuolto- tai kotitalousratkaisuihin (Hosny et al., 2023). Bekaroon ja Santokheen (2016) kokeiden perusteella RPi:n toisen sukupolven B-malli ottaa tyypillisessä käytössä, kuten videoita katsellessa, web-sivuja selatessa tai musiikkia kuunnellessa, keskimäärin

noin 2-3W tehoa. Laitteen kuluttama energia E saadaan integroimalla hetkellinen teho $p(t)$ tarkastelujakson yli

$$E = \int_0^T p(t)dt, \tag{1}$$

ja hetkellinen sähköteho saadaan hetkellisen jännitteen $u(t)$ ja hetkellisen virran $i(t)$ tulona

$$p(t) = i(t)u(t) \tag{2}$$

RPi on kuitenkin varsin monitoiminen laite, joten jos asian havainnollistamiseksi oletetaan, että tyypillinen IoT-laite käyttää tasaisesti vaikkapa noin 1W tehoa (mikä saattaa silti olla suuri kulutus useille IoT-laitteille) ja että maailmassa on 16 miljardia vastaavaa IoT-laitetta aktiivisessa käytössä (mikä vastaisi noin 16 %:n kasvua määrässä vuoden 2022 arvellusta määrästä), tämä tarkoittaisi yhtälön (1) mukaisesti, kun integroidaan vuoden 8760:n tunnin yli ja kerrotaan tulos laitteiden lukumäärällä $N_{\mathrm{IoT}} = 16\times10^9$, että IoT-laitteet kuluttaisivat vuodessa yhteensä noin 140 TWh energiaa. IEA:n vuoden 2023 raportin kuvaajan mukaan maailman sähköenergian kysyntä vuonna 2023 oli noin 27000 TWh, joten tämän karkean esimerkin perusteella IoT-laitteet olisivat kattaneet noin 0.5 % koko maailman sähköenergiasta vuonna 2023.

IoT-laitteiden energiankulutuksella on siis havaittavissa oleva vaikutus globaaliin sähköenergian kulutukseen, mutta tämän lisäksi, ja tämän työn motivaation suhteen tärkeämpänä havaintona, energiankulutuksella on tärkeä osuus IoT-sovelluksien yleisen toimivuuden ja kannattavuuden kannalta. Työn motivaatio syntyi Ikonen et al. (2022) tutkimuksesta, jossa arvioitiin Sigfox-verkossa tehtävien lähetyksien viivettä (eng. end-to-end delay) lähettimen ja palvelimen välillä, ja syvällisempi ymmärrys mobiilin mittausaseman energiankulutuksesta ilmeni tarpeelliseksi. Voidaan olettaa, että merkittävä osa IoT-laitteista operoi akkujensa tai paristojensa varassa sellaisissa ympäristöissä, joissa akkujen säännöllinen lataaminen tai paristojen säännöllinen vaihtaminen ei ole käytännöllistä. Luonnollisesti on siis toivottavaa, että tällaisten laitteiden luotettava toiminta-aika yhdellä latauksella tai yhdellä paristonvaihdolla olisi mahdollisimman pitkä. Alhainen laitteen käyttämä teho tarkoittaa yleensä myös pitempää akun kestoa, joten yleinen energiankulutuksen minimointi voidaan nähdä kriittisenä osana tätä tavoitetta.

Näiden harkintojen pohjalta on siis selvää, että IoT-tarkoituksiin soveltuvien laitteiden, kuten RPi:n, energiankulutuksen minimointi on ajankohtainen ja tärkeä aihe. Tässä työssä

tutkitaan kuinka RPi:n energiankulutusta voidaan minimoida ja minkälaisia vaikutuksia aiemmista tutkimuksista löytyneitä keinoja tai periaatteita soveltamalla mittauksia tehdessä havaitaan energiankulutuksen suhteen. Työtä varten toteutetun kirjallisuuskatsauksen perusteella voidaan todeta IoT-laitteiden energiankulutuksen minimoimisen olevan laaja aihealue, jota on jo aiemmin tutkittu useasta näkökulmasta. Tässä tapauksessa aiheen laajuuteen vaikuttaa myös se, että Linux-käyttöjärjestelmän energiankulutuksen optimointiin liittyvät tutkimukset on nähty aiheen kannalta relevantteina, sillä mm. RPi:n virallinen käyttöjärjestelmä, Raspberry Pi OS, on Linux-pohjainen (Raspberry Pi, 2024).

1.2 Tutkimuskysymykset

Työn tavoitteena on vastata kahteen tutkimuskysymykseen:

1. Millaisia keinoja tai periaatteita kirjallisuudessa käsitellään Raspberry Pi:n energiankulutuksen minimoimiseksi?
2. Millaisia vaikutuksia energiankulutuksen suhteen havaitaan, kun sovelletaan löydettyjä keinoja tai periaatteita ja tehdään mittauksia Raspberry Pi:llä?

Työtä rajataan keskittymällä ohjelmallisesti toteutettaviin konfiguraatioihin, kuten käyttöjärjestelmän asetusten muokkaamiseen, ja vähemmän tiettyihin sovelluksiin kohdistuviin keinoihin kuten koodin optimointiin. Lisäksi, RPi:n käyttöjärjestelmien suhteen työssä keskitytään ainoastaan Linux-pohjaisiin järjestelmiin.

1.3 Työn rakenne

Kappaleessa 2 esitellään lyhyesti työssä käytettyjä lähteitä ja kuinka kirjallisuuskatsaus toteutettiin. Kappaleessa 3 kuvaillaan yleisesti ja tarkemmin kirjallisuudesta löytyneitä menetelmiä tai periaatteita RPi:n ja yleisesti yksittäisten IoT-laitteiden energiankulutuksen minimoimiseksi. Kappaleessa 4 kuvataan energiankulutuksen mittausta varten laaditun järjestelmän vaatimuksia ja toteutusta. Kappaleessa 5 esitellään toteutetut mittauskokeet ja analysoidaan niiden tuloksia. Johtopäätökset ovat kappaleessa 6 .

2 Ratkaisumenetelmät ja kirjallisuuskatsaus

Tässä kappaleessa esitellään kirjallisuuskatsauksen prosessi sekä sen myötä löytyneet lähteet. Kappale 2.1 kuvailee prosessia, jolla kirjallisuuskatsaus tehtiin. Lähteisiin tutustutaan kappaleessa 2.2 .

2.1 Ratkaisumenetelmät

Kirjallisuuskatsauksen tavoitteena oli löytää käytännössä sovellettavia menetelmiä tai periaatteita, joilla RPi:n energiankulutusta voidaan laskea. Kirjallisuuskatsausprosessi toteutettiin pääosin Google Scholar -hakukoneen avulla. Kuva 1 esittelee energiankulutuksen minimoimisen suhteen sovellettavia keinoja tai periaatteita käsittelevien lähteiden määrän sekä prosessin, jonka myötä nämä lähteet löydettiin. Seuraavaksi kuvattavat vaiheet koskevat ainoastaan kyseistä prosessia:

1. Haulla ’("Raspberry Pi" OR Linux) AND ((energy OR power) AND (efficiency OR consumption))’ Google Scholar löysi 365 000 osumaa. Tulokset järjesteltiin Google Scholarin “sort by relevance”-optiolla.

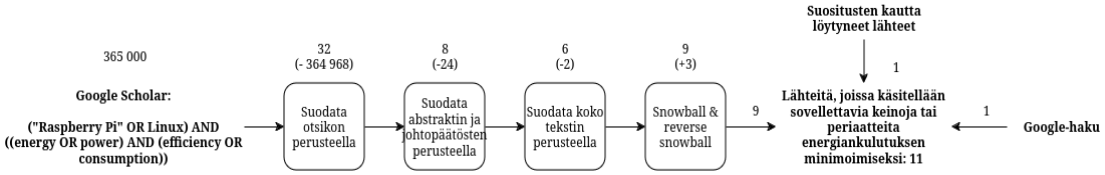
2. Hakutuloksesta valikoitiin ensimmäiseltä sivulta alkaen julkaisuja otsikoiden perusteella, kunnes otsikot eivät enää vaikuttaneet työhön liittyviltä. Tässä vaiheessa pelkästään otsikoiden pohjalta valikoitui 32 julkaisua ja otsikoita luettiin ensimmäiseltä sivulta alkaen läpi 80 kappaletta (8 sivua).

3. Kustakin vaiheessa 2 valituista julkaisusta, joihin saatiin pääsy, luettiin abstrakti ja johtopäätökset. Jos nämä vaikuttivat työn kannalta mielenkiintoisilta, julkaisu valittiin seuraavaa vaihetta varten. Tämän vaiheen jälkeen jäljelle jäi 8 julkaisua.

4. Valikoidut julkaisut luettiin kokonaan läpi. Jos teksti ei vaikuttanut työhön liittyvältä, julkaisu jätettiin hyödyntämättä. Tämän vaiheen jälkeen jäljelle jäi 6 julkaisua.

5. Jäljelle jääneistä töistä etsittiin lähdeluettelon kautta lisää lupaavalta vaikuttavia julkaisuja (snowballing). Lisäksi sovellettiin Google Scholarin “cited by”-toimintoa etsimään julkaisuja, jotka ovat viitanneet valikoituihin töihin (reverse snowballing). Tämän vaiheen myötä löytyi 3 julkaisua lisää.

Yllä kuvatun prosessin tuloksena löydettiin 9 lähdettä, joissa käsiteltiin tämän työn rajausten mukaisesti sovellettavia keinoja tai periaatteita RPi:n energiankulutuksen minimoimiseksi. Lisäksi, yksi vastaava lähde löytyi suosituksen kautta ja toinen ns. tavallisen Google-haun kautta.



Kuva 1. Kirjallisuuskatsausprosessi ja lähteet, joissa käsitellään energiankulutuksen minimoimisen keinoja

2.2 Kirjallisuuskatsaus

Tässä kappaleessa esitellään kappaleessa 2.1 kuvatun kirjallisuuskatsausprosessin myötä löytyneet lähteet tarkemmin. Yleisesti ottaen lähteistä löytyneet keinot energiankulutuksen minimoimiseksi voidaan luokitella seuraaviin kategorioihin: dynaaminen jännitteen ja kellotaajuuden säätely (DVFS), laiteominaisuuksien käytöstä poisto, ohjelmistojen konfigurointi, käyttöjaksottaminen ja lepotilan hyödyntäminen sekä verkkoliikenteen optimointi. Näihin kategorioihin ja niiden merkityksiin tutustutaan tarkemmin vielä kappaleessa 3 .

CPU:n kellotaajuuden säätely, tai läheiset teemat kuten CPU:n käyttötason (osuus ajasta, jona prosessori ei ole tyhjäkäyntitilassa) vaikutus energiankulutukseen, olivat työtä varten analysoidussa kirjallisuudessa suhteellisen usein mainittuja aiheita. Astudillo-Salinas et al. (2016) tutkivat mm. kuinka CPU:n, SDRAM:n ja GPU:n kellotaajuuksien laskemisella voidaan vaikuttaa RPi:n käyttämään tehoon. Kadota et al. (2020) vertasivat Rpi 3 B:n käyttämää energiaa konenäkösovelluksia suorittaessa, kun CPU:n kellotaajuutta vaihdeltiin. Kaup et al. (2014) loivat matemaattisen mallin RPi:n käyttämälle teholle ja mallissa CPU:n käyttötasolla oli merkitys käytettyyn tehoon. Karpowicz (2016) toteutti

CPU:n kellotaajuudensäätelyalgoritmin Linux-järjestelmälle, jonka ansiosta suorituskyky parani ja käytetty teho laski realistista koetta suorittaessa verrattuna Linux-järjestelmän muihin kellotaajuudensäätelyalgoritmeihin. Lambert et al. (2021) tutkivat mm. CPU:n kellotaajuuden vaikutusta RPi:n 4B-mallin käyttämään tehoon. Liang ja Lai (2010) kehittivät CPU-kellotaajuuden säätöä varten algoritmin Android-järjestelmälle, joka otti huomioon myös mm. muistinkäsittelyn määrän.

Valikoituneista lähteistä kolme käsitteli jonkinlaista laitteiden, laiteohjainten tai vastaavien ominaisuuksien käytöstä poistamisen vaikutusta energiankulutukseen. Astudillo-Salinas et al. (2016) arvioivat HDMI-ohjaimen ja Ethernetin käytöstä poistamisen vaikutusta RPi:n energiankulutukseen. Bekaroo ja Santokhee (2016) mainitsivat oheislaitteiden irrottamisen olevan mahdollisesti eduksi energiankulutuksen laskemiseksi, mutta he eivät toteuttaneet mittauksia tämän arvioimiseksi. Geerlingin (2021) blogikirjoituksessa mainittiin CPU-ytimien disabloinnin laskevan RPi Zero 2W -mallin käyttämää tehoa.

Ohjelmistojen konfiguroinnin aiheet olivat arvioiduissa lähteissä usein esillä. Astudillo-Salinas et al. (2016) arvioivat eri käyttöjärjestelmien hyödyntämisen vaikutusta RPi:n energiankulutukseen. Bekaroon ja Santokheen (2016) mukaan järjestelmän taustaprosessien (eng. daemon process) määrän vähentäminen tai optimointi voisi olla eduksi energiankulutuksen laskemisen suhteen. Kadotan et al. (2020) tutkimus osoittaa, että säietason rinnakkaisuutta säätelemällä voidaan vaikuttaa ohjelman suorituksen aikana kuluneeseen energiaan. Lambert et al. (2021) arvioivat eroa ns. ”bare-metal”-tilan ja käyttöjärjestelmällisen tilan käyttämässä tehossa RPi:lle. Shizukuishi ja Matsubara (2020) saivat laskettua Linux-järjestelmän käyttämää muistia ja tehoa poistamalla Linux-ytimestä (eng. kernel) ominaisuuksia, vaikka vain muistinkäytön lasku oli tilastollisesti merkitsevää.

Käyttöjaksottaminen (eng. duty cycling) viittaa tässä asiayhteydessä käytäntöön, jossa laitetta käytetään säännöllisin väliajoin aktiivisen ja inaktiivisen tilan välillä kokonaisenergiankulutuksen minimoimiseksi, ja siitä on muodostunut peruskeino IoT-verkkojen solmujen toiminnallisen ajan pidentämiseksi (Singh, et al., 2022). Amirtharaj et al. (2020) tutkivat kuinka RPi:n käynnistymis- ja sammumisaikaa saataisiin nopeutettua käyttöjaksottamisen tehostamiseksi. Sigfox-solmun toiminnallista aikaa saatiin pidennettyä merkittävästi Yamazakin ja Nakajiman (2023) tutkimuksessa pienentämällä lepotilan virrankäyttöä, mutta, koska kyseessä ei ollut tutkimuksen pääaihe, menetelmiä tämän saavuttamiseksi ei kuvailtu tarkemmin.

Viimeinen kategoria, eli verkkoliikenteen optimointi, on jo IoT-verkkojen määritelmän perusteella merkittävä tekijä IoT-solmujen energiankulutuksen suhteen. Jo pelkkä verkkokommunikaation mahdollistavan laitteiston ylläpito voi vaatia huomattavasti tehoa, sillä mm. Astudillo-Salinas et al. (2016) havaitsivat merkittävän laskun käytetyssä tehossa Ethernetin pois kytkemisen jälkeen, vaikka verkkoyhteyttä ei käytetty aktiivisesti. Bekaroo ja Santokhee (2016) mainitsivat verkkoyhteyttä käyttävien tehtävien vaativan keskimäärin enemmän tehoa offline-tilassa toteutettaviin tehtäviin nähden. Kaupin et al. (2014) laatimassa mallissa Ethernet- ja WiFi-rajapintojen kautta tapahtuva datasiirto vaikuttaa huomattavasti käytettyyn tehoon. Yamazakin ja Nakajiman (2023) mallin mukaan lähetteiden koko (tavuina) ja päivittäinen lähetemäärä vaikuttavat akuilla toimivan IoT-solmun toiminta-aikaan.

Aiemmat tutkimukset ovat kirjallisuuskatsauksen perusteella keskittyneet pääosin RPi:n vanhempiin malleihin tai kokonaan eri laitteisiin. Yksikään lähteistä ei harkinnut Rpi:n mallia 5 ja vain yksi käytti mallia 4B. Tässä työssä sovelletaan kirjallisuuskatsauksen perusteella tärkeimmiksi todettuja aiempien tutkimusten löytämiä keinoja tai periaatteita energiankulutuksen minimoimiseksi sekä toteutetaan mittauksia RPi:n malleilla 3B+, 4B ja 5 näiden menetelmien vaikutuksien arvioimiseksi.

3 Energiankulutuksen minimoimisen keinoja

Kirjallisuuskatsauksen perusteella RPi:n kaltaisten yksittäisten laitteiden energiankulutuksen minimoimisen keinoja tutkiessa muutama teema nousee esille usein. Tässä kappaleessa näitä aihealueita kuvaillaan tarkemmin. Taulukkoon 1 on muodostettu viisi energiankulutuksen minimoimiseen liittyvää kategoriaa, joiden alle kappaleessa 2.2 esiteltyjen lähteiden käsittelemiä aiheita on koottu. Tässä osiossa ei vielä käsitellä käytännön ohjeita tai toimenpiteitä keinojen soveltamiseksi, vaan menetelmien tarkempi esittely tehdään kappaleessa 5 mittauskokeisiin tutustumisen yhteydessä. On syytä mainita, että taulukossa 1 esiintyvät lähteet käsittelevät erilaisia laitteita ja erilaisia käyttöjärjestelmiä, joten kaikki taulukoidut keinot eivät välttämättä ole toimivia tässä työssä käsiteltävien laitteiden ja käyttöjärjestelmien kanssa.

Taulukko 1. Lähteissä käsiteltyjä energiankulutukseen vaikuttavia tekijöitä kategorisoituna viiteen eri luokkaan

Lähde	**Dynaaminen jännitteen ja kellotaajuuden säätely**	**Laiteominaisuuksien käytöstä poistaminen**	**Ohjelmistojen konfigurointi**	**Käyttöjaksottaminen ja lepotilan hyödyntäminen**	**Verkkoliikenteen optimointi**
Amirtharaj et al. (2020)			Systemd-konfigurointi, Pallex-viitekehys	Käynnistymis- ja sammumisajan nopeuttaminen	
Astudillo-Salinas et al. (2016)	CPU:n, SDRAM:n ja GPU:n kellotaajuuksien säätö	HMDI:n ja Ethernetin käytöstä poisto	Vähemmän resursseja vaativa käyttöjärjestelmä		Ethernetin käytöstä poisto
Bekaroo ja Santokhee (2016)		Oheislaitteiden ja laiteohjainten käytöstä poisto	Vähemmän taustaprosesseja	Laitteen sammuttaminen, kun ei käytössä	Offline-tilassa niin usein kuin mahdollista
Kadota et al. (2020)	Eri kellotaajuuksien hyödyntäminen		Rinnakkaisuuden hyödyntäminen		
Kaup et al. (2014)	CPU:n käyttötason				WiFi:n ja Ethernetin

	vaikutuksen arviointi				verkkokaistanleveyksien vaikutuksen arviointi
Karpowicz (2016)	CPU:n kellotaajuudensäätöalgoritmin implementointi				
Lambert et al. (2021)	Eri kellotaajuuksien hyödyntäminen		Käyttöjärjestelmällisen - ja ”bare-metal”-järjestelmän vertailu		
Liang ja Lai (2010)	Ennakoiva kellotaajuuden säätely muistikäsittelyn perusteella				
Shizukuishi ja Matsubara (2020)			Linux-ytimestä tarpeettomien osien tai toimintojen poisto		
Yamazaki ja Nakajima (2023)			Sulautettujen ohjelmistojen optimointi	Lepotilan virrankäytön optimointi	Lähetyskoon ja -määrän vaikutus toiminta-aikaan
Geerling (2021)		CPU-ytimien käytöstä poistaminen			

Kaikki taulukkoon 1 kirjatut menetelmät voidaan ymmärtää sähköenergian yhtälön (1) mukaisesti, ja energiankulutuksen minimoimisen tavoite voidaan tiivistää suurin piirtein näin: sähkötehoa käyttäviä töitä kannattaa olla kunakin hetkenä määrällisesti mahdollisimman vähän (minimoi järjestelmän resurssikäyttö) ja kukin niistä on syytä tehdä mahdollisimman nopeasti (minimoi integrointiaika T) sekä mahdollisimman pienellä teholla (minimoi p), joka puolestaan voidaan minimoida sähkövirran i ja jännitteen u tulon kautta. Jos järjestelmällä ei ole tehtäviä töitä, se kannattaa sammuttaa tai viedä lepotilaan, jotta teho saadaan mahdollisimman alas. Seuraavaksi taulukossa 1 kuvattuja aiheita käsitellään tarkemmin.

3.1 Dynaaminen jännitteen ja kellotaajuuden säätely

Kirjallisuuskatsauksen perusteella DVFS (Dynamic Voltage and Frequency Scaling) oli usein mainittu menetelmä energiankulutuksen laskemiseksi. Le Sueur ja Heiser (2010) antavat hyvän kuvauksen DVFS-tekniikasta, ja heidän työnsä pohjalta tätä tekniikkaa kuvaillaan seuraavaksi tarkemmin. DVFS tarkoittaa tehonhallinnan menetelmää, jossa CPU:n kellotaajuutta lasketaan, mikä mahdollistaa myös syöttöjänniteen (eng. supply voltage) pudotuksen. CMOS-piirien (complementary metal-oxide-semiconductor), eli myös useimpien modernien prosessorien, taajuudensäätelyn myötä saatu vaikutus tehon suhteen määräytyy yhtälön

$$P = CfU^2 + P_{st} \tag{3}$$

perusteella. Yhtälössä C on transistoriporttien kapasitanssi, U on syöttöjännite, jonka suuruus riippuu myös käyttötaajuudesta f, ja P_{st} tarkoittaa piirin vuototehoa eli staattista tehoa (eng. leakage power). Yhtälöstä nähdään, että taajuutta f pudottamalla voidaan laskea tehoa. Taajuuden laskeminen laskee myös vaadittua jännitetasoa ja, koska teho P riippuu osaksi jännitteen neliöstä U^2, tämän myötä voidaan teoriassa saavuttaa merkittävää energiansäästöä. Asia on kuitenkin monimutkainen, koska taajuutta f laskemalla yhtälössä (1) esitetty prosessointitehtävän vaatima aika T kasvaa, koska yleisesti $T \propto 1/f$. Tämän vuoksi taajuutta ei usein ole tavoitteenmukaista suoraan pudottaa aina matalimmalle tasolle, vaan sen säätelyä varten on kehitelty algoritmeja, jotka pyrkivät laskemaan optimaalisen taajuuden eri sääntöjen pohjalta. Lisäksi on huomattava, että nykyään CPU:ssa käytettävien transistorien jännitteet ovat pienempiä verrattuna siihen aikaan kun DVFS ensimmäisen kerran keksittiin, joten DVFS:n mahdolliset hyödyt ovat laskeneet suhteessa vuototehoon P_{st}.

Linux-käyttöjärjestelmän ydin (eng. kernel) toteuttaa CPU:n kellotaajuuden säätelyn toiminnallisuuden CPUFreq-alijärjestelmässään (eng. subsystem), joka mahdollistaa taajuudensäätöalgoritmin vaihtamisen. CPUFreq-järjestelmässä tällaista tajuudeensäätöalgoritmin implementoivaa osaa kutsutaan yleisnimellä säädin (eng. governor). CPUFreq tarjoaa valmiiksi muutaman geneerisen säätimen, joilla on nimet *performance, powersave, userspace, schedutil, ondemand* ja *conservative*. Userspace-säädin on erityinen muihin säätimiin nähden, koska sen avulla käyttäjä voi toteuttaa oman säätöalgoritminsa asettamalla itse haluamiaan kellotaajuuksia. (Wysocki, 2017.)

Kirjallisuuskatsauksen perusteella tutkimuksissa vertailu tehdään usein *ondemand*-säätimeen nähden. Tässä työssä *ondemand* on käytössä, ellei toisin mainita.

Kirjallisuuskatsauksen perusteella aiemmissa tutkimuksissa DVFS:n suhteen on havaittu vaikutuksia, jotka ovat yhteensopivia yllä kuvatun suhteen. Esimerkiksi Astudillo-Salinas et al. (2016) saavuttivat parhaimmillaan 6.8 % laskun tehonkäytön suhteen CPU:n kellotaajuutta laskiessa. Karpowiczin (2016) tutkimus sen sijaan todistaa, että kellotaajuuden säätelyalgoritmeissa voi olla optimoimisen varaa riippuen sovelluskohteesta, sillä työssä implementoidun kellotaajuudensäätöalgoritmin suorituskyky oli testiympäristössä ondemand-säätimeen nähden 16 % parempi ja energiankulutus 3 % matalampi.

3.2 Laiteominaisuuksien käytöstä poistaminen

On selvää, että elektronisten laitteiden käyttö edellyttää sähkötehoa, mikä tarkoittaa, että yleensä niitä käytöstä poistamalla energiankulutusta voidaan laskea. Oletettavasti tämän vuoksi Bekaroo ja Santokhee (2016) mainitsivat tarpeettomien laitteiden kokonaan irti kytkemisen olevan helpoin keino energiankulutuksen pienentämiseksi. Lisäksi, myös laiteohjainten käytössä pitäminen voi kasvattaa käytettyä tehoa. Esimerkiksi Astudillo-Salinas et al. (2016) havaitsivat HDMI-ohjaimen pois kytkemisen madaltavan energiankulutusta noin 8 % verrattuna kokeissa määriteltyyn perustilaan.

3.3 Ohjelmistojen konfigurointi

Tämän työn puitteissa ohjelmistojen konfigurointi viittaa etupäässä tarpeettomien ominaisuuksien poistamiseen resurssikäytön, kuten muistin tai CPU-käyttötason, laskemiseksi, tai ohjelmistojen suoritusnopeuden parantamista mm. rinnakkaisuutta hyödyntämällä prosessointitehtävän ajan minimoimiseksi. Tähän ohjelmiston sovelluskohteen kannalta tarpeettomaan tai liialliseen määrään ominaisuuksia viitataan englannin kielellä sanalla ”bloat”, ja sen lisääntyminen on usein yhteydessä suurempaan energiankulutukseen (Bhattacharya et al., 2011). Tästä eteenpäin vähän resursseja vaativiin

ohjelmistoihin viitataan termillä *kevyt* ja paljon resursseja vaativiin ohjelmistoihin termillä *raskas.*

Aiemmat tutkimukset vaikuttavat tukevan kevyiden järjestelmien mahdollisia hyötyjä. Esimerkiksi Astudillo-Salinas et al. (2016) vertailivat keskenään eri käyttöjärjestelmiä RPi B+ V1.2:lle ja havaitsivat muihin testattuihin järjestelmiin, eli Rasbian-, PiCore- ja Pidora-käyttöjärjestelmiin, verrattuna Raspbian LITE -käyttöjärjestelmän kuluttavan vähemmän energiaa lähes jokaisessa testissä. Shizukuishi ja Matsubara (2020) puolestaan saivat Linux-ydintä konfiguroimalla laskettua järjestelmän käyttämää muistia ja tehoa.

Yleisesti ottaen ohjelmistojen keventämisen voidaan nähdä vähentävän järjestelmän tekemän prosessointityön määrää. Verrattuna raskaisiin järjestelmiin, tämä sallii esimerkiksi CPU:n viettävän enemmän aikaa tyhjäkäyntitilassa, jossa se voi mahdollisesti kuluttaa vähemmän energiaa (Wysocki, 2018). Vähäinen työmäärä voi tehtävästä riippuen vaikuttaa positiivisesti energiankulutukseen aiheuttamalla myös vähäistä määrää verkkoliikennettä. Verkkoyhteyden käytöstä puhutaan lisää kappaleessa 3.5 .

3.4 Käyttöjaksottaminen ja lepotilan hyödyntäminen

Käyttöjaksottaminen voi olla etenkin IoT-sovelluksissa energiankulutuksen suhteen hyödyksi, sillä monen IoT-laitteen voidaan olettaa tekevän tietty toimenpide tietyin ennalta määrätyin väliajoin ja näiden toimenpiteiden välissä ne saattavat olla tyhjäkäynti- tai lepotilassa. Käyttöjaksottamisen hyötyjä on mahdollista parantaa pelkän ohjelmallisen konfiguraation avulla, sillä esimerkiksi Amirtharaj et al. (2020) saivat systemd-yksiköitä (eng. unit) disabloimalla nopeutettua Linux-pohjaisen solmun käynnistymis- ja sammumisnopeutta ja siten laskemalla käyttöjaksosyklien aikana aiheutunutta kokonaisenergiankulutusta.

Käyttöjaksottamisen suhteen lepo- tai tyhjäkäyntitilassa kulutettu energia on siis selvästi tärkeässä roolissa akun varassa toimivan laitteen toiminta-ajan maksimoimiseksi. RPi:n tapauksessa sammutettu laite, eli tämän työn kontekstissa Linux-järjestelmissä `poweroff`-komennon kautta saavutetussa tilassa oleva laite, kuluttaa ilman erityistoimenpiteitä tietyn määrän vakiotehoa. Esimerkiksi Astudillo-Salinas et al. (2016) mittasivat RPi B+ V1.2:lle sammutettuna noin 0.3 W vakiokulutuksen, mikä vastaa heidän mukaansa noin 78 %

matalampaa kulutusta verrattuna käynnissä oleviin tiloihin. Kenties suurin parannus lepotilassa käytettyyn energiaan voidaan kuitenkin saavuttaa laitteiston ja ohjelmistojen konfiguroinnin yhteisvaikutuksen myötä, sillä mm. Yamazaki ja Nakajima (2023) saivat laskettua lepotilan virrankäytön jopa 1/40-osaan alkuperäisestä konfiguroimalla sulautettuja ohjelmistoja ja parantamalla elektronisten piirien vuotovirran vähentämisen keinoja.

Tähän liittyen RPi 5 tarjoaa joitain etuja malleihin 3B+ tai 4B nähden. RPi 5:n asetuksia voidaan konfiguroida siten, että – ainakin dokumentaation mukaan – sammutetussa tilassa sen tehonkäyttö laskee noin 0.01 wattiin. Ennen näiden asetuksien muokkaamista se kuluttaisi sammutetussa tilassa noin 1-1.4 W. Lisäksi, RPi 5.n kanssa voidaan hyödyntää paristolla toimivaa RTC-moduulia (real time clock), jonka avulla se voidaan herättää lepotilasta, mikä ei ole kaikilla RPi-malleilla mahdollista. (Raspberry Pi, 2024.)

3.5 Verkkoliikenteen optimointi

Verkkoliikenteen määrä riippuu oletettavasti etupäässä sovelluksesta, mutta tiedostamalla sen vaikutukset energiankulutukseen voidaan pyrkiä suunnittelemaan sovellus tavalla, jolla verkkotoimintojen aiheuttama energiankulutus minimoituu. Tämä on tärkeää etenkin IoT-sovelluksissa, joissa jo määritelmän perusteella verkonkäyttö on oleellinen osa.

Verkkolaitteiden ja niiden kanssa toimivien ohjelmien käynnissä pitäminen vaatii jo jonkin verran energiaa, sillä mm. Astudillo Salinas et al. (2016) havaitsivat Ethernet-kaapelin irrottamisen saavan aikaan jopa 19 % matalamman tehonkäytön vertailutilaan nähden. Lisäksi, Bekaroon ja Santokheen (2016) kokeiden perusteella verkosta irtautuminen ”aina, kun mahdollista” saattoi laskea käytettyä keskimääräistä teho jopa 30 %. Myös datasiirron määrä (bittejä per sekunti) verkon yli vaikuttaa energiankulutukseen. Kaup et al. (2014) luomassa mallissa WiFi:n ja Ethernetin avulla tehtyä datasiirtoa mallinnettiin toisen ja neljännen asteen polynomeilla pienellä virheellä kaistanleveyden (W/Mbps) funktiona. Nämä havainnot vihjaavat, että esimerkiksi protokollan valinta on tärkeä päätös myös energiankulutuksen suhteen, ja tämä on yksi syy minkä takia protokollat kuten MQTT ovat suosittuja IoT-verkoissa (MQTT, 2022).

4 Järjestelmä energiankulutuksen mittaamiseksi

Tässä kappaleessa esitellään järjestelmä, jolla energiankulutuksen mittaus sekä kokeet toteutetaan. Kappaleessa 4.1 kuvaillaan vaatimuksia, joiden pohjalta mittausjärjestelmä on toteutettu. Kappaleessa 4.2 tutustutaan kaavioon mittausjärjestelmästä ja komponentteihin, joista mittausjärjestelmä muodostuu.

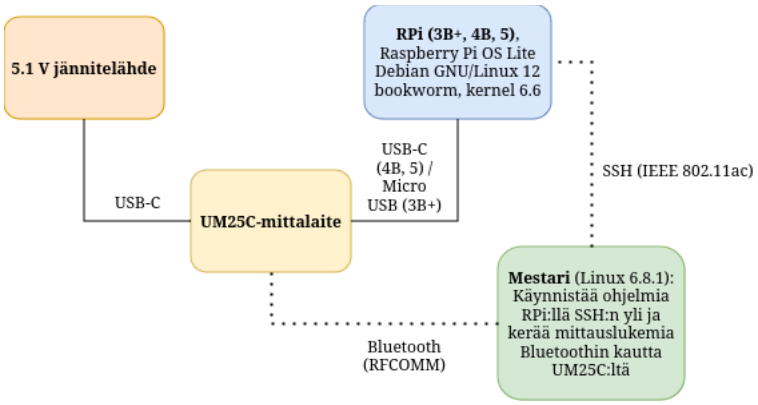
4.1 Vaatimukset

Mittarin on oltava riittävän tarkka, siten, että sillä kyetään havaitsemaan mahdollisesti pieniä muutoksia RPi:n energiankulutuksen suhteen, kun sovelletaan kappaleessa 3 kuvattuja menetelmiä. Lisäksi, kuten myös Kaup et al. (2014) kuvailevat, mittari ei saa aiheuttaa niin suurta pudotusta jännitteessä, että RPi ei käynnistyisi tai muuten kykenisi toimimaan oikein. Vaaditun mittaustarkkuuden suhteen esimerkiksi Astudillo-Salinas et al. (2016), Liang ja Lai (2010) ja Kaup et al. (2014) kykenivät mittauksissaan havaitsemaan muutaman milliwatin ja millijoulen resoluutiolla riittävän hyvin eroja tehonkäytössä ja energiankulutuksessa. Tämän perusteella mittarin suhteen tulisi pyrkiä mahdollisuuksien mukaan saavuttamaan vähintään yhtä hyvä tarkkuuden taso.

Mittauslukemat on kyettävä saamaan helposti talteen. Tämä vaatimus on edellytys mittaustietojen analyysiä varten. Mittauksia on myös kyettävä toistamaan helposti samalla tavalla monta kertaa. Tätä vaaditaan, jotta analyysiä varten saadaan systemaattisesti kerättyä riittävästi näytteitä luotettavia päätelmiä varten. Mittauksia tulisi myös pystyä automatisoimaan siten, että inhimillisiltä virheiltä vältytään näytteitä kerätessä. Lisäksi, mittausjärjestelmä ei saa aiheuttaa suuria virheitä mittaustuloksissa. Esimerkiksi, koska RPi voi olla yhteydessä paikalliseen verkkoon SSH:ta varten, on varmistettava, että tällöin mahdolliset häiriösignaalit ovat joko kokonaan estettävissä tai havaittavissa ja käsiteltävissä analyysivaiheessa.

4.2 Mittausjärjestelmän komponentit

Kappaleen 4.1 kuvaamiin vaatimuksiin vastaavan mittausjärjestelmän kaavio on esitelty kuvassa 2. Mittausjärjestelmän pääkomponentit ovat mittari (kuvassa keltainen laatikko), tietokone, jolla järjestelmää hallitaan (Mestari, kuvassa vihreä laatikko) sekä itse mittauksen kohde eli Raspberry Pi (joko 3B+, 4B tai 5, kuvassa sininen laatikko). Hallintatietokoneelle annettu nimi ”Mestari” sekä kuvan 2 kaavion tyyli saivat inspiraationsa Amirtharajin et al. (2020) työstä. Kukin näistä pääkomponenteista esitellään seuraavaksi tarkemmin.



Kuva 2. Mittausjärjestelmä

Mittaukset toteutetaan HangZhou RuiDeng Technologies Co. Ltd:n (2022) UM25C-mittarilla, jonka avulla saadaan kerralla sekä jännite-, virta-, teho- että energiankulutusmittaukset. Mittausjärjestelmässä tämä mittari asetetaan 5.1 V jännitelähteen ja RPi:n väliin, kuten kuvassa 2 (keltainen laatikko). Valmistajan esittämät mittarin kyvykkyydet jännitteen ja virran mittaamisen suhteen on esitelty taulukossa 2 – näiden tietojen perusteella mittarin on todettu olevan riittävän tarkka tämän työn tarkoituksiin. Mittarin aiheuttama jännitelasku on riittävän pientä siten, että RPi saa tarvitsemansa jännitteen. Mittarin virkistystaajuuden kuvataan olevan 1 Hz, joten hetkellisiä suureita ei voida mitata kovin tiheään tahtiin. Tämä ei kuitenkaan ole ongelma, koska työssä keskitytään mittaamaan aina yhtä tiettyä laitetilaa kerralla, joten, jos laitteen tila pysyy stabiilina ja mittausotoksia kerätään riittävästi riittävän pitkältä aikaväliltä, voidaan tehdä luotettavia päätelmiä hetkellisistä suureista, vaikka mittarin virkistystaajuus ei olisikaan kovin korkea. Taulukossa 2 mainittujen tietojen lisäksi mittari sisältää DX-

BT18-Bluetooth-moduulin, jonka avulla mittaustietoja voidaan hakea Bluetooth-yhteyden kautta (Sigrok, 2023).

Taulukko 2. UM25C-mittarin jännite- ja virtamittausominaisuudet

Suure	Mittausväli	Resoluutio	Tarkkuus
Jännite	4-24 V	0.001 V	0.05% ± 0.002 V
Virta	0-5 A	0.0001 A	0.1% ± 0.0004 A

Kuvassa 2 oikeassa alakulmassa esiintyvä vihreä laatikko edustaa tietokonetta, jolla RPi:tä ja mittaria hallitaan, eli Mestaria. Kyseessä on Linux 6.7.4 -pohjaista käyttöjärjestelmää hyödyntävä tietokone. Mestarilla voidaan tarpeen tullen ohjata Raspberry Pi:tä SSH-yhteydellä (Secure Shell) WiFi:n (IEEE 802.11ac) kautta, ja hakea Bluetoothilla mittarin lukemia. Tällä tavoin mittauksia voidaan hallita tarkasti yhdestä paikasta käynnistämällä koeohjelmia, jotka takaavat samanlaisen koesuoritteen joka iteraatiolla. Mittaustietojen hakua varten Mestarilla ajetaan itse kirjoitettua Python-ohjelmaa, joka muodostaa RFCOMM-protokollan (Radio frequency communication) mukaisen yhteyden mittariin. Ohjelma hakee mittarilta tämän yhteyden kautta tarkennetuin väliajoin hetkellisen virran (mA), jännitteen (mV) ja tehon (mW) sekä kertyneen energiankulutuksen (mWh). Ohjelma lisää mittauksiin mittaushetkeä kuvaavan liukuluvun (UNIX-aikaleiman) ja kirjoittaa tulokset CSV-muodossa analyysiä varten. Vaikka mittarin kanssa olisi mahdollista käyttää valmista ohjelmaa joko Android- tai Windows-järjestelmällä (HangZhou RuiDeng Technologies Co., Ltd, 2022), käytännöllisyyden ja kokeiden suhteen paremman hallittavuuden saavuttamiseksi ohjelma päätettiin kirjoittaa itse. Mittarin protokolla saatiin selville Sigrokin (2023) web-sivulta.

Bluetooth-yhteyden käyttö mittaustietojen hakemiseen saattaa vaikuttaa mittarin palauttamiin lukemiin. Koska vaikutuksesta ei löydetty dokumentoitua tietoa, Bluetoothin käytön mahdollinen vaikutus mittarin palauttamiin lukemiin arvioidaan ensimmäisen mittauskokeen avulla kappaleessa 5 . Vastaavasti, kuten kappaleessa 3.5 mainittiin, verkkoyhteyden käyttö – tässä SSH:ta varten – saattaa vaikuttaa huomattavasti RPi:n käyttämään tehoon kokeiden aikana, joten myös tämän mahdollinen vaikutus arvioidaan kappaleen 5 alussa.

Varsinainen mittauksen kohde on kuvassa 2 mittalaitteen ja Mestarin välissä esiintyvä Raspberry Pi (sininen laatikko). Työssä käytetään malleja 3B+, 4B ja 5, joiden tekniset tiedot on esitelty taulukossa 3 Raspberry Pi Ltd (2023a) ja (2023b) sekä Magpi (2019) raportoimien tietojen perusteella. Raspberry Pi OS Lite on kevyin Debian-pohjainen variantti Raspberry Pi OS -käyttöjärjestelmästä ja siinä ei ole mukana valmista työpöytäympäristöä (eng. desktop environment). Kyseinen käyttöjärjestelmä valikoitui, koska työpöytäympäristöä ei koettu tarpeelliseksi ja Astudillo-Salinasin et al. (2016) kokeiden perusteella Lite-versio vaikuttaa käyttävän vähän tehoa verrattuna esimerkiksi PiCore- tai Pidora-järjestelmiin.

Verkkoyhteyden suhteen RPi on kokeiden aikana oletuksena yhteydessä verkkoon langattomasti WiFi:n avulla – jos Ethernet-yhteyttä tai muita perustilasta poikkeavia asetuksia käytetään kappaleessa 5 esitettävissä kokeissa, tästä mainitaan erikseen. Kokeissa kohdataan verkkoyhteyden suhteen kolme mahdollista tapausta: WiFi on käytössä (rajapintaa ei disabloitu) ja Ethernet ei (kaapelia ei kytketty ja/tai rajapinta disabloitu), WiFi sekä Ethernet ovat käytössä tai vain Ethernet on käytössä. RPi:lle valikoitunut käyttöjärjestelmä hyödyntää oletuksena *NetworkManager*-palvelua. Mitään kyseisen palvelun oletusasetuksia ei ole muokattu kokeiden aikana. Seuraavaksi kuvattavat faktat varmistettiin RPi 4B -mallilla *NetworkManager*-palvelun lokikirjauksista sekä `ip`-työkalun avulla. *NetworkManager*-palvelun DHCP-implementaation (Dynamic Host Configuration Protocol) toimesta RPi pyytää heti käynnistyttyään automaattisesti reitittimeltä IP-osoitteen (Internet Protocol) ja *NetworkManager* asettaa jonkin saatavilla olevan verkkorajapinnan oletusrajapinnaksi. Kun vain WiFi tai Ethernet on käytössä, mutta ei molemmat, *NetworkManager* asettaa vastaavan verkkorajapinnan (tässä tapauksessa ”wlan0” tai ”eth0”) oletusrajapinnaksi. Kun sekä WiFi että Ethernet ovat käytössä, kumpikin rajapinta saa IP-osoitteen, mutta Ethernet-rajapinta ”eth0” asetetaan oletusrajapinnaksi. Tällöin WiFi-rajapinnan kautta ei siis tapahdu verkkoliikennettä (varmistettu lukemalla /proc/net/dev-tiedostoa), vaan kaikki datasiirto tehdään oletuksena Ethernetin välityksellä.

Jos kokeet eivät aiheuta suurta rasitusta siten, että tuuletin olisi tarpeen, RPi 5:llä tuuletinta ei käytetä, koska sen kuormitus ei kokeiden perusteella pysy tasaisena. Myös mm. Lambert et al. (2021) havaitsivat huomattavia eroja tuuletinta hyödyntävän ja ilman tuuletinta toimivan RPi:n tehonkäytön välillä ainakin käynnistysvaiheessa. Lisäksi,

energiankulutuksen minimoimista tavoittelevien IoT-laitteiden suhteen voidaan olettaa, että jäähdytys toteutetaan passiivisesti eikä aktiivisin menetelmin kuten tuulettimilla.

Taulukko 3. Raspberry Pi -mallien tiedot

	Raspberry Pi 3B+	Raspberry Pi 4B	Raspberry Pi 5
Käyttöjärjestelmä	Raspberry Pi OS Lite. Debian GNU/Linux 12 (bookworm). 64 bit. Kernel 6.6	Raspberry Pi OS Lite. Debian GNU/Linux 12 (bookworm). 64 bit. Kernel 6.6	Raspberry Pi OS Lite. Debian GNU/Linux 12 (bookworm). 64 bit. Kernel 6.6
Järjestelmäpiiri	Broadcom BCM2837B0	Broadcom BCM2711	Broadcom BCM2712
CPU	1.4 GHz quad-core 64-bit ARM Cortex-A53	1.5 GHz quad-core 64-bit ARM Cortex-A72	2.4GHz quad-core 64-bit Arm Cortex-A76
GPU	VideoCore IV	VideoCore VI	VideoCore VII
Verkko	5 GHz IEEE 802.11ac WiFi ja Gigabit Ethernet USB-väylän kautta	5 GHz IEEE 802.11ac WiFi ja Gigabit Ethernet	5 GHz IEEE 802.11ac WiFi ja Gigabit Ethernet
Bluetooth	Bluetooth 4.2, BLE	Bluetooth 5.0, BLE	Bluetooth 5.0, BLE
GPIO	Standard 40-pin GPIO header	Standard 40-pin GPIO header	Standard 40-pin GPIO header
RAM	1GB LPDDR2 SDRAM	2GB LPDDR4 SDRAM	8GB LPDDR4X-4267 SDRAM
Talletus	Verbatim Premium MicroSDHC, 16 GB	Verbatim Premium MicroSDHC, 16 GB	Verbatim Premium MicroSDHC, 16 GB
Syöttöteho	5V/2.5A, Micro USB	5V/3A, USB-C	5V/3A, USB-C
HDMI	1x full HDMI	2x micro HDMI	2x micro HDMI
USB	4x USB2	2x USB2 ja 2x USB3	2x USB2 ja 2x USB3

5 Kokeet ja tulokset

Tässä kappaleessa esitellään energiankulutuksen minimoinnin keinojen arvioimiseksi laadittuja kokeita sekä analysoidaan kokeiden tuloksia. Kaikki kappaleen kuvaajat ja kaaviot sekä tilastollinen analyysi on toteutettu R-kielellä (Linux-versio 4.3.2). Ensimmäiset kaksi koetta arvioivat mittausjärjestelmän Bluetooth- ja SSH-yhteyden mahdollisia vaikutuksia koetuloksiin. Näitä seuraavat kokeet keskittyvät kirjallisuuskatsauksen avulla löydettyjen energiankulutuksen laskemisen menetelmien tai periaatteiden arviointiin. Kaikki kokeita ja tuloksien analysointia varten luotu koodi on julkisesti saatavilla työtä varten luodussa GitHub-repositoriossa[1].

Kokeiden suhteen on tärkeä tietää, että mikäli koetuloksissa raportoidaan mittausjakson aikana kertynyt energiankulutus, tämä lukema on skaalattu kertomalla kulutus tavoitemittausajan ja todellisen mittausajan suhteella. Tämä johtuu siitä, että mittausjaksojen pituuksia on haastavaa ajoittaa täsmällisesti mittarilta tiedonhakuun liittyvien mahdollisten viiveiden myötä. Energiankulutus on testauksen perusteella tasaisen kuormituksen kokeissa pääosin lineaarista, joten skaalaus tekee lukemista vertailukelpoisia vääristämättä tuloksia. Lisäksi, näistä mahdollisista viiveistä johtuen, jos kokeessa kuvaillaan, että kerätään esimerkiksi 1500 sekunnin ajan yksi otos per sekunti, tämä tarkoittaa, että tavoitetaajuus on yksi otos per sekunti ja mittausjakso on 1500 sekuntia, mutta lopputuloksena ei välttämättä saada mahdollisten viiveiden vuoksi täsmälleen 1500 otoksen näytettä. Jos tarvitaan täsmällinen määrä otoksia, joudutaan luopumaan tarkasta mittausjakson ajallisesta pituudesta.

Koetulosten havainnollistamiseksi mittauslukemat voidaan tulkita kuten Yamazakin ja Nakajiman (2023) työssä käyttäen seuraavaa (tositilanteen suhteen yksinkertaistettua) yhtälöä:

$$t = \frac{Q}{I_d} = \frac{Q}{(P_d / U_d)} \tag{4}$$

1 https://github.com/jaakkopy/rpi_measurements

Yhtälössä t (tunteina) on paristolla tai akulla toimivan RPi:n toiminta-aika, Q [Ah] on pariston tai akun alkuperäinen varaus ja I_d [A], P_d [W] sekä U_d [V] ovat päivittäisen virrankulutuksen, tehon ja jännitteen keskiarvot. Yhtälöä (4) voidaan käyttää arvioimaan suhteellista toiminta-ajan pidentymistä, kun on saatu aikaan säästöä virrankulutuksessa. Esimerkiksi, jos kokeen tuloksena saavutetaan keskiarvoltaan 20 % matalampi virrankulutus, mutta kaikki muu pysyisi samana verrattavaan tilaan nähden, yhtälön (4) perusteella tämä vastaisi 25 % pitempää laitteen toiminta-aikaa (1/0.8 = 1.25). Vastaavasti 50 % matalampi virrankulutus verrattuna alkuperäiseen riittäisi jo tuplaamaan laitteen toiminta-ajan. Tositilanteessa laitteen ottama virta ei pysy vakiona ja akusta ei realistisesti saada käytettyä koko varausta 100 % hyötysuhteella, mutta (4) antaa yksinkertaisen viitekehyksen, jonka avulla tuloksia voidaan arvioida.

5.1 Mittarin Bluetooth-yhteyden vaikutus energiankulutukseen

Koska Mestari hakee UM25C:ltä mittauslukemat Bluetooth-yhteyden kautta, tämän toimenpiteen mahdollinen vaikutus mittauslukemiin on syytä arvioida. Tässä kokeessa Bluetooth-yhteyden vaikutusta arvioidaan säätämällä tiheyttä (hakuja per mittausjakso), jolla Mestari hakee mittaustietoja mittarilta. Koe toteutettiin RPi 4B -mallilla.

Kun UM25C-mittarilta pyydetään lukemia, se palauttaa aina 130 tavua. Pyyntö lukemista tehdään lähettämällä mittarille yhden tavun (tässä tapauksessa 8 bittiä) pituinen koodi. Ottamatta huomioon RFCOMM-protokollan vaikutusta siirrettyyn datamäärään, jokainen pyyntö aiheuttaa mittarilla siis 1 tavun vastaanoton ja 130 tavun lähetteen. Tässä kokeessa tilannetta arvioitiin neljässä tapauksessa: kun mittaustietoja haettiin 1 s, 5 s, 10 s ja 15 s välein. Nämä aikaerot kuvastavat aiemmin kuvatun perusteella merkittäviä eroja mittarilta/mittarille siirrettyjen tavujen määrissä koetta kohti. Kutakin hakuväliä kohden koe toistettiin 250 kertaa. Kullakin koeiteraatiolla Mestari haki Bluetoothin kautta mittarilta lukemia 30 sekunnin ajanjakson (tavoitepituus jaksolle, ei välttämättä tarkka mainituista viiveistä johtuen) aikana aiemmin mainituin aikavälein – eli 1 s, 5 s, 10 s ja 15 s hakuvälejä käyttäen 30 s jakson aikana haettaisiin lukemia 30, 6, 3 ja 2 kertaa. Lukutiheyden madaltaminen, eli mittauksien hakemisen välisen aikavälin pidentäminen, ei vaikuta mittausjakson aikana kertyneen kokonaisenergiankulutuksen lukeman tarkkuuteen,

koska mittari integroi energiankulutusta jatkuvasti taustalla itse, vaikka tietoja ei välissä haettaisi Bluetoothin kautta.

Kokeen tuloksena saatiin lukemia, joiden pohjalta jokaista käytettyä hakuväliä kohden keskiarvo 30 s ajanjakson energiankulutuksille voidaan pyöristää yhden desimaalin tarkkuudella 12 mWh:iin. Nollahypoteesia kokeen neljän tapauksen 30 s aikajaksojen energiankulutuksien mediaanien yhtäläisyydestä voidaan testata Kruskal-Wallisin H-testillä ja keskiarvojen yhtäläisyyttä voidaan testata ANOVA-testillä (kulutus on approksimaalisesti normaalijakautunutta). Yhtenä selittävänä tekijänä käytetään kummassakin tilastollisessa testissä hakuaikaa ja vasteena mittausjakson aikana käytettyä energiaa. ANOVA-kokeen tuloksena saatiin testitilasto $F(3) = 0.799$ ja P-arvo 0.494. Kruskal-Wallis-testistä saatiin testitilasto $H(249) = 249$ ja P-arvo 0.4881. Riskitasolla 0.05 tulos siis vihjaa, että mittarista Bluetooth-yhteyden kautta tietojen hakemisella 1s, 5s, 10s tai 15s välein ei vaikuta olevan merkitsevää eroa energiankulutuslukemien mediaaniin tai keskiarvoon. Tämän perusteella seuraavissa kokeissa oletetaan, että Mestarin Bluetoothin käyttö mittauslukemien hakemiseen mittarilta ei häiritse mittarin palauttamia lukemia merkitsevästi.

5.2 SSH-yhteyden ja verkonkäytön vaikutus

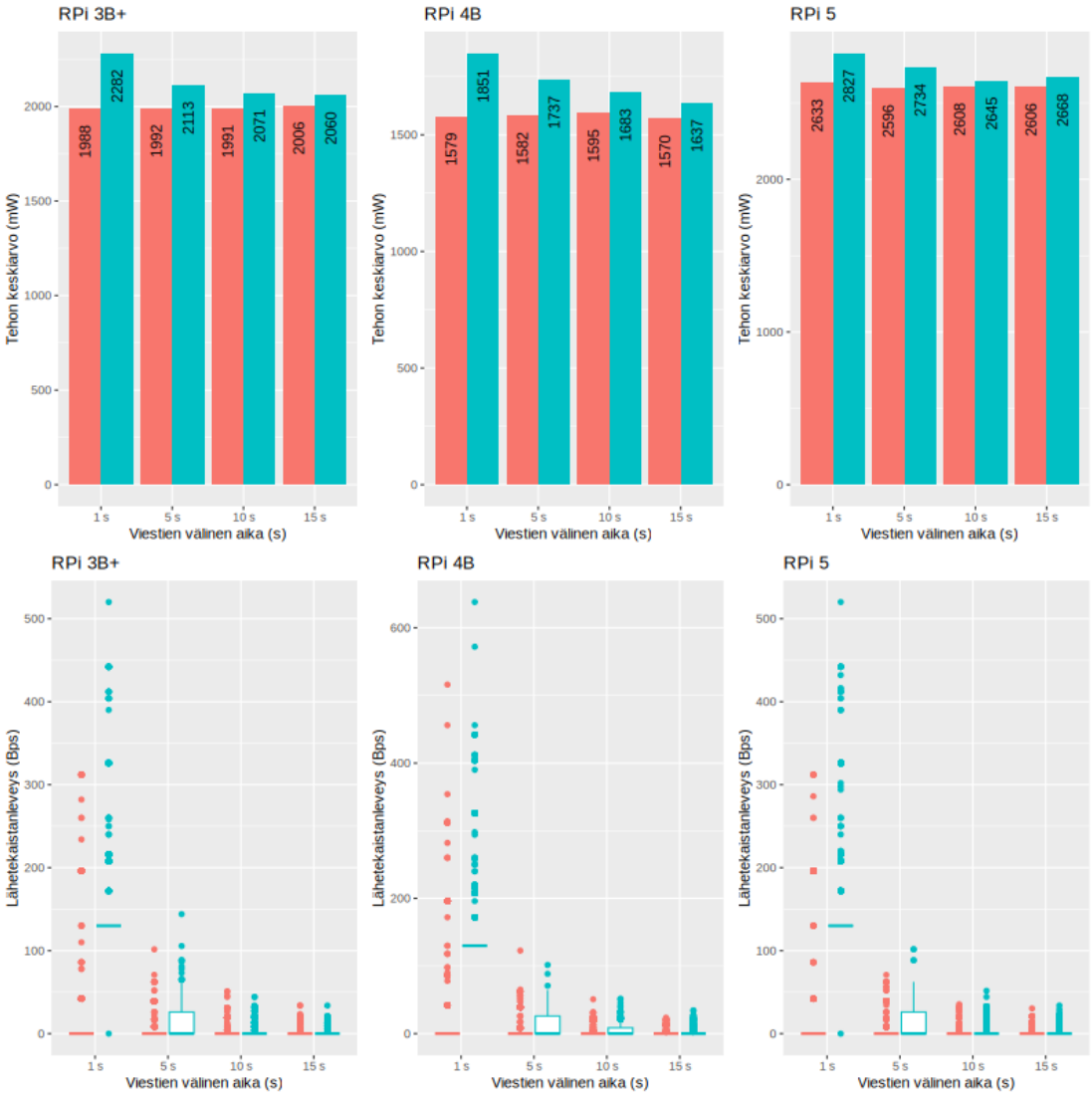
Kun SSH-yhteyttä ylläpidetään, ja RPi:llä ajava ohjelma tulostaa jotain päätteelle, tuloste kirjoitetaan verkkoyhteyden yli myös Mestarille. Tämä aiheuttaa selvästi tietyn verran verkkoliikennettä ja muuta prosessointityötä, joiden yhteisvaikutus käytettyyn tehoon halutaan arvioida, koska SSH-yhteyttä käytetään mahdollisesti kokeiden aikana. Koetta varten RPi ohjataan suorittamaan Python-ohjelmaa, joka laskee tietyn verkkorajapinnan yli tapahtuvien lähetysten kaistanleveyttä (Bps) ja optionaalisesti lähettää viestejä Mestarille SSH-yhteyden kautta tarkennetuin väliajoin. Kaistanleveys lasketaan vastaavalla tavalla kuin Kaupin et al. (2014) tutkimuksessa: ohjelma lukee joka sekunti /proc/net/dev-tiedostosta montako tavua on lähetetty tietyn verkkorajapinnan kautta ja laskee

$$r_i = \frac{B_i - B_{i-1}}{\Delta T}, \tag{5}$$

jossa i=0,1,…,n kuvastaa monesko laskettu tai luettu arvo on kyseessä, r_i on i:nnes laskettu kaistanleveys, B_i on i:nnes /proc/net/dev-tiedostosta luettu tieto tarkennetun

verkkorajapinnan yli lähetetystä määrästä tavuja ja ΔT on aikaväli sekunteina (tässä kokeessa käytetty arvoa 1), joka odotetaan tiedoston lukemisten välissä (eli B_i ja B_{i-1} lukemisen välissä). Ohjelma voidaan ohjata joko kirjoittamaan tai olemaan kirjoittamatta vakiokokoinen (koko mitattu tavuina) viesti SSH-yhteyden yli tarkennetuin aikavälein, kuten vaikka joka sekunti tai joka viides sekunti. RPi:llä ajava ohjelma kirjoittaa luetut ja lasketut tiedot iteroidessa muistiin ja vasta lopuksi tiedostoon, jotta vältetään ylimääräinen I/O-kuormitus (input/output).

Koe toteutettiin seuraavalla tavalla. RPi ohjattiin suorittamaan edellä kuvattua ohjelmaa neljä eri kertaa käyttäen kullakin suorituskerralla viestien lähetysten välissä eri odotusaikoja: ensimmäisellä kerralla odotusaikana käytettiin 1 s, toisella 5 s, kolmannella 10 s ja neljännellä 15 s. Jokaisella koesuoritteella ohjelma oli RPi:llä ajossa 1500 sekunnin ajan ja samaan aikaan Mestari keräsi UM25C-mittarilta lukemia RPi:n ottamasta hetkellisestä tehosta joka sekunti. Vaihtelevan viestien välisen odotusajan lisäksi koe suoritettiin toisen kerran muuten täsmälleen samalla tavalla ja samoilla odotusajoilla, mutta sillä poikkeuksella, että viestejä ei lähetetty Mestarille ollenkaan – tällä tavoin voidaan arvioida odotusaikojen välisten suhteellisen muutosten lisäksi myös absoluuttisia muutoksia tehossa tapauskohtaisesti. Tuloksia tulkitessa on syytä pitää mielessä, että SSH-yhteyden yli tapahtuvan tiedonsiirron vaikutus energiankulutukseen ei johdu pelkästään verkkoliikenteen lisääntyneestä määrästä, vaan kulutukseen liittyy myös esimerkiksi SSH-toteutuksen mukainen salausalgoritmin suoritus jokaista lähetystä varten (Ylönen & Lonvick, 2006). Kokeen tulokset on esitetty kuvassa 3. Kuvassa sininen väri edustaa tapauksia, joissa RPi lähetti viestejä Mestarille ja punainen väri edustaa tapauksia, joissa viestejä ei lähetetty lainkaan.



Kuva 3. Eri aikavälein SSH-yhteyden kautta lähetettyjen viestien vaikutus RPi:n käyttämään tehoon. Sininen väri tarkoittaa tapauksia, joissa RPi lähetti Mestarille viestejä ja punainen väri tapauksia, joissa viestejä ei lähetetty.

Kuvan 3 ylemmän palkkikuvaajan sinisten ja punaisten palkkien korkeuseron perusteella voidaan jo silmämääräisesti erottaa jonkinlainen korrelaatio suuremman verkkoliikenteen määrän ja kohonneen tehon keskiarvon välillä. Vastaavasti, kuten olettaa saattoi, kun viestien lähetysten välinen aika (vaaka-akseli kuvaajissa) kasvaa, alemman kuvaajan esittämä lähetekaistanleveyden mediaani näyttää laskevan (viiva laatikkokuvaajan laatikoiden sisällä). Suurin käytetyn tehon keskiarvo ilmeni jokaisella laitteella, kun viestien lähettämisen välinen aika oli yksi sekunti. Nollahypoteesia käytetyn tehon keskiarvon yhtäläisyydestä tapauksille, joissa viestejä lähetettiin ja ei lähetetty, voidaan testata kutakin odotusaikaa kohden kahden näytteen T-testillä. Käyttäen 95 % luottamustasoa jokaista viestien lähetyksen välistä aikaa kohden (1, 5, 10 ja 15 sekuntia)

havaittiin tilastollisesti erittäin merkitsevä ero käytetyn tehon keskiarvossa ($p < 0.001$ jokaisessa tapauksessa). Joten riskitasolla 0.05 voidaan todeta SSH-yhteyden kautta tapahtuvien lähetysten vaikuttavan merkitsevästi RPi:n käyttämään tehoon, ja sen vuoksi SSH-yhteyden käyttö kokeiden aikana tulisi pitää minimissään luotettavien koetulosten edistämiseksi.

5.3 Laiteominaisuuksien käytöstä poistaminen

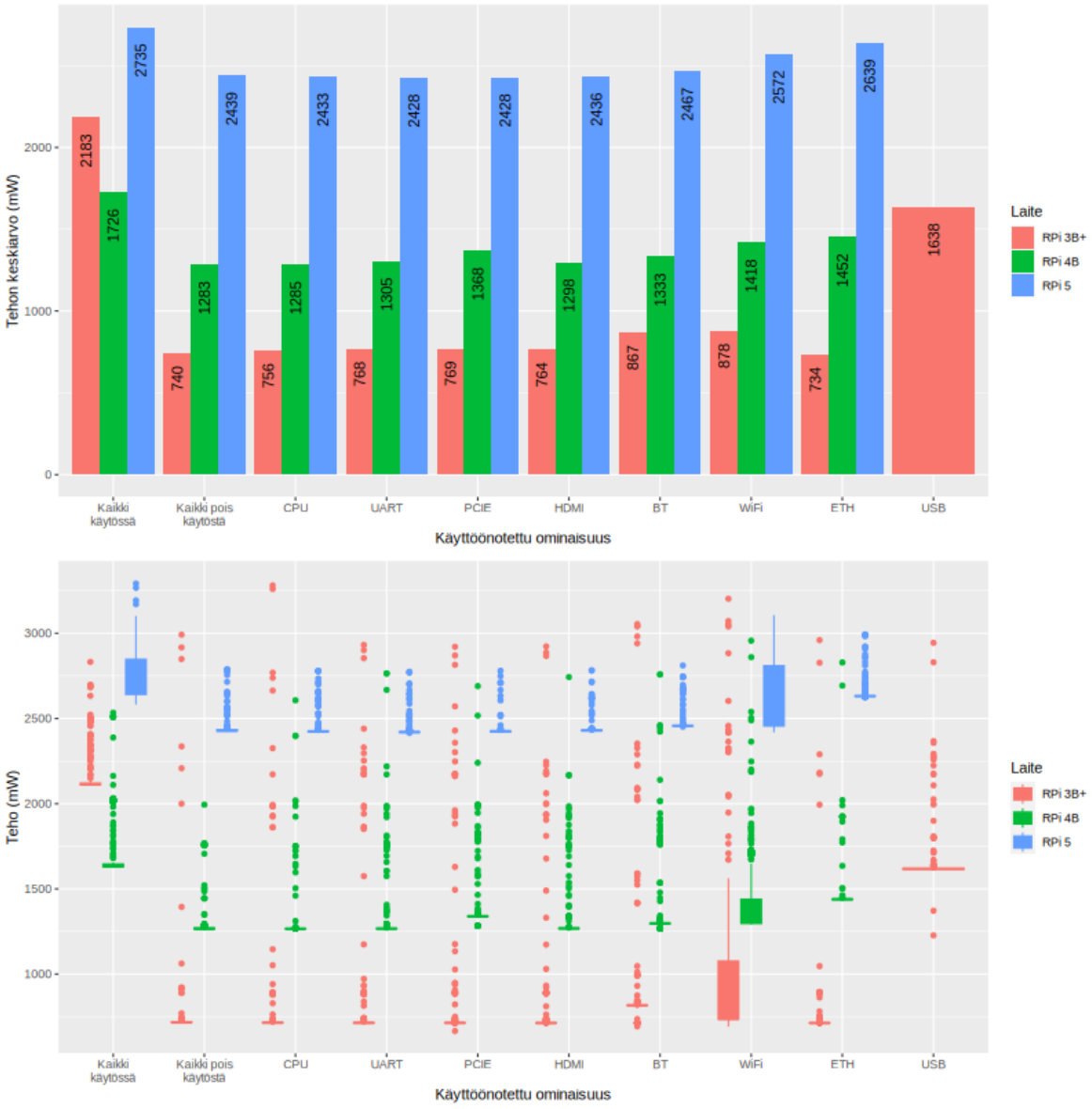
Tässä kokeessa arvioidaan minkälaisia vaikutuksia RPi:n ottamassa tehossa havaitaan, kun siltä kytketään pois kaikki taulukon 4 mukaiset ominaisuudet ja sitten aktivoidaan ominaisuuksia erikseen takaisin käyttöön. Kokeessa poistetaan käytöstä tiettyjä laiteominaisuuksia hyödyntämällä RPi:n config.txt-tiedostoa. Kyseessä on BIOS:n (Basic Input-Output System) toiminnallisuutta muistuttava tiedosto, jonka avulla järjestelmäparametreja, kuten WiFi:n tai Bluetoothin käyttöönottoa, voidaan hallita (Raspberry Pi, 2024). Kukin käytetyistä config.txt-tiedoston optioista on dokumentoitu Raspberry Pi OS Lite -käyttöjärjestelmän mukana tulevassa tiedostossa. Taulukkoon 4 valikoidut optiot löydettiin lukemalla kyseinen tiedosto läpi ja valikoimalla lupaavia asetuksia, jotka eivät olleet jo oletuksena poissa käytöstä. Ohessa on esitelty koetta varten käytetyn config.txt-tiedoston asetukset ennen kokeen suoritusta.

```
auto_initramfs=1
disable_fw_kms_setup=1
arm_64bit=1
arm_boost=1
```

Aiemmista kokeista poiketen myös Ethernet-kaapeli on kytkettynä RPi:hin tätä koetta varten, jotta sen vaikutus tehoon voidaan arvioida. Jokaista kokeen aikana toteutettua muutosta kohden Mestari kerää UM25C-mittarilta tehosta mittausotoksia 1000 sekunnin ajan yksi otos sekunnissa. RPi 3B+:n suhteen myös USB-ohjaimen käytöstä poistamisen vaikutusta voidaan arvioida, mutta RPi 4B:llä tai RPi 5:llä vastaavaa portti-tunniste-paria (1-1) ei löydetty. Lisäksi, on syytä mainita, että RPi 3B+:n järjestelmäpiirissä USB-ohjain ja Ethernet ovat samalla sirulla, kuten taulukossa 3 on kirjattu, joten USB-ohjaimen kytkeminen pois kytkee kyseisellä laitteella pois myös Ethernetin. Kokeen tulokset on esitelty kuvassa 4.

Taulukko 4. Laiteominaisuuksien käytöstä poistamisen menetelmiä

Ominaisuus	Käytöstä poiston menetelmä
WiFi	dtoverlay=disable-wifi (config.txt)
Bluetooth	dtoverlay=disable-bt (config.txt), systemctl disable bluetooth
HDMI	dtparam=hdmi=off (config.txt)
PCIE	dtparam=pcie=off (config.txt)
UART	systemctl disable hciuart
Yhden CPU-ytimen käyttö	maxcpus=1 (cmdline.txt)
USB-ohjain (Vain RPi 3B+. RPi 3B+ tapauksessa kytkee pois myös Ethernetin)	echo '1-1' /sys/bus/usb/drivers/usb/unbind
Ethernet	ip link set eth0 down



Kuva 4. Laiteominaisuuksien käyttöönoton vaikutus tehoon. Vaaka-akseli kertoo mikä ominaisuus on käytössä.

Kuvassa 4 ylemmän kaavion palkit edustavat RPi:n ottaman tehon keskiarvoa, kun taulukon 4 ominaisuuksia on otettu käyttöön yksi kerrallaan tilaan, jossa kaikki harkitut ominaisuudet ovat poissa käytöstä. Ensimmäisenä palkkikuvaajan vasemmanpuolisimmista palkeista voidaan havaita, että, kenties hieman yllättäen, kun mitään ominaisuuksia ei ole kytketty pois käytöstä, RPi 3B+ näyttää käyttäneen enemmän tehoa kuin RPi 4B. Toisaalta, kun kaikki taulukon 4 ominaisuudet ovat kytketty pois käytöstä, mallin 3B+ kulutus on noin 58 % mallin 4B kulutuksesta. Suurin tekijä, joka koetulosten perusteella tähän vaikuttaa, on 3B+:sta USB-ohjaimen pois kytkeminen – kun USB-ohjain otettiin käyttöön, verrattuna tilaan, jossa kaikki ominaisuudet olivat poistettu käytöstä, 3B+:n

ottama teho kasvoi noin 121 %. Lisäksi, kuvan 4 alemmasta laatikkokuvaajasta voidaan havaita kuinka WiFi:n käyttöönotto näyttää kasvattaneen hetkellisen tehon hajontaa huomattavasti. WiFi:n käyttöönoton aikaansaannoksena, verrattuna tilaan, jossa kaikki taulukon 4 ominaisuudet ovat kytketty pois käytöstä, hetkellisen tehon näytteiden keskihajonnat kasvoivat noin 67.6 %, 171.4 % ja 239.6 % malleilla 3B+, 4B ja 5. Mikäli tavoitteena on saada mahdollisimman tarkka arvio jonkin option vaikutuksesta RPi:n ottaman tehon keskiarvoon ja WiFi-yhteyttä ei tarvita, tämä vihjaa, että WiFi kannattaa pitää poissa käytöstä hajonnan minimoimiseksi.

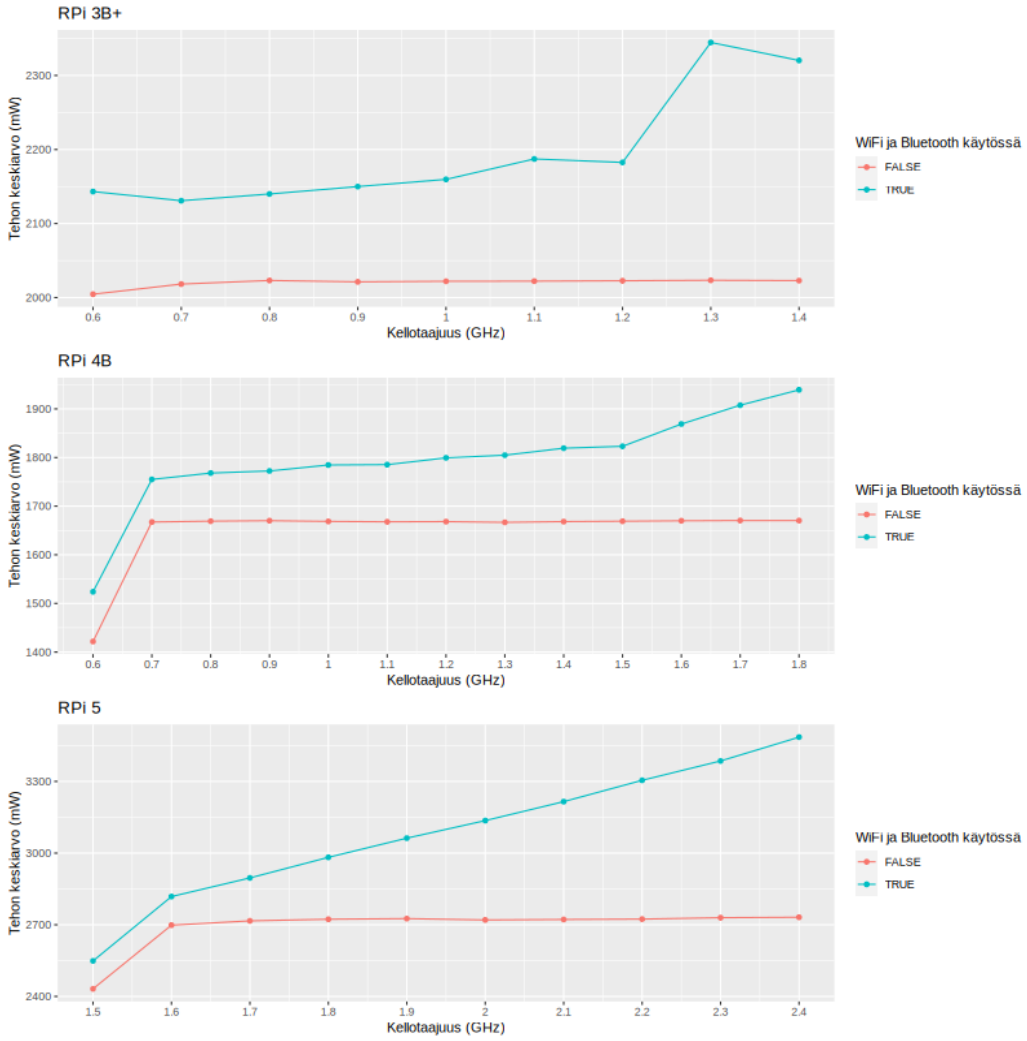
Yksittäisten muutosten merkitsevyyden arvioimiseksi voidaan soveltaa kahden näytteen T-testiä verraten kunkin muutoksen aikaansaamaa laitetilaa siihen tilaan, jossa kaikki taulukon 4 ominaisuudet ovat kytketty pois käytöstä. Tämän tuloksena mallilla 3B+ ne muutokset, joihin ei liittynyt riskitasolla 0.05 tilastollisesti merkitsevää muutosta tehossa olivat Ethernet, jolle $t(1222) = -0.68654$ ja $p = 0.4925$, HDMI, jolle $t(1156.4) = 1.8367$ ja $p = 0.06651$ ja CPU, jolle $t(1168.5) = 1.2412$ ja $p = 0.2148$, mutta on myös syytä mainita, että UART, jolle $t(1103.9) = 2.0846$ ja $p = 0.03734$ sekä PCIE, jolle $t(1057.7) = 2.1301$ ja $p = 0.03339$ olivat riskitasolla 0.05 vain täpärästi merkitseviä. Mallille 4B, vain CPU-optioon, jolle $t(1101.6) = 0.22894$ ja $p = 0.819$, ei liittynyt tilastollisesti merkitsevää muutosta. Lopuksi, mallille 5 HDMI-optio, jolle $t(1165) = -1.1776$ ja $p = 0.2392$, ei ollut merkitsevä.

Tulosten perusteella Bluetooth, WiFi ja Ethernet saivat aikaan merkittävimmät säästöt tehossa johdonmukaisesti kullakin RPi-mallilla, joskin mallin 3B+ kohdalla tämän kokeen toteutustavasta johtuen Ethernetin vaikutus vääristyy USB-ohjaimen kanssa tapahtuvan interaktion vuoksi. Bluetoothin enablointi aiheutti vertailutilaan nähden 17 %, 3.9 % ja 1.1 % kasvun tehon keskiarvossa malleille 3B+, 4B ja 5, vastaavasti. WiFi:n käyttöönotto puolestaan kasvatti tehon keskiarvoa 18.6 % (3B+), 10.5 % (4B) ja 5.4 % (5). Lopuksi, Ethernetille vastaavat kasvut olivat 13.1 % (4B) ja 8.2 % (5).

Tämän kokeen harkitsemista laitetiloista suurimmat laskut käytetyn tehon keskiarvossa ovat koetulosten perusteella 66.4 %, 25.6 % ja 11.2 % malleille 3B+, 4B ja 5. Mikäli jännite pysyisi vakiona ja saavutettaisiin vastaavat laskut tehossa, yhtälön (4) perusteella vastaavat laskut merkitsisivät noin 197.6 %, 34.4 % ja 12.6 % pitempiä toiminta-aikoja malleille 3B+, 4B ja 5.

5.4 Kellotaajuuden säätely

Tässä kokeessa arvioidaan kuinka prosessorin kellotaajuutta säätämällä voidaan vaikuttaa RPi:n tyhjäkäyntitilassa ottamaan tehoon. Koe toteutettiin ottamalla käyttöön CPUFreq-alijärjestelmän *userspace*-säädin, joka sallii käyttäjän asettaa prosessorin kellotaajuuden itse, ja keräämällä jokaista sallittua kellotaajuutta kohden mittausotoksia tyhjäkäyntitilan tehosta. Jokaista sallittua taajuutta kohden otoksia kerättiin 800 sekunnin ajan yksi otos sekunnissa. Koska kappaleen 5.3 kokeessa WiFi:n ja Bluetoothin käytön havaittiin aiheuttavan suurta hajontaa mittauslukemissa, koe toteutettiin kaksi kertaa: kun WiFi ja Bluetooth pidettiin aktivoituna ja, kun ne pidettiin disabloituna. Kokeen tulokset ovat nähtävissä kuvassa 5.



Kuva 5. Tyhjäkäyntitilan tehonkäyttö suhteessa prosessorin kellotaajuuteen, kun WiFi ja Bluetooth ovat käytössä ja pois käytöstä

Kuvan 5 perusteella nähdään, että WiFi:n ja Bluetoothin käyttö vaikuttaa aiheuttavan jyrkempää ja epätasaisempaa kasvua käytetyn tehon keskiarvossa suhteessa kasvavaan kellotaajuuteen verrattuna tilanteeseen, jossa WiFi ja Bluetooth ovat poistettu käytöstä. Kappaleen 5.3 kokeen tuloksien pohjalta havaittiin etenkin WiFi:n ylläpidon aiheuttavan muihin optioihin nähden suurta hajontaa hetkellisen tehon lukemissa, joten kuvan 5 havainnollistama ero tapausten välillä saattaa johtua WiFi:n aiheuttaman lisäprosessoinnin ja kasvaneen kellotaajuuden yhdistelmästä.

Kun WiFi ja Bluetooth olivat enabloituina, korkeimpia kellotaajuuksia edustavat tehonkäytön keskiarvot olivat 8.3 %, 27.2 % ja 36.8 % suuremmat verrattuna matalimpiin taajuuksiin malleille 3B+, 4B ja 5, vastaavasti. Kun WiFi ja Bluetooth olivat disabloituina, vastaavat erot olivat 0.9 % (3B+), 17.5 % (4B) ja 12.3 % (5). Jos WiFi ja Bluetooth pidettäisiin käytössä ja teho sekä jännite pysyisivät vakiona, tulokset tarkoittavat, että matalimmalla kellotaajuudella tyhjäkäyntitilassa toimivat laitteet pysyisivät vastaavilla keskimääräisen tehonkäytön laskuilla, eli -7.6 % (3B+), -21.4 % (4B) ja -26.9 % (5) laskuilla, toimintakuntoisina noin 8.2 % (3B+), 27.2 % (4B) ja 36.8 % (5) kauemmin verrattuna korkeinta kellotaajuutta hyödyntäviin laitteisiin.

6 Johtopäätökset

Työssä tutkittiin millaisia käytännössä sovellettavia menetelmiä tai periaatteita aiemmissa tutkimuksissa on käsitelty RPi:n energiankulutuksen minimoimiseksi ja millaisia vaikutuksia nätä keinoja tai periaatteita soveltamalla voidaan havaita RPi:n mallien 3B+, 4B ja 5 suhteen, kun niiden ottamaa tehoa mitataan erilaisissa koetilanteissa. Kirjallisuuskatsauksen myötä löytyneet menetelmät kategorisoitiin viiteen luokkaan: dynaaminen jänniteen ja kellotaajuuden säätely, laiteominaisuuksien käytöstä poisto, ohjelmistojen konfigurointi, käyttöjaksottaminen ja lepotilan hyödyntäminen sekä verkkoliikenteen optimointi. Työssä toteutetut kokeet liittyvät pääosin laiteominaisuuksien poistoon sekä kellotaajuuden säätelyyn, mutta myös verkkoliikenteen optimoinnin aihetta sivuttiin.

Koetulosten pohjalta tiettyjä laiteominaisuuksia käytöstä poistamalla havaittiin 66.4 %, 25.6 %, ja 11.2 % matalampaa keskimääräistä tehonkäyttöä malleilla 3B+, 4B ja 5, vastaavasti. CPU-kellotaajuutta säätämällä havaittiin pienimmän ja suurimman kellotaajuuden välillä 0.9 % (3B+), 17.5 % (4B) ja 12.3 % (5) suhteellista kasvua käytetyn tehon keskiarvossa, kun WiFi ja Bluetooth olivat poissa käytöstä, ja 8.3 % (3B+), 27.2 % (4B) ja 36.8 % (5) kasvua, kun WiFi ja Bluetooth olivat käytössä. Näiden tulosten pohjalta voidaan väittää, että RPi:n energiankulutusta on mahdollista laskea, ja siten vastaavan akkujen varassa operoivan IoT-solmun toiminta-aikaa on mahdollista pidentää, jo pelkän ohjelmallisen konfiguraation avulla.

Lähteet

Amirtharaj, I., Groot, T., & Dezfouli, B. (2020). Profiling and improving the duty-cycling performance of Linux-based IoT devices. *Journal of Ambient Intelligence and Humanized Computing*, *11*(5), 1967–1995.

Astudillo-Salinas, F., Barrera-Salamea, D., Vazquez-Rodas, A., & Solano-Quinde, L. (2016). Minimizing the power consumption in Raspberry Pi to use as a remote WSN gateway. *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*, 1–5.

Bekaroo, G., & Santokhee, A. (2016). Power consumption of the Raspberry Pi: A comparative analysis. *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, 361–366.

Bhattacharya, S., Gopinath, K., Rajamani, K., & Gupta, M. (2011). Software bloat and wasted joules: Is modularity a hurdle to green software? *Computer*, *44*(9), 97–101.

Geerling, J. (2021), Disabling cores to reduce the Pi Zero 2 W's power consumption by half. Viitattu 28.2.2024. Saatavilla: https://www.jeffgeerling.com/blog/2021/disabling-cores-reduce-pi-zero-2-ws-power-consumption-half

HangZhou RuiDeng Technologies Co., Ltd (2022), Instructions for USB Tester with Full Colour Display -Model: UM25/UM25C. [Online]

Ikonen, J., Nelimarkka, N., Nardelli, P. H. J., Mattila, N., & Melgarejo, D. C. (2022). Experimental evaluation of end-to-end delay in a sigfox network. *IEEE Networking Letters*, *4*(4), 194–198.

International Energy Agency (2023), Electricity Market Report 2023. Viitattu 21.1.2024. Saatavilla: https://www.iea.org/reports/electricity-market-report-2023

Karpowicz, M. P. (2016). Energy-efficient CPU frequency control for the Linux system. *Concurrency and Computation: Practice and Experience*, *28*(2), 420–437.

Kadota, K., Taniguchi, I. & Tomiyama, H. (2020). Measurement of performance and energy consumption of opencv programs on raspberry pi. Bulletin of Networking, Computing, Systems, and Software, 9(1), pp.35-39.

Kaup, F., Gottschling, P., & Hausheer, D. (2014). PowerPi: Measuring and modeling the power consumption of the Raspberry Pi. *39th Annual IEEE Conference on Local Computer Networks*, 236–243.

Lambert, J., Monahan, R., & Casey, K. (2021). Power consumption profiling of a lightweight development board: Sensing with the ina219 and teensy 4. 0 microcontroller. *Electronics*, *10*(7), 775.

Le Sueur, E. & Heiser, G. (2010). Dynamic voltage and frequency scaling: The laws of diminishing returns, Proceedings of the 2010 international conference on Power aware computing and systems (pp. 1-8).

Liang, W.Y. & Lai, P.T. (2010). Design and implementation of a critical speed-based DVFS mechanism for the android operating system. In 2010 5th International Conference on Embedded and Multimedia Computing (pp. 1-6). IEEE.

Magpi (2019). Raspberry Pi 3B+ Specs and Benchmarks. Viitattu 13.3.2024. Saatavilla: https://magpi.raspberrypi.com/articles/raspberry-pi-3bplus-specs-benchmarks

M. Hosny, K., Magdi, A., Salah, A., El-Komy, O., & Lashin, N. A. (2023). Internet of things applications using Raspberry-Pi: A survey. *International Journal of Electrical and Computer Engineering (IJECE)*, *13*(1), 902.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, *10*(7), 1497–1516.

MQTT (2022). MQTT: The Standard for IoT Messaging. Viitattu 3.2.2024. Saatavilla: https://mqtt.org/

Raspberry Pi (2024). Raspberry Pi Documentation. Viitattu 27.1.2024. Saatavilla: https://www.raspberrypi.com/documentation

Raspberry Pi Ltd (2023a). Raspberry Pi 4 Model B. Viitattu 6.2.2024. Saatavilla: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf

Raspberry Pi Ltd (2023b). Raspberry Pi 5. Viitattu 6.2.2024. Saatavilla: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf

Satyajit S. (2023). State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally. IoT Analytics. Viitattu 21.1.2024. Saatavilla: https://iot-analytics.com/number-connected-iot-devices/

Sigrok (2023). RDTech UM series. Viitattu 7.2.2024. Saatavilla: https://sigrok.org/wiki/RDTech_UM_series

Singh, P. K. & Rana, B. (2022). Duty-Cycling Techniques in IoT: Energy-Efficiency Perspective, in Lecture Notes in Electrical Engineering. Singapore: Springer Singapore Pte. Limited. pp. 505–512.

Shizukuishi, T., & Matsubara, K. (2020). An efficient tinification of the linux kernel for minimizing resource consumption. *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 1228–1237.

Systemd (2023). System and Service Manager. Viitattu 11.3.2024. Saatavilla: https://systemd.io/

Wysocki R. J. (2017). CPU Performance Scaling. The Linux Kernel. Viitattu 18.1.2024. Saatavilla: https://docs.kernel.org/admin-guide/pm/cpufreq.html

Wysocki R. J (2018). CPU Idle Time Management. The Linux Kernel. Viitattu 4.2.2024. Saatavilla: https://docs.kernel.org/driver-api/pm/cpuidle.html

Yamazaki, S., & Nakajima, Y. (2023). A sigfox energy consumption model via field trial: Case of smart agriculture. *IEEE Access*, *11*, 145320–145330.

Ylönen, T. & Lonvick, C. M. (2006). *The secure shell (SSH) transport layer protocol* (Request for Comments RFC 4253). Internet Engineering Task Force. https://datatracker.ietf.org/doc/rfc4253/