Open your mind. LUT.
Lappeenranta University of Technology

MITTAUSSIGNAALIN KÄSITTELY RASPBERRY PI:LLÄ
SIGNAL PROCESSING WITH RASPBERRY PI

Jouni Kortelainen

Kandidaatintyö
LUT School of Energy Systems
Sähkötekniikka

21.5.2017

TIIVISTELMÄ

Jouni Kortelainen
Opiskelijanumero
LUT School of Energy Systems
Sähkötekniikka
Mikko Kuisma

Mittaussignaalin käsittely Raspberry Pi:llä

2017

Kandidaatintyö.
23 s.

Työn tavoitteena oli selvittää millaiseen signaalinkäsittelyyn Raspberry Pi 3 B soveltuu. Soveltuvuutta arvioitiin tutkimilla laitteelle saatavia työkaluja, laitteella toteutettuja sovelluksia sekä sen tehokkuutta. Työ perustui kirjallisiin lähteisiin, sekä laitteella suoritettuihin testeihin.

Raspberry Pi:llä voidaan käyttää MATLAB:a, Simulink:iä sekä LabVIEW:a. MATLAB:n ja LabVIEW:n käyttö edellyttää ethernet yhteyttä toiseen tietokoneeseen. Simulink-mallit voidaan kääntää Raspberry Pi:llä toimiviksi, eikä käytönaikaista yhteyttä tarvita.

Raspberry Pi:llä voidaan käyttää korkean-tason kieliä. Stackexchangen tietokantojen mukaan suosituimmat Linuxin datankäsittelyn kielet olivat R ja Python. Molempiin on saatavalla monia signaalinkäsittelyssä käytännöllisiä kirjastoja, jotka ovat ilmaisia. R:n on myös saatavalla MATLAB:n Signal Processing Toolbox:n sisältämät funktiot.

Viime vuosina Raspberry PI on saanut suosiota myös kaupallisella puolella. Sitä käytetään usein yksinkertaisiin tehtäviin, joihin ei tahdota ostaa kallista laitteistoa, kuten digitaalisten mainoskylttien ohjaamiseen. Parhaimpina puolina pidetään halpaa hintaa, laskentatehoa, pientä virrankulutusta sekä joustavuutta. Laitteessa on myös omia GPIO pinnejä, joihin voidaan kytkeä suoraan kiinni erilaisia sensoreita, jotka eivät käy perinteisiin tietokoneisiin.

Yksi laitteen yleisistä käyttökohteista on dataloggeri, jolla käsitellään ja talletetaan mittausdataa. Dataloggerissa voidaan laskea esimerkiksi liikkuvaa keskiarvoa näytteiden välillä. Testin mukaan ikkunakoolla 10 Raspberry Pi laski liikkuvan keskiarvon 300 datapisteelle noin 11 millisekunnissa.

FFT-testissä Raspberry Pi laski Fourier muunnoksen nopeinten 2^4 datapisteelle, johon kului aikaa noin 43 mikrosekuntia. Liikkuvan keskiarvon sekä FFT:n laskenta onnistuu siis hyvin esimerkiksi 1 Hz näytteistystaajuudella jokaiselle näytteelle.

Tutkimuksia tehdessä laitteesta ei löydetty suuria puutteita, ja se osoittautui hyväksi vaihtoehdoksi pieniin signaalinkäsittelyn tehtäviin, joissa ei tarvita suurta laskentatehoa.

ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
Electrical Engineering

Jouni Kortelainen
Signal processing with Raspberry Pi

2017

Bachelor's Thesis.
23 p.

Examiner: professor Mikko Kuisma

The objective of this paper was to find out what kind of signal processing Raspberry Pi 3 is capable of. Capability was measured by researching what kind of tool are available for the device, what kind of applications have been done and how powerful the device is. The paper was based on written sources and test done with the device.

MATLAB, Simulink and LabVIEW can be used with Raspberry Pi. The use of MATLAB and LabVIEW require Ethernet connection between Raspberry Pi and a host computer. Simulink models can be compiled and executed on Raspberry Pi, without connection to the host.

According to the StackExchange's database, High-level languages R and Python are currently the most used data-analysis languages on Linux. Both of the languages have multiple free libraries for signal processing, which makes them well suited. All of MATLAB's Signal Processing Toolbox functions are also available in R from a free library.

During the last few years, Raspberry Pi has gained a lot of favor in commercial sector. The device is often used in simple tasks were a PC is not an affordable investment. One of these tasks is control unit of digital signs. Cheap price, low power consumption, computing power and flexibility are often regarded as the best parts of Raspberry Pi. The device also has its own GPIO pins that can be used with sensor that are not suitable to be used with ordinary computers.

One of the common applications is a datalogger, which handles and saves data. Datalogger could also be used to calculate moving average for the measurement signal between samples. According to a test, with window size of 10, Raspberry Pi calculate the moving average for 300 data points in about 11 milliseconds.

In a FFT-test Raspberry Pi calculated the Fourier transformation fastest for 2^4 data points. Calculation time was about 43 microseconds. Low time of moving average and FFT calculation indicates that the device could easily be used with 1 Hz sampling frequency and moving average or FFT can be calculated for every sample.

During the research device didn't appear to have big shortcomings and it proved to be a good option in small signal processing applications, that don't need a lot of computing power.

SISÄLLYSLUETTELO

KÄYTETYT MERKINNÄT JA LYHENTEET

A/D	Analogia-Digitaali muunnin
DFT	Discrete Fourier Transformation
D/A	Digitaali-analogia muunnin
FFT	Fast Fourier Transformation
FIR	Finite Impulse Response
GPIO	General Purpose Input Output
IIR	Infinite Impulse Response
IoT	Internet of Things
SBC	Single-Board Computer
WiMAX	Worldwide Interoperability for Microwave Access

K	Laskentanopeutta kuvaava luku
N	Datapisteiden lukumäärä
e	Virhe
T	Aika
y	Lähtösignaali
x	Tulosignaali

Alaindeksit

RMSE	Neliöllisen keskiarvon virhe

1. JOHDANTO

Digitaalinen signaalinkäsittely sai alkunsa jo 1960 luvulla, mutta 1990 luvulla tietokoneiden räjähdysmäinen lisääntyminen muutti markkinoita, mahdollistaen digitaalisen signaalinkäsittely julkisella puolella. Tänä päivänä suurin osa signaalinkäsittelystä toteutetaan digitaalisesti

Kaikki signaalit ovat lähtökohtaisesti analogisia ja ne on muunnettava digitaalisiksi AD-muuntimella. Muunnoksessa analoginen signaali kvantisoidaan tasoiksi. Jokainen kvantisointitaso vastaa yhtä binäärilukua. Lisäämällä kvantisointitasoja voidaan lisätä muunnoksen tarkkuutta ja siten digitaalisen signaalin oikeellisuutta. Muunnoksessa häviää silti aina jonkun verran dataa.

Digitaalisessa muodossa signaalia on helppo käsitellä ja se on valmiiksi oikeassa muodossa esimerkiksi tallentamista varten. Halutessaan digitaalisen signaalin voi muuttaa aina takaisin analogiseksi.

Molemmissa signaalinkäsittely muodoissa on hyviä ja huonoja puolia. Digitaalinen käsittelytapa on analogiseen verrattuna paljon joustavampi. Digitaalisessa käsittelyssä käsittely tapahtuu ohjelmallisesti, jonka koodia voidaan muuttaa helposti. Analogisessa käsittelyssä signaalia käsitellään puolestaan fyysisellä piirillä, jonka muokkaaminen jälkeenpäin on työlästä. Käsittelyn aikana analogiseen signaaliin voi muodostua häiriöitä, toisin kuin digitaaliseen. Digitaalinen tapa on kuitenkin kalliimpi kuin analoginen.

Raspberry Pi on yhden piirilevyn tietokone, joka on noussut lähivuosina suureen suosioon niin harrastelijoiden kuin ammattilaistenkin joukossa. Tässä työssä selvitetään, millaiseen digitaalisen signaalin käsittelyyn kaikille saatavissa oleva tietokone kykenee.

1.1 Tavoite, tutkimuskysymykset ja rajaus

Työn tavoitteena on selvittää, millaiseen signaalinkäsittelyyn Raspberry Pi soveltuu. Soveltuvuutta arvioidaan testaamalla erilaisia menetelmiä käytännössä ja arvioimalla niiden tehokkuutta. Tehokkuutta mitataan suoritusnopeuden sekä käsitellyn datajoukon koon suhteena.

Työssä pyritään löytämään vastaus siihen,

- kuinka tehokas Raspberry Pi on signaalinkäsittelyssä
- millaiseen signaalinkäsittelyn sovelluksiin Raspberry PI käy
- kuinka Raspberry Pi:tä voidaan hyödyntää mittausjärjestelmässä
- kuinka Raspberry Pi:n arkkitehtuuri soveltuu signaalinkäsittelyyn
- millaisia valmiita ohjelmistoja tai kirjastoja voidaan käyttää

1.2 Menetelmät

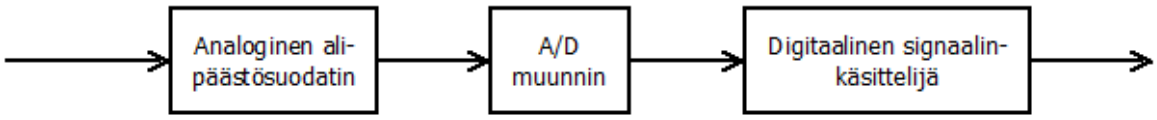
Työssä käytettävät menetelmät pohjautuvat kirjallisiin lähteisiin. Menetelmiä testataan käytännössä Raspberry Pi:llä. Laitteen soveltuvuutta arvioidaan tarkastelemalla testattujen menetelmien tehokkuutta. Tehokkuutta arvioidaan eri käyttöjärjestelmien, koodikielien ja valmiiden ohjelmistojen välillä.

2. SIGNAALIN DIGITAALINEN KÄSITTELY

Digitaalisen signaalinkäsittelyn katsotaan saaneen alkunsa vuonna 1948, jolloin alettiin tutkia viestien digitaalista lähettämistä sekä pulssikoodausmodulaatiota sekä virheenkorjausta. Käytännössä digitaalista signaalinkäsittelyä on suoritettu 1960 luvulta lähtien, kun ensimmäiset tietokoneet kehitettiin. Tuolloin laitteet olivat lähinnä armeijan, valtion sekä lääketieteen käytössä. 1980 luvulla tietokoneiden lisääntyminen toi digitaalisen signaalinkäsittelyn kaupalliselle puolelle, joka johti muun muassa matkapuhelimien yleistymiseen. Nykyään digitaalista signaalinkäsittelyä käytetään kaikkialla. (Nebeker, 1998)

2.1 Analogisesta digitaaliseksi

Kuvassa 2.1 on esitetty mittaussignaalin ja käsitellyn signaalin välillä olevat laitteet.

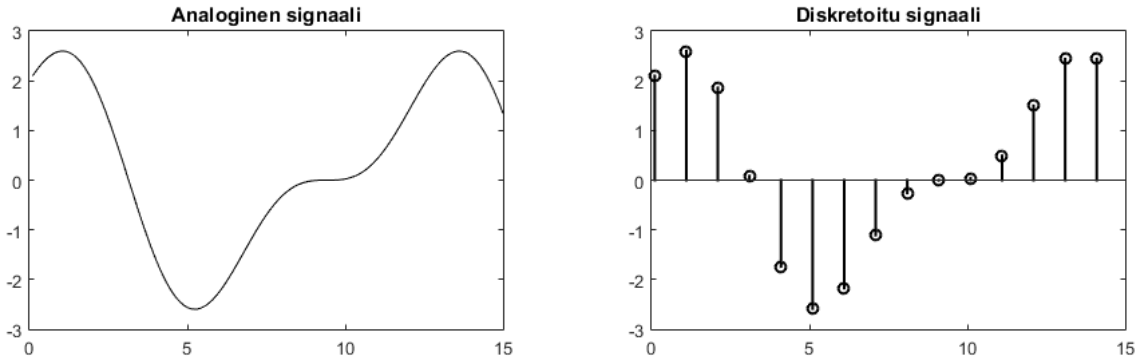


Kuva 2.1 Saadun analogisen signaalin ja käsitellyn signaalin väliset komponentit.

Mittalaitteelta saatu analoginen signaali suodatetaan analogisella alipäästösuodattimella, jonka tarkoituksena on poistaa Nyquist taajuutta korkeammat taajuudet signaalista. Nyquist taajuutta korkeammat taajuudet aiheuttaisivat signaalissa laskostumista ja tekisivät muunnetusta digitaalisesta signaalista epätarkemman. Suodatettu analoginen signaali muutetaan digitaaliseksi A/D muuntimella, jonka sisältö on esitetty kuvassa 2.2. Muunnoksen jälkeen digitaalisella signaalinkäsittelijällä voidaan suorittaa tahdotut operaatiot, jonka jälkeen voidaan suorittaa jatkotoimenpiteitä, kuten signaalin talletus tai analogiseksi muuntaminen.

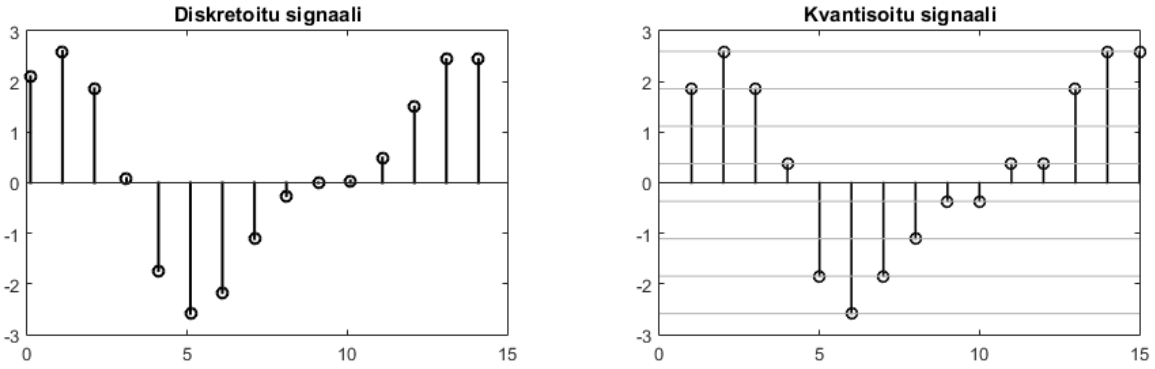
Kuva 2.2 A/D muuntimen toiminta.

Näytteenotossa analogisesta signaalista otetaan näytteitä tietyllä taajuudella, joista muodostetaan diskreettiaikainen signaali. Kuvassa 2.3 on otettu signaalista näytteitä 1Hz taajuudella. Signaalin muodot ovat vielä selvästi nähtävissä.



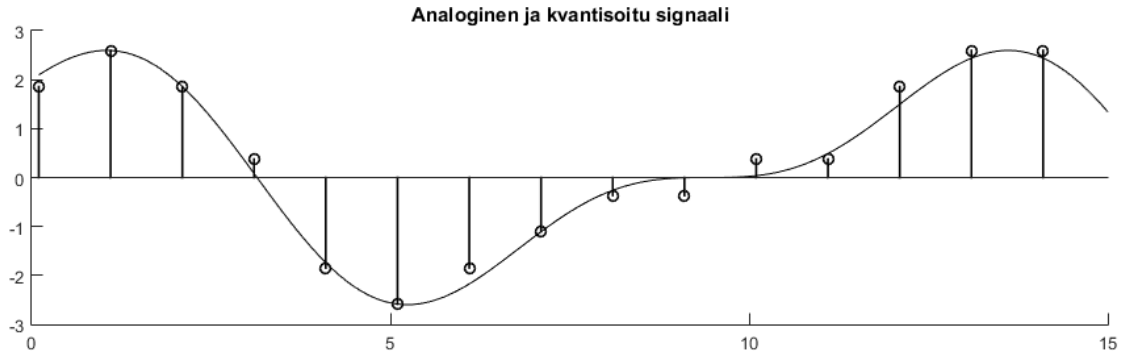
Kuva 2.3 Näytteenotto 1Hz taajuudella. Vasen kuvaaja on analoginen signaali ja oikeanpuoleinen kuvaaja on diskretoitu signaali.

Näytteenoton jälkeen diskreettiaikainen signaali kvantisoidaan tasoille, joista jokainen vastaa omaa binäärilukuaan. Kuvassa 2.4 diskreettiaikainen signaali on kvantisoitu kahdeksalle tasolle, eikä sen perusteella enää saisi rakennettua alkuperäistä signaalia. Kvantisoinnin jälkeen muodostetaan tasojen mukainen digitaalinen signaali.



Kuva 2.4 Kvantisointiin käytettiin kolme bittiä, eli tasoja saatiin kahdeksan.

Signaalin kvantisoinnissa hävittää aina informaatiota analogisen signaalin muodosta. Kuvasta 2.5 nähdään, että suurin osa kvantisoiduista pisteistä ei vastaa alkuperäistä analogista signaalia. Tarkkuutta voidaan kasvattaa huomattavasti lisäämällä kvantisointitasojen lukumäärää.



Kuva 2.5 Kvantisoitu signaali sekä alkuperäinen analoginen signaali.

2.2 Haitat ja hyödyt analogiseen verrattuna

Suurin etu digitaalisessa signaalin käsittelyssä analogiseen nähden on se, että digitaalista käsittelyä on mahdollista muokata suunnittelun jälkeen. Analogisessa signaalinkäsittelyssä piirit tuotetaan yleensä massana, jolloin hinta saadaan alhaiseksi. Jos massana valmistettu kappale ei kuitenkaan käy, sen muuttaminen tai yksittäisen kappaleen suunnittelu on kallista.

Digitaalinen käsittely on myös tarkempaa ja varmempaa kuin analoginen. Analogisessa käsittelyssä esimerkiksi lämpötilan vaihtelut piirissä voivat muuttaa saatua tulosta ja piireissä käytetyt komponentit voivat muuttua ajan ja käytön myötä. Digitaalisessa käsittelyssä ohjelma suorittaa aina samat laskennat samalla tavalla (Broge Jean, 2011).

Hintavertailussa digitaalisella laitteella saadaan paljon parempi suorituskyky kuin samanhintaisella analogisella laitteella (National Instruments1).

Korkean kertaluvun digitaaliset suodattimet ovat halvempia verrattuna analogisiin. Samalla on myös mahdollista tehdä suodattimista adaptiivisia. Suunnitellussa ei myöskään tarvitse huolehti analogisten komponenttien resistanssista, kapasitanssista ja induktanssista. Digitaalisia suodattimia ei myöskään tarvitse huoltaa.

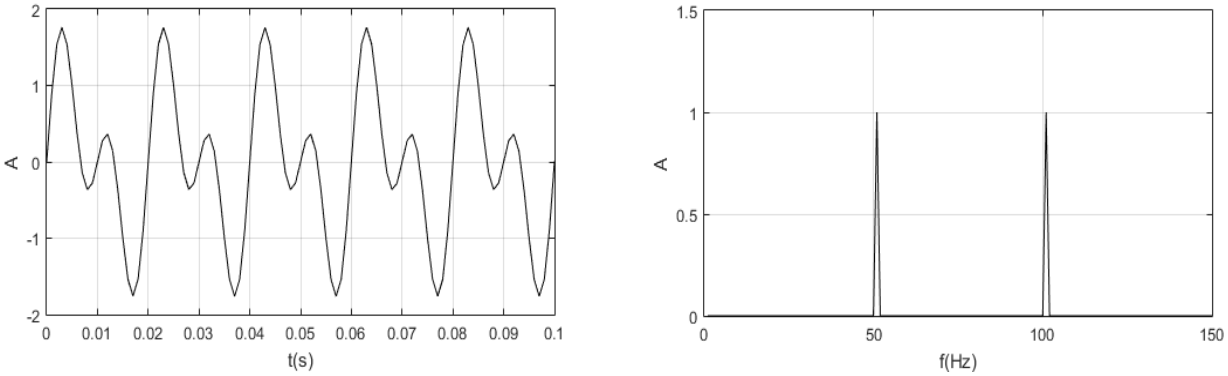
Toisaalta analoginen signaalinkäsittely voi olla digitaalista halvempaakin. Digitaalista käsittelyä varten on usein hankittava A/D ja D/A muuntimet sekä tietokone, tällöin yksittäisen analogisen piirin korvaaminen tulee todella kalliiksi (DePiero, 2013). Tämän takia yksinkertaiset operaatiot, kuten normaalit suodattimet, kannattaa kuitenkin suorittaa analogisesti, jollei signaalille tarvitse tehdä jatkokäsittelyä.

2.3 FFT

Kaikkien analogisten signaalien voidaan ajatella muodostuvan siniaalloista, joilla voi olla eri taajuudet, vaiheet sekä amplitudit. Aikatason esityksestä on mahdoton havaita yksittäisiä aaltoja ja signaalin käsittely on monissa tapauksissa hankalaa. Tämän takia signaalit muunnetaan taajuustasoon, jossa on helpompi käsitellä tietyn taajuisia signaalinosia. Taajuustasossa kuvataan, kuinka suuri osa signaalista on milläkin taajuudella ja se mahdollistaa tahdottujen siniaaltojen suodattamisen pois tutkittavasta signaalista.

Aikatasosta taajuustasoon ja taajuustasosta aikatasoon voidaan siirtyä käyttämällä FFT:tä. FFT eli nopea Fourier muunnos, on diskreetin Fourier muunnoksen laskemiseen kehitetty nopeampi ja tarkempi algoritmi (Schatzman, 1996). Fourier muunnos muuntaa signaalin aikatasosta taajuustasoon, jolloin signaalin käsittely voi usein olla helpompaa.

Kuvassa 2.6 on esimerkki signaalista aikatasossa ja taajuustasossa. Signaali koostuu kahdesta sini-funktioista, joiden amplitudit ovat 1 ja taajuudet 100 Hz ja 50 Hz. Taajuustason kuvassa funktiot näkyvät selkeästi omina piikkeinään vastaavilla taajuuksilla ja amplitudit ovat molemmilla 1.



Kuva 2.6 Vasemmalla $\sin(2\pi t \cdot 100) + \sin(2\pi t \cdot 50)$ aikatasossa ja oikealla sama funktio taajuustasossa.

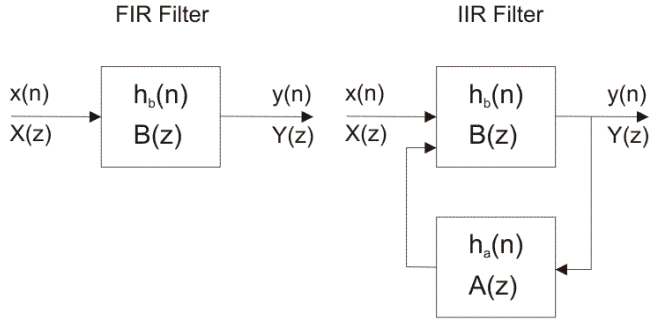
FFT muunnos on yksi tunnetuimmista ja eniten käytetyistä algoritmeista ja sitä käytetään hyväksi lukemattomissa käyttökohteissa. FFT analyysi on käytännöllinen mm. differenssiyhtälöiden ratkaisemiseen, matriisilaskentaan ja suodantinsuunnitteluun.

Signaalissa voi olla kohinaa, joka voidaan suodattaa pois taajuustasosta, kun tunnetaan kohinan taajuus. Monet kuva- ja ääni-formaatit tekevät pakkauksen poistamalla pieniä yksityiskohtia FFT analyysiä käyttäen. Tällöin kuvaan tai audioon jää ainoastaan oleellinen tieto ja tiedostokoko ovat huomattavasti raakadataa pienempi. Radiota käytettäessä haluttu kanava valitaan käyttäen taajuusanalyysiä.

2.4 Suodatus

Digitaalisissa suodattimissa suoritetaan matemaattista operaatiota diskreetille signaalille. Suodattimien suunnitteluun voidaan käyttää useita eri tapoja, jonka valintaan vaikuttaa suodattimen tyyppi sekä monimutkaisuus. Yksinkertaiset suodattimet on helppo laskea käsin, mutta monimutkaisempien suodattimien tekoon voidaan käyttää esimerkiksi MATLAB:n DSP system toolbox:n työkaluja tai vastaavia R:n funktioita.

Digitaaliset suodattimet voidaan jakaa kahteen luokaan FIR (ei-rekursiivisiin) ja IIR (rekursiivisiin) suodattimiin. Suurin ero tyyppien välillä on se, että IIR-suodattimissa on takaisinkytkentä. Takaisinkytkentä tekee IIR-suodattimien suunnittelusta paljon vaikeampaa kuin FIR tyyppisten, mutta se tekee IIR-suodattimista paljon tehokkaampia. Tehokkuuden ansiosta suodatin tarvitsee yleensä vähemmän laskutoimituksia ja muistia. Kuvassa 2.7 on esitetty yksinkertaiset suodattimet lohkokaavioina.



Kuva 2.7 FIR- ja IIR-suodattimien lohkokaavioesitys. Lohkoissa $h_b(n)$ kuvaa impulssivastetta ja $B(z)$ sekä $A(z)$ suodattimien siirtofunktioita. (Milivojević, 2009)

2.4.1 Liikkuva keskiarvo

Liikkuva keskiarvo on yksinkertaisin ja yleisin digitaalinen suodatin. Pienillä ikkunoilla se poistaa signaalista kohinaa ja suuremmilla se voi tuoda esiin signaalissa olevan pidemmän aikavälin trendin. Liikkuvan keskiarvon suodatin toimii hyvin aikatasossa, mutta taasjuustasossa se ei kykene poistamaan kohinaa. Taajuustasoon parempia vaihtoehtoja ovat muun muassa Gaussian ja Blackman suodattimet (Downey, 2016).

Yksinkertaisuuden puolesta se sopisi hyvin Raspberry Pi:llä suoritettavaksi. Mittaussignaalin käsittelyn jälkeen se voitaisiin esimerkiksi tallentaa tiedostoon dataloggerin tavoin.

Liikkuvaa keskiarvoa laskettaessa signaalin arvoja käydään läpi ja jokaisen arvon kohdalla lasketaan uusi keskiarvo käyttäen valitun ikkunan kokoista pisteryhmää. Liikkuvaa keskiarvo voidaan kuvata kaavalla

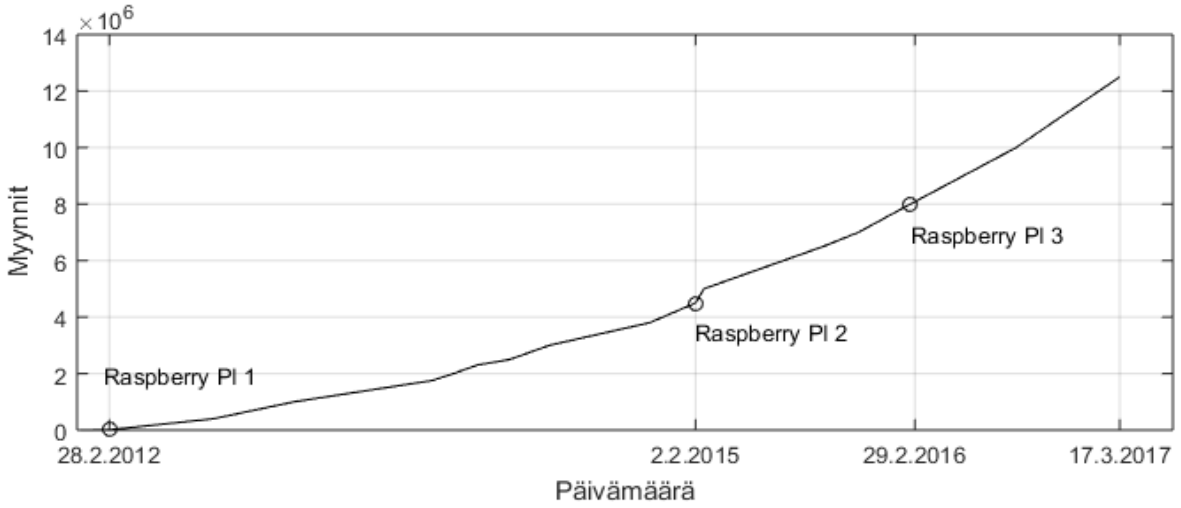
$$y[i] = \frac{1}{N}\sum_{j=0}^{N-1} x[i+j] \tag{1}$$

jossa N on keskiarvossa käytettävien pisteiden määrä, eli ikkunan koko, y lähtösignaali ja x tulosignaali. $x[i + j]$:ssä i kertoo kuinka monta signaalin jäsentä summataan ja j indeksiä.

Esimerkiksi jos i on 5 ja j on 3, summattaisiin tällöin indeksit $x[5]$, $x[6]$ ja $x[7]$. Pisteet voidaan valita myös symmetrisesti, jolloin summattavat indeksit olisivat $x[4]$, $x[5]$ ja $x[6]$, tällöin j:n on oltava pariton.

3. RASPBERRY PI

Raspberry Pi on yhden piirilevyn tietokone eli SBC. Raspberry Pi:n suosion myötä SBC:t ovat yleistyneet hurjasti viimeisten vuosien aikana. Raspberry Pi on kuitenkin selvästi tunnetuin valmistaja, joka tekee laitteita harrastekäyttöön. Harrastekäytön ohella laitteita käytetään muun muassa sulautetuissa järjestelmissä ja tietotekniikan opetusvälineinä. Kuvassa 3.1 on esitetty Raspberry Pi levyjen myyntilukemat julkaisusta asti (Raspberry Pi sales, 2017).



Kuva 3.1 Raspberry Pi A, B ja Zero yhteenlasketut myynnit.

Raspberry Pi suunniteltiin käytettäväksi Linux-käyttöjärjestelmän kanssa. Suurin osa käyttää joko Raspbian tai Pidora käyttöjärjestelmää, jotka ovat molemmat optimoitu Raspberry PI:lle (Opensource).

Vaikka Raspberry Pi suunniteltiin Linux mielessä pitäen, siinä voidaan silti käyttää muitakin käyttöjärjestelmiä. Esimerkiksi Windows 10:stä on tehty Windows 10 IoT Core – versio, joka on kevyempi kuin perusversio ja suunniteltu IoT laitteille (Health, 2015).

3.1 Tyypit

Ensimmäinen Raspberry Pi versio, Raspberry Pi 1 Model A, kehitettiin eritoten opetuskäyttöön eikä se vielä tarjonnut suurta määrää liitäntöjä. Malli keräsi kuitenkin kovasti suosiota harrastelijoiden keskuudessa ja Raspberry Pi sai kasvatettua ympärilleen suuren yhteisön. Nykyään tyyppejä on jo neljä ja versioita on yhteensä 13. Tässä työssä käytetty malli on Raspberry Pi 3 Model B, joka on uusin Model B – tyypin versio.

Raspberry PI on etenkin harrastelijoiden suosiossa, mutta sille on kysyntää myös kaupallisellakin puolella. Tämän johdosta Raspberry Pi on julkaissut erillisen laskentayksikön, jossa suurimpana lisäyksenä on 46 GPIO:ta.

Raspberry Pi:stä on myös minimalistisen tyypin, Zero. Zeron parhaimpina puolina on sen hinta, joka on vain 5 dollaria. Zeron perus versiossa ei ole Bluetooth- tai verkkoyhteyttä, mutta suosion ja pyyntöjen myötä siitä tuli versio, joka pitää ne sisällään. Tällöin Zeron hinnaksi tulee 10 dollaria.

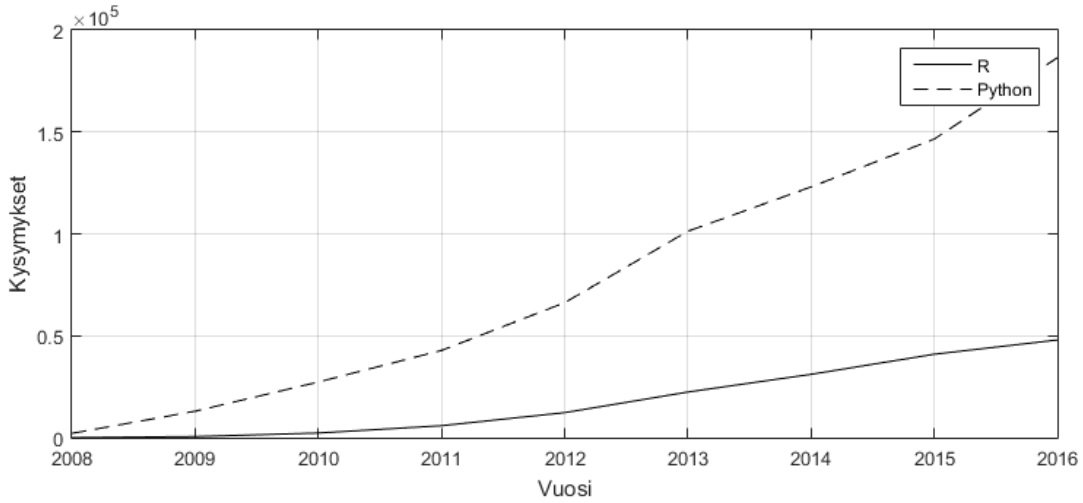
Kokojen puolesta kaikki tyypit ovat pieniä. Model A ja Model B ovat 85.6 mm x 56.6 mm x 17 mm ja Zero on 65 mm x 30 mm x 5 mm.

Signaalinkäsittelyn kannalta Raspberry Pi 3 Model B on levyistä tehokkain, se sisältää 1 GB muistia ja 1.2 GHz 64-bit neljä ytimisen ARM Cortex-A53 prosessorin.

4. OHJELMISTOT JA KIELET

Signaalinkäsittelyyn on olemassa valmiita kaupallisia työkaluja kuten LabVIEW ja MATLAB. Nämä työkalut ovat kuitenkin liian raskaita Raspberry Pi:llä, eikä niitä voi käyttää. Molemmat tarjoavat kuitenkin mahdollisuuden koodin kääntämiselle Raspberry Pi:llä käytettäväksi. Viime vuosina kaupallisten signaalinkäsittely-työkalujen rinnalle on noussut Python sekä R. (Tjoa, 2010; Sachs, 2013)

Molempien ohjelmointikielten suosiota kasvattaa kaupallisten lisenssien hinta, Python ja R ovat molemmat ilmaisia. Ilmaisuuden lisäksi ohjelmointikielet ovat kehittyneet kovasti ja niiden ympärille on rakentunut yhteisöt, jotka kehittävät kielille lisää ilmaisia kirjastoja. Kuvassa 4.1 näkyy Pythonin ja R:n kasvava suosio.



Kuva 4.1 Pythonin ja R:n suosio. Y-akseli kuvaa avoimia kysymyksiä stackexchange-sivustolla. Tiedot on haettu stackexchangen tietokannasta.

4.1 MATLAB, Simulink ja LabVIEW

MATLAB:a ei voi suoraan käyttää Raspberry Pi:llä, mutta sitä voi käyttää etänä. Tietokone ja Raspberry Pi voidaan yhdistää toisiinsa Ethernet verkkokaapelilla tai WiFi:llä, jolloin tietokoneella olevaa MATLAB:a voidaan käyttää Raspberry Pi:n sensoreista saatavan datan käsittelyyn.

Jos toisen tietokoneen käyttö ei ole mahdollista, algoritmien tekemiseen käyttää Simulink:iä Simulink:ssä luoduista vuokaavioista voidaan generoida koodi, joka toimii Raspberry Pi:ssä. Tarvittaessa datan voi ladata ThigsSpeak datapankkiin, josta ne voidaan ladata MATLAB:iin jatkokäsittelyä varten.

MATLAB:a ja Simulink:iä käyttämällä voidaan olla varmoja työkalujen laadusta ja tuesta. Molempien käyttöä varten on ladattava paketit MATLAB:iin, jotka ovat ilmaisia (MathWorks).

LabVIEW on Simulink:n tapainen graafinen ohjelmointikieli, jolla tehdyt ohjelmat voidaan kääntää Raspberry PI:llä toimiviksi. Tätä varten LabVIEW:iin täytyy ostaa maksullinen lisäosa. LabVIEW:n graafisen käyttöliittymän käyttö vaatii Raspberry Pi:n ja tietokoneen välille ethernet yhteyden (National Instruments2).

Valmiit ohjelmistot täyttävät usein kaikki tarpeet, mutta niiden käyttö on kallista. Raspberry Pi valitaan käyttökohteeseen usein muun muassa sen halvan hinnan puolesta, eikä kalliita ohjelmistoja voida ostaa. Ethernet yhteys toiseen tietokoneeseen ei aina myöskään ole mahdollinen, joka sulkee pois MATLAB:n ja LabVIEW:n.

4.2 R

R on tilastolaskentaan kehitetty kieli, josta löytyy kaikki MATLAB:n perus ominaisuudet. Kielessä on myös valmiit funktiot muun muassa FFT:n ja konvoluution laskuun, jotka ovat tarpeellisia signaalinkäsittelyn kannalta. R kielen sisällä on mahdollista käyttää C, C++ sekä Fortran kieliä, kun tarvitaan lisää optimointia (Project-R).

Jos R:stä tuntuu puuttuvan jotain toiminnollisuuksia, ne voi usein löytyä ilmaisista paketeista. CRAN on R-kirjastojen varasto, josta löytyy yli 10 000 kirjastoa. Varastosta löytyy kirjasto tilanteeseen kuin tilanteeseen, mutta signaalinkäsittelyssä hyödyllisiä ovat muun muassa nämä.

- Signals: Signal Processing. Suodattimien suunnitteluun ja testaukseen sekä visualisointiin ja interpolointiin käytettäviä funktioita, jotka kirjoitettiin alun perin MATLAB:n Signal Processing Toolbox:lle ja Octavelle. Kirjastosta löytyy muun muassa suodatinsuunnittelussa käytettävät *butter, cheby1* sekä *ellip*. Testaamiseen käytettäviä funktioita kuten *chirp, filter, filtfilt* sekä *freqz*.

- R.Matlab. Kirjaston funktioiden avulla R voi keskustella ja vaihtaa dataa MATLAB:n kanssa. Funktioissa käytetään MATLAB:n syntaksia. Kirjasto tarvitsee toimiakseen lokaalin tai etäyhteydessä olevan MATLAB v6 version tai uudemman. Funktioiden käyttöä varten MATLAB:iin on asennettava lisäosa.

- FFTW. Wrapperi C-kielellä toteutetulle FFTW-kirjastolle joka on yksi nopeimmista ilmaisista Fourier muunnoksen laskevista algoritmeista (Frigo 2014).

4.3 Python

Python sekä siitä tarvittavat kirjastot ovat ilmaisia käyttää ja monet kirjastojen funktioista vastaa hyvin pitkälti MATLAB-funktioita. Raspberry Pi:tä ja Pythonia käytettäessä säästytään myös paljolta säädöltä, sillä Python-skriptit voidaan ajaa vaivatta Linux pohjaisissa käyttöjärjestelmissä.

Pythonilla on tehty lukuisia signaalin käsittelyä helpottavia kirjastoja. Kirjastojen asennuksesta on tehty entistä helpompaa valmiiden pakettien avulla, joita ovat muun muassa Anaconda ja SciPy. Anaconda on näistä paketeista suurempi ja se sisältää yli sata kirjastoa Pythonille, Scalelle ja R:lle, monet kirjastoista ovat kuitenkin signaalin käsittelyssä turhia. Raspberry Pi:n rajatun muistin takia on hyvä valita pienempi SciPy, johon on koottu kaikki tärkeimmät kirjastot.

SciPy:n sisältämät kirjastot ovat NumPy, SciPy Library, Matplotlib, Sympy sekä pandas ja signaalin käsittelyn kannalta tärkeimmät ovat NumPy ja SciPy Library.

- NumPy mahdollistaa MATLAB-tyyppisten matriisien käytön pythonissa ja sisältää pitkän listan erilaisia lineaari-algebraan sekä Fourier-muunnoksiin liittyviä funktioita.

- SciPy Library puolestaan sisältää monia numeerisia funktioita integraalien laskemiseen, differentiaaliyhtälöiden ratkomiseen sekä käyrien sovitukseen sekä optimointiin.

- Matplotlib mahdollistaa kaikkien tavanomaisten ja ei niin tavanomaisten kaavioiden piirtämisen.

- Sympy:n avulla voidaan käsitellä symbolisia matemaattisia yhtälöitä.

- pandas tuo Pythoniin joukon datankäsittelyssä hyödyllisiä funktioita, joita voidaan hyödyntää niin perus matriiseissa, kuin SQL-tauluissakin.

Näillä edellä mainituilla ilmaisilla kirjastoilla on mahdollista korvata monet MATLAB:n työkalut. Jos kirjastojen sisältö ei riitä, voidaan Pythonissa käyttää R:ää sekä siten myös sen kirjastoja rpy2-kirjastoa käyttäen.

5. KÄYTTÖKOHTEITA

Raspberry Pi:n käyttö kaupallisella puolella on vielä alussa eikä siitä ole tehty kattavia tutkimuksia. Laitteen omilla foorumeilla useat käyttäjät ovat kertoneet käyttävänsä laitetta digitaalisten kylttien näyttämiseen. Ennen Raspberry Pi:tä kylttien ylläpito oli kallista eikä pienillä yrityksillä ollut niihin varaa. Nyt monta suurta näyttöä voidaan liittää samaan halpaan Raspberry Pi:hin kalliin tietokoneen sijaan.

Toinen yleiseksi muodostunut käyttökohde on dataloggeri. Dataloggereita käytetään mitatun datan tallentamiseen pitkällä aikavälillä. Lappeenrannan Teknillisessä Yliopistossa toteutetussa testissä Raspberry Pi:tä käytettiin taajuusmuuntajan tietojen lukuun käyttäen Modbus-protokollaa. Raspberry Pi havaittiin tarpeeksi tehokkaaksi pitkäkestoisessa mittauksessa ja tarkkoja tuloksia saatiin yksinkertaisella ohjelmalla vielä 0.1…1 sekunnin näytteistysajoillakin. (Kontkanen, 2016)

Dataloggauksesta löytyy esimerkki myös kaupalliselta puolelta, jossa Arduinolla mitattiin serverihuoneen lämpötilaa ja Raspberry Pi:llä kirjattiin mittaustulos pilveen. Järjestelmän hyviksi puoliksi todettiin sen halpuus, asennettavuus, skaalautuvuus ja muokattavuus (Ferdoush 2014).

Rooman ulkopuolella testattiin Raspberry Pi:tä risteyksen katuvalojen saarekekäytön pääyksikkönä. Risteys valittiin testiin, koska sinne ei saatu kaupungilta normaalin tavoin verkkoyhteyttä. Pääyksikkö valvoi valojen tilaa sekä lähetti niistä tietoja WiMAX-verkon kautta keskukselle, joka mahdollistaa yhteyden jopa 70km päähän. Raspberry Pi valittiin sen tehokkuuden, halvan hinnan, lisälaitteiden sekä virrankulutuksen perusteella. Valot kytkettiin päälle, kun sensoreilla havaittiin lähestyvä jalankulkija tai auto. Helmikuussa tehdyn testin aikana kaikki tarvittava energia tuotettiin aurinkopaneeleilla. (Leccese, 2014)

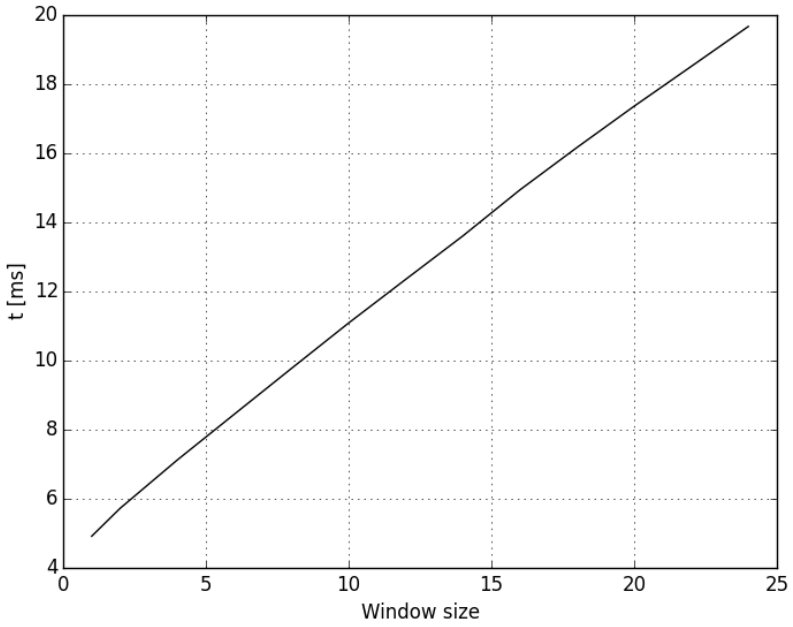
Yllämainittujen käyttökohteiden puolesta voidaan todeta, että Raspberry Pi sopii hyvin moniin käyttötarkoituksiin sen halvan hinnan, pienen virrankulutuksen, tehon sekä joustavuuden ansiosta. Laite voi olla hyvä vaihtoehto, kun toimitaan pienellä budjetilla ja käyttökohteessa tarvitaan perustietokonetta. Raspberry Pi:ssä on myös omia IO-pinnejä, joita voidaan käyttää suoraan eri sensoreiden kanssa.

5.1 Kohinanpoisto

Raspberry Pi:tä testattiin kohinanpoistossa laskemalla 100 signaalille liikkuva keskiarvo 11 eri ikkunakoolla ja tutkimalla laskenta-aikoja. Samalla tutkittiin, kuinka tehokkaasti kohina poistuu, laskemalla alkuperäisen kohinattoman signaalin sekä suodatetun signaalin välinen virhe käyttäen neliöllistä keskiarvoa

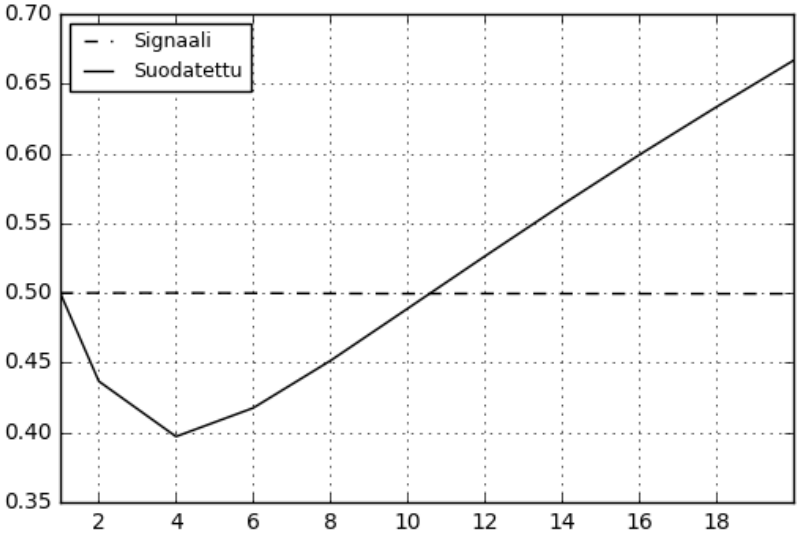
$$e_{RMSE} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\big(x(n) - y(n)\big)^2} \quad , \tag{2}$$

jossa N on datapisteiden lukumäärä. Signaalin pituutena käytettiin 300 datapistettä ja se sisälsi kolme eripituista pulssia, joiden amplitudit olivat satunnaiset. Signaali luotiin pulsseista, koska nopeat muutokset tekevät ikkunakoon valinnasta vaikeampaa. Signaaliin lisättiin kohinaa normaalijakauman mukaan käyttäen odotusarvona 0 ja varianssina 0.5. Testiin käytetty koodi on liitteessä 1 ja kuvassa 5.1 on esitetty laskenta-aikojen keskiarvo ikkunakoilla.



Kuva 5.1 Raspberry Pi:llä suoritettujen laskentojen keskiarvot ikkunakokoa kohden.

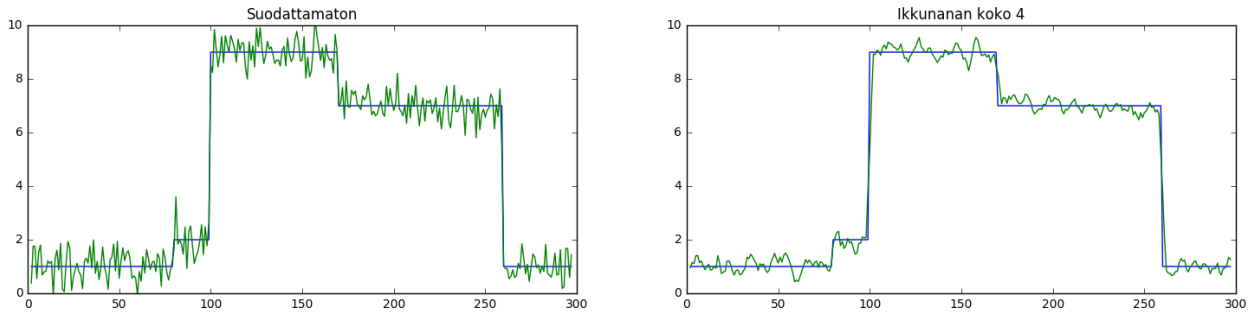
Kuvasta nähdään laskenta-ajan kasvavan lineaarisesti ikkunakoon mukaan kulmakertoimella 0.65. Suuremmilla ikkunoilla laskenta-aika on siis suhteellisesti pienempi, kuin pienemmillä ikkunoilla. Ikkunoita vastaavat virheet ovat esitetty kuvassa 5.2.



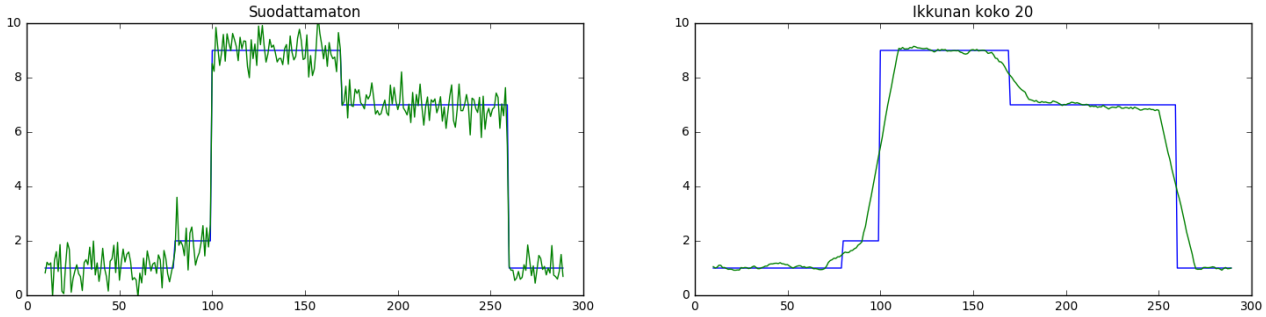
Kuva 5.2 Virhe ikkunakoon-funktiona kohinaiselle signaalille sekä suodatetulle signaalille eri ikkunoilla. Katkoviiva on tasaisesti 0.5 kohdalla, koska sitä käytettiin kohinan normaalijakauman hajontana.

Kuvaajasta voidaan todeta, että optimaalinen ikkunankoko on tässä tapauksessa 4. Vaikka suuremmilla ikkunoilla kohina poistuu paremmin, huomataan että ikkunakoon ollessa 11 tai suurempi, pulssin reunojen vääristyminen vääristää signaalia enemmän kuin kohinanpoisto pystyy sitä selventämään. Ikkunakoolla 4 Raspberry Pi käytti laskentaan noin 7 millisekuntia, joten se kykenisi laskemaan sen helposti jokaisen näytteen yhteydessä, jos näytteenottotaajuutena olisi 1 Hz.

Ikkunakoolla 4 suodatettu signaali on esitetty kuvassa 5.3 ja ikkunakoolla 20 suodatettu on esitetty kuvassa 5.4.



Kuva 5.3 Sininen signaali kuvaa oikeaa, kohinatonta signaalia ja vihreä kohinallista signaalia. Suodatus tehtiin ikkunakoolla 4. Pienelläkin ikkunalla on suuri vaikutus kohinaan.



Kuva 5.4 Sininen signaali kuvaa oikeaa, kohinatonta signaalia ja vihreä signaali kohinallista signaalia. Suodatus tehtiin ikkunankoolla 20. Oikean puoleisesta kuvaajasta nähdään, että kohina on hävinnyt lähes kokonaan, mutta pulssin reunat ovat todella vääristyneet. Jos signaalissa ei olisi teräviä reunoja, olisi suurempi ikkunakoko parempi valinta.

Koska laskenta-aika on suurilla ikkunakoilla suhteessa pienempi kuin pienillä ja kohinanpoisto tehostuu suurilla ikkunakoilla, kannattaa valita optimaalinen koko, jos siihen on mahdollisuus.

6. SUORITUSKYKY

DSP-laitteille on olemassa tilanteista riippuen erilaisia suorituskykytestejä. Testit voidaan jakaa neljään ryhmään.

- Applikaatiotesteissä DSP-laitteen tehokkuutta tutkitaan oikeassa käyttökohteessa. Tämä on paras tapa varmistaa, että sovellus toimii käytännössä. Emuloiduissa tilanteissa ei voida aina ottaa kaikkia rasituksia huomioon.
- Synteettisissä testeissä emuloidaan jotain DSP-laitteen käyttökohdetta. Emuloidessa otetaan huomioon käyttökohteessa olevat rasitukset.
- Kernelitesteissä tutkitaan yhtä funktiota tai algoritmia, jota käytetään käyttökohteessa.
- Teknologiatesteissä tutkitaan tiettyjen laitteiden välisiä erilaisuuksia.

(Stefani, 2004)

Suurin vaikutus suorituskykyyn on prosessorin kellotaajuus. Kellotaajuus kuvaa sitä, kuinka monta käskyä sekunnissa prosessori kykenee suorittamaan. Joissakin tilanteissa ytimien sekä RAM muistin määrä ja laatu voivat myös vaikuttaa suorituskykyyn. Jos signaalinkäsittelyssä käytetty funktio olisi optimoitu usealle ytimelle, kasvaisi suorituskyky selvästi, jos ytimiä olisi optimoitu määrä tai enemmän. Ilman optimoitua funktiota ydinten lukumäärä ei juuri vaikuta suorituskykyyn. RAM muistissa tärkeintä on sen riittävyys. Tietokone alkaa käyttää kovalevyä muistintalletukseen automaattisesti, jos RAM muistista loppuu tila. Näin ollen laskennassa tarvittavien arvojen hakeminen muistista hidastuu.

Raspberry Pi 3 Model B:ssä on muistia 1Gt, joka riittää hyvin suurimpaan osaan perussovelluksista. Vaikuttavin tekijä tulee olemaan sen prosessori, ARM Cortex–A5, jonka kellotaajuus on 1.2GHz

6.1 Kernelitesti

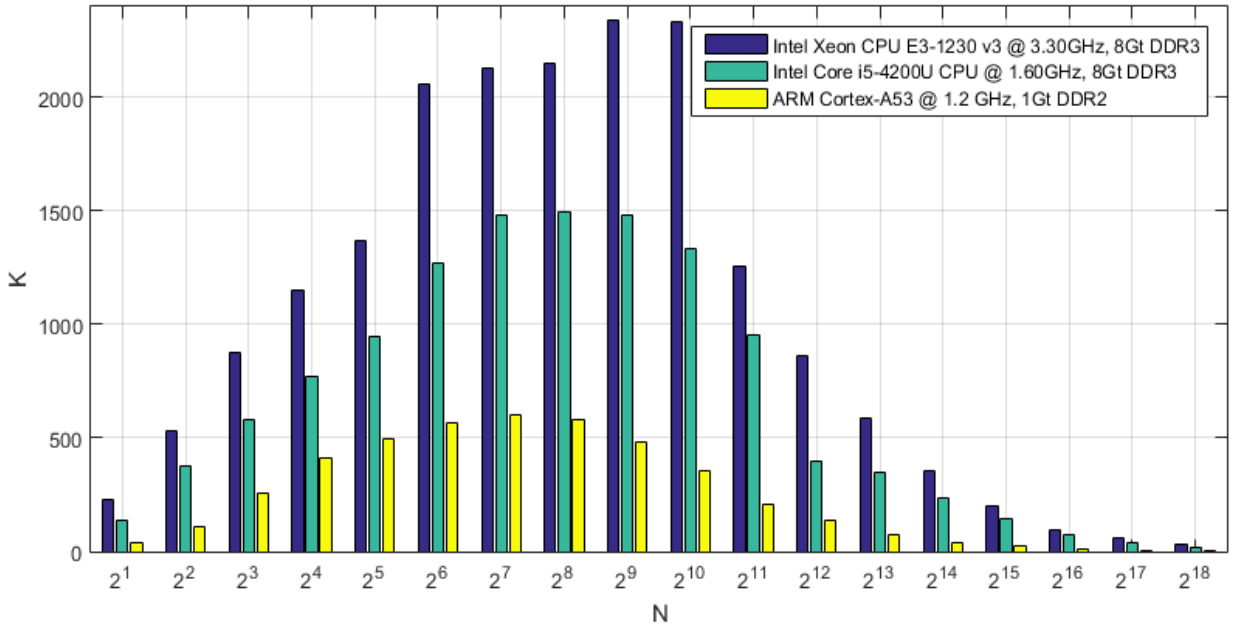
Helppo tapa testata Raspberry Pi:n tehokkuutta on suorittaa kernelitesti. Testin tuloksena saadaan yksi numero, jolla kuvataan tehokkuutta. Luvun suuruus kuvaa laitteen tehokkuutta, mitä suurempi luku, sitä paremmin laite suoriutui testistä. Testi on myös helposti toistettavissa sekä siirrettävissä laitteelta toiselle.

Käytettäväksi kernelitestiksi valittiin FFT-suorituskykytesti, joka valittiin FFT:n yleisen käytön ja yksinkertaisuuden vuoksi. Testissä tehdään Fourier-muunnos yksi ulotteiselle vektorille, joka sisältää N kappaletta reaalilukuja. Laskentanopeutta kuvaava luku K lasketaan kaavan 1 mukaisesti, jossa T on FFT-muunnoksessa kulunut aika mikrosekunteina ja N datasetin pituus (Frigo, 2014).

$$K = \frac{5 \cdot N \cdot \log_2(N)}{T} \tag{3}$$

Yksinään testin luvut eivät kerro paljoa, tämän takia testi toteutettiin Raspberry Pi:n lisäksi myös kahdella toisella laitteella. Liitteessä 2 on kaikissa laitteissa käytetty Pythonilla tehty skripti.

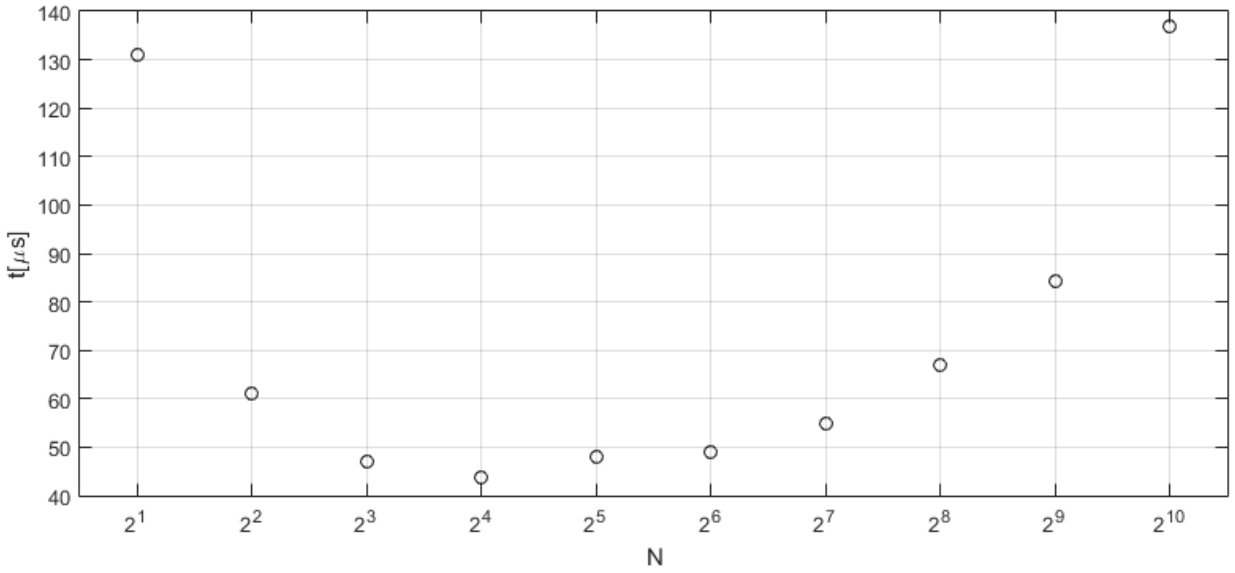
Testissä käytettiin scipy-kirjastosta löytyvää FFT-funktiota ja muunnettavan datasetin pituus tuplattiin jokaisen kierroksen alussa. Testin tulokset ovat verrannollisia vain toisiin samalla funktiolla tehtyihin testeihin, koska tulokset riippuvat paljon funktion optimoinnista. Testi ajettiin laitteilla viisi kertaa ja tulokseksi valittiin jokaisen kierroksen paras tulos. Saadut tulokset on esitetty kuvassa 6.1.



Kuva 6.1 FFT-suorituskykytestin tulokset. Keltaisella Raspberry Pi 3 Model B.

Testin tuloksista nähdään, että Raspberry Pi häviää jokaisessa testissä vertailulaitteille. Kun otetaan huomioon verrattavien prosessorien hinta ja muistin laatu ja määrä, voidaan todeta, että Raspberry Pi suoriutui testistä hyvin. Testistä käy ilmi myös se, että yksi ulotteisten reaalisten datajoukkojen tapauksessa tehokkain FFT:n lasku tapahtuu silloin, kun datapisteitä on 2^7 eli 128.

kuvassa 6.2 on esitetty pisteiden 2^1-2^{10} laskentoihin kuluneet ajat Raspberry Pi:llä. Aikojen perusteella Raspberry Pi voisi toimia hyvinkin yksi ulotteisten reaaliarvoisten datapisteiden Fourier muunnosten laskemiseen reaaliajassa, kun pieni viive sallitaan.



Kuva 6.2 Raspberry Pi:n FFT-suorituskykytestin laskentaan käytetty aika mikrosekunteina.

Laskentaan kulunut aika kasvoi lähes lineaarisesti, kun datapisteitä oli 2^{10} tai enemmän. Laskuajaltaan 2^2-2^8 ovat hyvin samaa luokkaa. Mielenkiintoista on myös se, että 2^1 laskuaika oli alkupään suurin, vasta 2^{10}:llä oli pidempi laskuaika. Ensimmäisen muunnoksen hitaus johtuu luultavasti pythonin suorittamasta käännöksestä.

7. POHDINTA

Työssä jäi tutkimatta ja testaamatta muun muassa käyttöjärjestelmien, ohjelmointikielien sekä monien optimoitujen kirjastojen käyttö. Reaaliaikaisessa signaalinkäsittelyssä pienelläkin suorituskyvyn parannuksella voi olla suuri vaikutus.

Windows 10 IoT core julkaistiin samaan aikaan kun työtä tehtiin, eikä sitä testattu. IoT core:ssa voi olla merkittäviä ominaisuuksia, jotka tekevät siitä merkittävän vaihtoehdon Raspberry Pi:n käyttöjärjestelmäksi. Käyttöjärjestelmän vaihto toisi lisäksi joukon uusia käytettäviä kieliä, kuten C#, jota olisi ollut mielenkiintoista kokeilla signaalinkäsittelyssä.

Työn aikana selvisi, että R-kieli mahdollistaa usean hyödyllisen MATLAB-funktion käytön Linuxilla. Käytettäväksi kieleksi valittiin kuitenkin Python aiemmat kokemuksen puolesta, mutta tutkimuksen R olisi voinut olla erittäin hyvä vaihtoehto. Linuxilla myös C olisi voinut olla hyvä vaihtoehto. Raspberry Pi:llä olisi voinut kokeilla myös ARM Assembleria, joka olisi ollut mielenkiintoista sekä todennäköisesti tuottanut hyviä tuloksia reaaliaikaisen signaalinkäsittelyn puolesta.

Pythonissa on monia kirjastoja, kuten FFTW, jotka olisivat voineet vaikuttaa tuloksiin. Yksinkertaisuuden vuoksi suurempien kirjastojen valinta kuitenkin kannatti, sillä optimaalisten kirjastojen valinta olisi vaatinut paljon aikaa ja testailua.

8. YHTEENVETO

Työn tavoitteena oli selvittää, kuinka tehokas Raspberry Pi on signaalinkäsittelyssä, sekä minkälaiseen käsittelyyn sitä voitaisiin käyttää. Kirjallisuudesta etsittiin myös toteutettuja sovellusesimerkkejä laitteen käytöstä. Samalla tutkittiin mahdollisia koodikieliä sekä valmiita ohjelmistoja, jotka helpottaisivat signaalinkäsittelyä. Raspberry Pi:tä verrattiin muihin laitteisiin sen sisältämän arkkitehtuurin perusteella.

Käytännöntestien kautta Raspberry Pi osoittautui melko hyväksi signaalinkäsittely. FFT-testin tuloksien mukaan laiteella kesti keskimäärin 44 μs yhden muunnoksen laskemiseen, kun datajoukon koko oli 2^4. Muunnoksen lisäksi ohjelma suorittaa usein muitakin toimintoja, joten todellisuudessa yhdessä syklissä kuluu kauemmin. Tämä ei kuitenkaan haittaa, jos sovelluksessa sallitaan pieni viive.

Raspberry Pi:n yleinen käyttökohde on dataloggeri. Laite on todettu toimivaksi pitkäkestoisessa mittauksessa, kun taajuusmuuntajalta luettiin Modbus-väylän kautta 0.1…1 sekunnin syklillä. Raspberry Pi:tä voitaisiin myös käyttää hyvin liikkuvan keskiarvon suodattimella, jolla voidaan lieventää mittaussignaalissa esiintyvää kohinaa.

Raspberry PI:lle ei voida asentaa MATLAB tai LabVIEW ohjelmistoja, mutta molempia voidaan käyttää ethernet-yhteyden välityksellä. Simulink-mallit voidaan siirtää Raspberry Pi:lle, eikä niiden käyttöön tarvita toista tietokonetta. Jokaisen ohjelmiston käyttöä varten on asennettava ohjelmistopaketti vastaavaan ohjelmistoon. LabVIEW:n paketti osoittautui maksulliseksi.

Linux-käyttöjärjestelmässä signaalinkäsittelyn parhaiksi kieliksi todettiin Stackexchangen tietokannan sekä niiden kirjastojen perusteella Python ja R. Kieliin on olemassa valmiita kirjastoja, jotka sisältävät tärkeimmät käsittelyssä tarvittavat funktiota. R:n kirjastoista löytyy jopa MATLAB:n signaalinkäsittely-funktioita. Stackexchangen tulos kertoo kielien saamasta tuesta.

Tutkimuksia tehdessä laitteesta ei löydetty suuria puutteita, ja se osoittautui hyväksi vaihtoehdoksi pieniin signaalinkäsittelyn tehtäviin, joissa ei tarvita suurta laskentatehoa.

LÄHTEET

Broge Jean L., 2011, The benefits and advantages of digital signal processing [Verkkodokumentti] [Viitattu 28.3.2017] Saatavissa http://articles.sae.org/14409/

DePiero Fred, 2013, DSP Notes: Digital vs. Analog Signal Processing [Verkkodokumentti] [Viitattu 28.3.2017], Saatavissa https://courseware.ee.calpoly.edu/~fdepiero/fdepiero_dsp_notes/notes%20&%20materials/dsp_analog_vs_digital.pdf

Downey, A.B., 2016. Think DSP: digital signal processing in Python. " O'Reilly Media, Inc.".

Ferdoush, S. and Li, X., 2014. Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications. Procedia Computer Science, 34, pp.103-110., Saatavissa http://www.sciencedirect.com/science/article/pii/S1877050914009144

Frigo Matteo, Johnson Steven G., 2014, MIT, BenchFFT, [Viitattu 28.3.2016], http://www.fftw.org/speed/

Health Nick, 2015, Windows 10 on the Raspberry PI: What you need to know [Verkkodokumentti], [Viitattu 25.03.2017], Saatavissa http://www.techrepublic.com/article/windows-10-on-the-raspberry-pi-what-you-need-to-know/

Kontkanen, J., 2016. Raspberry Pi-pohjainen dataloggeri, Saatavissa: http://www.doria.fi/bitstream/handle/10024/124734/kandidaatinty%F6_kontkanen_joonas.pdf;jsessionid=C652CE6073D5BC37A53E8193C2BCB447?sequence=2

Leccese, F., Cagnetti, M. and Trinca, D., 2014. A smart city application: A fully controlled street lighting isle based on Raspberry-Pi card, a ZigBee sensor network and WiMAX. Sensors, 14(12), pp.24408-24424.

MathWorks, Raspberry Pi Programming with MATLAB and Simulink [Verkkodokumentti], [Viitattu 25.3.2017], https://se.mathworks.com/discovery/raspberry-pi-programming-matlab-simulink.html

MILIVOJEVIC, Z., 2009. Digital filter design. MikroElektronika.

National Instruments1, Advantages of Digital Filtering Compared to Analog Filtering [Verkkodokumentti] [Viitattu 28.3.2017], https://zone.ni.com/reference/en-XX/help/371361J-01/lvanlsconcepts/lvac_adv_dig_filt/

National Instruments2, LabVIEW for Raspberry PI [Verkkodokumentti], [Viitattu 30.3.2017], https://www.tsxperts.com/labviewforraspberrypi/

Nebeker, N., 1998. Fifty Years of Signal Processing: The IEEE Signal Processing Society and its Technologies 1948-1998. The IEEE Signal Processing Society.

Opensource, What is Raspberry PI [Verkkodokumentti], [Viitattu 25.3.2017], Saatavissa https://opensource.com/resources/what-raspberry-pi

Project-R, [Viitattu 29.3.2017], https://www.r-project.org/about.html

Raspberry Pi sales, 2017, https://docs.google.com/spreadsheets/d/1zWwpcck-DEEVAhNH3y7JQGxxbjP42nUywPOzDWr1fH28/edit#gid=0

Sachs Jason, 2013, Adventures in Signal Processing with Python [verkkodokumentti] [Viitattu 11.12.2016]. Saatavissa https://www.embeddedrelated.com/showarticle/197.php

Schatzman, J.C., 1996. Accuracy of the discrete Fourier transform and the fast Fourier transform. SIAM Journal on Scientific Computing, 17(5), pp.1150-1166. Saatavissa https://www.researchgate.net/publication/2788563_Accuracy_Of_The_Discrete_Fourier_Transform_And_The_Fast_Fourier_Transform

Stefani, F., Moschitta, A., Macii, D. and Petri, D., 2004. FFT benchmarking for digital signal processing technologies. University of Trento. Saatavissa http://www.academia.edu/5776227/FFT_BENCHMARKING_FOR_DIGITAL_SIGNAL_PROCESSING_TECHNOLOGIES

Tjoa Steve, 2010, I used Matlab. Now I use Python. [verkkodokumentti] [Viitattu 11.12.2016]. Saatavissa https://stevetjoa.com/305/

LIITTEET

Liite 1. liikkuvan keskiarvon koodi

```
%matplotlib inline
from IPython.core.pylabtools import figsize
import numpy as np
import matplotlib.pyplot as plt
from timeit import default_timer as timer

windows = [1,2,4,6,8,10,12,14,16,18,20]

signalLen = 300
noiseMax = 0.5
numberOfTests = 100
numberOfWindows = len(windows)

TableRmseSignal = np.ones((numberOfTests,numberOfWindows))
TableRmseAverage = np.ones((numberOfTests,numberOfWindows))
TableTimes = np.ones((numberOfTests, numberOfWindows))

#Four was the optimal window, saving result for later plotting
filtered_4 = np.ones(signalLen-4)
ty_4 = np.ones(signalLen-4)
X_4 = np.ones(signalLen-4)
Signal_4 = np.ones(signalLen-4)

#Calculating rmse for different signal
for signalNum in range (0,numberOfTests):
    x = np.ones(signalLen)

    #Creating the random signal
    randInts = np.random.randint(1,10,size=3)
    for i in range(0,signalLen):
        if(i >= 80 and i < 100):
            x[i] = x[i] * randInts[0]
        elif(i >= 100 and i < 170):
            x[i] = x[i] * randInts[1]
        elif(i >= 170 and i < 260):
            x[i] = x[i] * randInts[2]

    #Adding noise to the signal
    noise = np.random.normal(0,noiseMax,signalLen)
    signal = x + noise

    #Calculating averages for different windows and rmsqe
    for windowIndex in range(0, numberOfWindows):
        window = windows[windowIndex]
```

```
#Averaged signal and its time-indexes
y = np.ones(signalLen-window)
t_y = np.ones(signalLen-window)

Start = Timer()

# 1/M * sum x[i+j]
for j in range(0,signalLen-window):
    sum = 0
    for i in range(0,window):
        sum = sum + signal[j+i]
    y[j] = sum / window
    t_y[j] = j + window / 2

Elapsed = (Timer() – Start) * 10 ** 3

#The original signal and the noise-signal are trimmed to match vector sizes
X = np.ones(signalLen-window)
Signal_Window = np.ones(signalLen-window)

for i in range(0,signalLen-window):
    X[i] = x[int(t_y[i])]
    Signal_Window[i] = signal[int(t_y[i])]
sum = 0

#Saving data for plotting
if(window == 4):
    filtered_4 = y
    ty_4 = t_y
    X_4 = X
    Signal_4 = Signal_Window

#Calculating the errors
rmseSignal = np.sqrt(np.mean((X-Signal_Window)**2))
rmseFiltered = np.sqrt(np.mean((X-y)**2))

#Saving the results
TableRmseSignal[signalNum,windowIndex] = rmseSignal
TableRmseAverage[signalNum,windowIndex] = rmseFiltered
TableTimes[signalNum, windowIndex] = Elapsed
```

Liite 2. FFT-testin koodi

```
import numpy as np
from scipy import fftpack
from timeit import default_timer as timer

Datapoints = 1

for i in range(1,19):
    Datapoints = Datapoints * 2
    Arr = np.random.rand(Datapoints,1)

    Start = timer()
    f = fftpack.fft(Arr)
    Elapsed = (timer() - Start) *  10 ** 3

    K = 5 * np.log2(Datapoints)/Elapsed
```