OULUN SEUDUN
AMMATTIKORKEAKOULU

Tommi Tauriainen

LAAJENNETTAVA MITTAUSJÄRJESTELMÄ PULSSIANTUREILLE

LAAJENNETTAVA MITTAUSJÄRJESTELMÄ PULSSIANTUREILLE

Tommi Tauriainen
Opinnäytetyö
Kevät 2011
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, elektroniikan suunnittelu ja testaus

---

Tekijä: Tommi Tauriainen
Opinnäytetyön nimi: Laajennettava mittausjärjestelmä pulssiantureille
Työn ohjaaja: Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Kevät 2011 Sivumäärä: 57 + 2 liitesivua

---

Tämä opinnäytetyö on tehty omasta aloitteesta ja tavoitteena oli kehittää mittausjärjestelmä, joka käyttää pulssiantureita ja on helposti laajennettavissa sekä siirrettävissä erilaisille laite- ja ohjelmistoalustoille. Kiinnostus mittausjärjestelmän kehittämiseen syntyi, koska sille oli tarve ja kaupalliset ratkaisut todettiin joko käyttöön sopimattomiksi tai liian kalliiksi.

Työn laitteiston- ja ohjelmiston kehitys jaettiin pienempiin osa-alueisiin, jotta työn aikataulu ja dokumentointi saataisiin selkeäksi.

Työssä kehitettiin oma mittausjärjestelmä neljälle pulssianturille ja ominaisuudet tulevaisuuden laajentamisvaralle otettiin huomioon suunnittelussa. Työ sisältää mittauskortin elektroniikan suunnittelun, kehityksen ja testauksen sekä mittauskortin mikrokontrollerien ja tietokoneen ohjelmiston suunnittelun, kehityksen ja testauksen.

Mittausjärjestelmästä saatiin toimiva, ja työn tulokset tarjoavat huomattavan määrän apua seuraavan version kehittämiseen. Nykyinen mittausjärjestelmä ei ole lopullinen, vaan se edustaa prototyyppiä ja tutkimusalustaa jatkokehitystä varten.

Suunniteltaessa seuraavaa mittausjärjestelmää käytössä on tietotaitoa, kuinka suorituskyky ja toiminta saadaan mahdollisimman hyväksi tässä työssä tehtyjen havaintojen ja ideoiden pohjalta. Työn tulokset ovat myös sovellettavissa hyvin monenlaisiin ratkaisuihin, joissa tarvitaan ulkoisen sensoridatan siirtoa tietokoneelle.

Asiasanat:

Pulssianturit, mittausjärjestelmät, mikrokontrollerit, quadrature-koodaus, I²C, USB, Qt

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology and Telecommunications, Option of Electronics Design and Testing

Author: Tommi Tauriainen
Thesis title: Expandable Measurement System for Rotary Encoders
Supervisor: Kari Jyrkkä
Date: Spring 2011 Number of pages: 57 + 2 appendixes

Starting point of this bachelor's thesis was initiative and the object was to develop a measurement system which uses incremental-type rotary encoders, is easily extensible and portable to different kind of hardware and software platforms. The interest of developing own measurement system started from need to this kind of system. Ready commercial products did not meet the requirements of the system or they were too expensive for the work.

Thesis work has been subdivided to smaller parts concerning both hardware and software development to get clear timetable and documentation.

A new measurement system was developed in this thesis work for four rotary encoders and the option of extension was part of the design plan. Thesis work includes design, development and testing for the hardware of measurement system and to the software of microcontrollers and PC-software.

As a result of this bachelor's thesis a working measurement system was created. The information and know how gained from the work helps the developer when developing a new version of the system. This measurement system is not the final solution but rather a prototype and research device for further development work.

When developing the next system we now have much more information and knowledge how to get the performance and operating of the system as good as possible with the selection of components and the way how to make the software. The results of this work are also applicable to different kind of solutions where sensors are needed to connect to a computer to get some other kind of data.

Keywords:

Rotary encoders, measurement systems, microcontrollers, quadrature, I²C, USB, Qt

ALKULAUSE

Työn aihe syntyi alun perin oikeasta käytännön tarpeesta. Käytössä olleita vanhempia puun- ja metallintyöstökoneita haluttaisiin modernisoida sähköisellä mittausjärjestelmällä, mutta saatavilla olevat valmiit laitteistot ovat kalliita ja oman järjestelmän toteutus vaatisi aikaa, tietotaitoa ja sopivia työvälineitä. Opinnäytetyö sopi mainiosti mittausjärjestelmän toteutuksen tutkimiseen, koska tällöin olisi käytössä aikaa ja opinnot olisivat jo siinä vaiheessa että tietotaitoakin olisi riittävästi työn toteuttamiseen.

Opinnäytetyö järjestelmän kehityksestä vaikutti mielenkiintoiselta, koska sen toteutus vaatii sekä elektroniikan suunnittelua että ohjelmistonkehitystä. Työtä ei haluttu tehdä kuitenkaan mahdollisimman helpoksi vaan järjestelmästä pyrittiin kehittämään monipuolinen ja jatkokehitysmahdollisuudet huomioitiin työn aikana.

Alkuperäinen idea mittausjärjestelmän kehittämisestä pulssiantureille tuli isältä, joten kiitokset sinnepäin haastavasta ja mielenkiintoisesta opinnäytetyön ideasta. Työn ohjaavana opettajana toimi Kari Jyrkkä, jolle haluan esittää lämpimät kiitokset erittäin laadukkaasta työn ohjauksesta ja hyvistä ideoista niin toteutuksen kuin opinnäytetyön sisällön kannalta. Tuula Hopeavuorelle haluan esittää myös kiitokset työn kieliasun tarkistuksesta.

Oulussa 27.4.2011

Tommi Tauriainen

SISÄLTÖ

LYHENTEET

ASIC	Application-Specific Integrated Circuit, sovelluskohtainen mikropiiri
C++	Ohjelmointikieli
CAN	Controller Area Network, automaatioväylä
CW	Clockwise, myötäpäivään
CCW	Counterclockwise, vastapäivään
FPGA	Field-Programmable Gate Array, ohjelmoitava piiri
GCC	GNU Compiler Collection, GNU-kääntäjien kokoelma
HID	Human Interface Device, USB:n laiteluokka
I/O	Input/output, tulo/lähtö
I^2C	Inter-Integrated Circuit = TWI, tiedonsiirtoväylä
IC	Integrated Circuit, mikropiiri
kbit/s	Kilobittiä sekunnissa, tiedonsiirtonopeus
Mbit/s	Megabittiä sekunnissa, tiedonsiirtonopeus
PDIP	Plastic Dual In-line Package, IC-piirin kotelotyyppi
Qt	Alustariippumaton ohjelmistojen ja käyttöliittymien kehitysympäristö
RPM	Revolutions per minute, kierrosta minuutissa
SCL	Serial Clock, I^2C-väylän kellosignaali

SDA	Serial Data Line, I^2C-väylän datalinja
SOIC	Small-Outline Integrated Circuit, IC-piirin kotelotyyppi
SPI	Serial Peripheral Interface Bus, tiedonsiirtoväylä
TSSOP	Thin-Shrink Small Outline Package, IC-piirin kotelotyyppi
TWI	Two Wire Interface = I^2C, tiedonsiirtoväylä
USB	Universal Serial Bus, tiedonsiirtoväylä
XOR	Exclusive Or, “ehdoton tai”-operaatio digitaalitekniikassa

1 JOHDANTO

Tarkkaa mittaustekniikkaa tarvitaan monenlaisissa erilaisissa käyttöympäristöissä ja työkohteissa. Mittauskohteena voi olla esimerkiksi puun pituus, jyrsinkoneen tekemän uran leveys tai robotin etenemä matka. Mittausjärjestelmä voi olla esimerkiksi osa tehtaan linjaa määrittämässä työstökohteen paikkaa ja pituutta, jonka perusteella sahaus ja lajittelu voidaan tehdä sekä loppukädessä tarkistaa valmiiden töiden määrä tietokonejärjestelmästä. Mittaustulosten tulisi olla sekä luotettavia että tarkkoja ja helposti saatavilla jatkokäsittelyä varten.

Yksi monista mittaustavoista on käyttää pulssiantureita, jotka nimensä mukaisesti antavat tietoa pulsseina. Antureita on saatavana eri tekniikoilla toteutettuina niin edullisia kuin kalliimpia tarkkuuden ja teknisen laadun mukaan. Tällaisesta anturista ei yleensä saada suoraan mittaustuloksia ilman sille rakennettua laskentajärjestelmää, joka laskee pulssit sekä päättelee anturin akselin pyörimissuunnan.

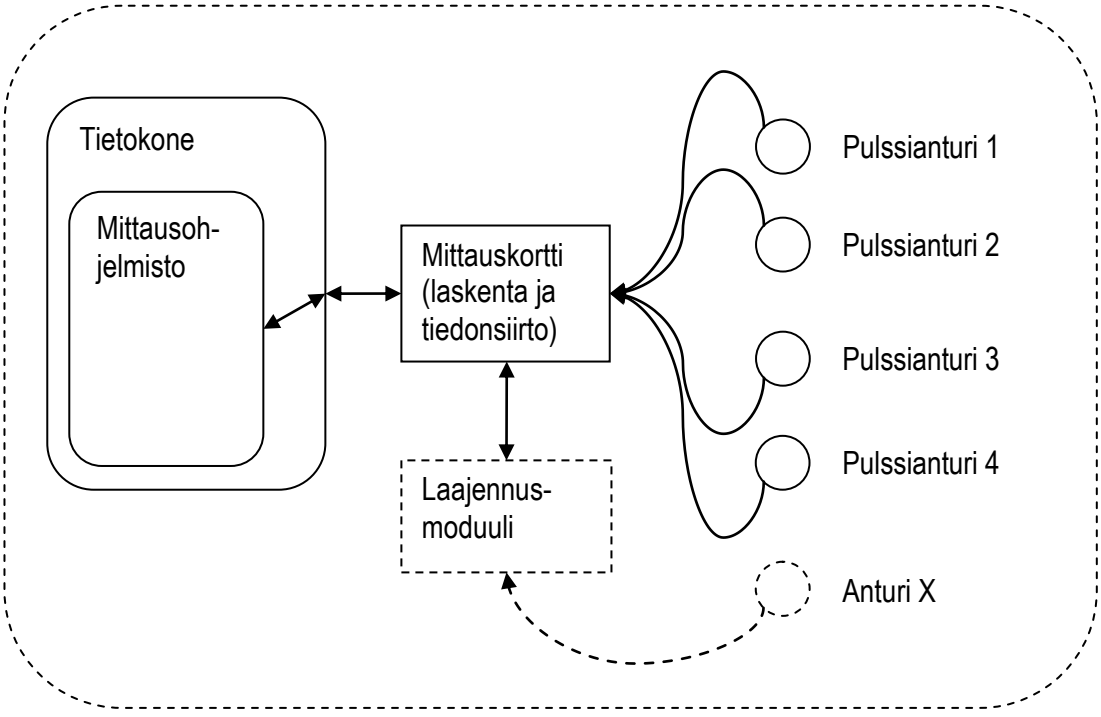
Tässä opinnäytetyössä tutkitaan yleiskäyttöisen mittausjärjestelmän toteuttamista, joka voidaan integroida mihin tahansa käyttökohteeseen. Suunniteltavan laitteiston vaatimuksiksi asetettiin, että se on edullinen ja tukee useaa yhtäaikaista anturia sekä kykenee laskemaan korkeintaan 100 kHz:n pulssiaaltoa syöttävien pulssiantureiden tulokset. Mittaustieto välitetään USB-väylän kautta tietokoneelle, jolla suoritetaan mittaustulosten jatkokäsittely.

Työssä käydään läpi mittausjärjestelmän laitteiston suunnittelu ja toteutus sekä ohjelmiston toteutus tietokoneelle ja mikrokontrollereille. Tietokoneelle luodaan mittaustulosten graafinen käyttöliittymä sekä ohjausmahdollisuus ja mikrokontrollereilla toteutetaan niin laskenta, kuin kommunikointi. Työssä selvitetään myös mikrokontrollerien soveltuvuutta pulssiantureiden laskureiksi ja verrataan työn tulosta valmiisiin kaupallisiin ratkaisuihin.

2 MITTAUSJÄRJESTELMÄ

Mittausjärjestelmän vaatimuksiksi asetettiin, että sen tulisi tukea neljää samaan aikaan toimivaa pulssianturia sekä kyetä määrittämään pulssien määrä ja suunta maksimissaan 100 kHz:n pulssitaajuutta tuottavilta pulssiantureilta. Pulssianturit valittiin käytettäväksi järjestelmässä, koska niillä ei ole rajoitettua laskenta-aluetta ja laskenta voidaan nollata tarvittaessa. Pulssiantureita on myös saatavilla todella hyvällä erottelukyvyllä, jonka ansiosta mittauksesta saadaan tarkka.

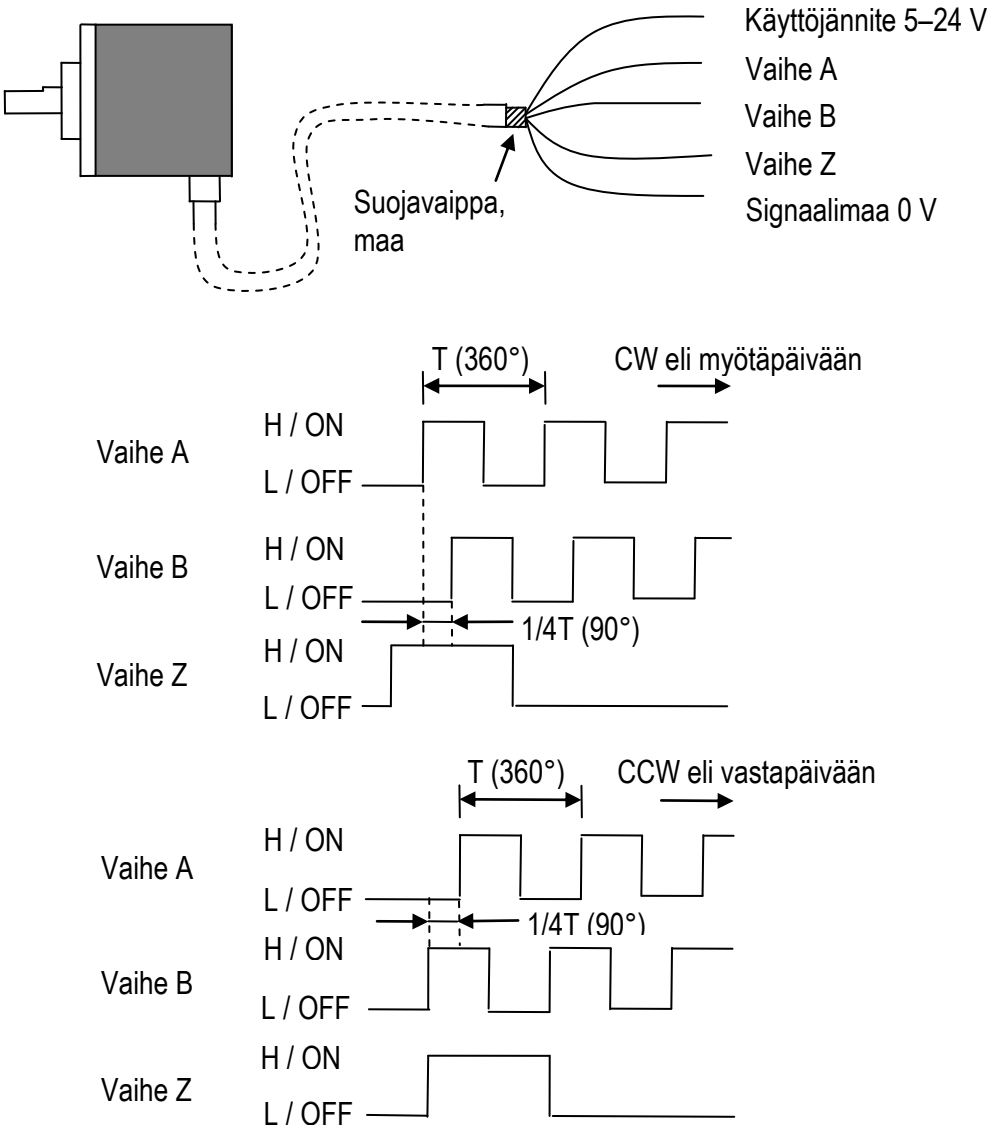
Järjestelmän tulisi olla helposti laajennettavissa esimerkiksi lisämoduuleilla tulevissa versioissa. Mittaustulokset välitettäisiin tietokoneelle USB-väylän kautta. Tietokoneella oleva ohjelma suorittaisi tarvittavat muunnokset haluttuun mittayksikköön tunnetun kalibrointikertoimen avulla, joka saataisiin käyttäjän suorittaman kalibroinnin avulla. Mittausohjelmistoon haluttiin myös mahdollisuus nollata laskuripiirien tulokset käsin. Suunniteltu mittausjärjestelmä on esitelty kuvassa 1.



KUVA 1. Suunniteltu mittausjärjestelmä

2.1 Pulssianturi

Pulssianturi antaa nimensä mukaisesti tietoa pulsseina pyörityssuunnan ja -nopeuden mukaan. Pulssianturia kutsutaan myös inkrementiaalianturiksi ja se antaa ulos yleensä kahta tai kolmea signaalia. Näistä A ja B ovat yleensä varsinainen pulssitieto, jossa pyörityssuunnan mukaan joko A- tai B-signaali on 90 astetta toista signaalia edellä. Tällaista signaalia kutsutaan quadrature-koodatuksi. Kolmas signaali Z on yleensä niin sanottu nollapulssi, jonka perusteella voidaan määrittää anturin nollakohta. (2 ; 3.)



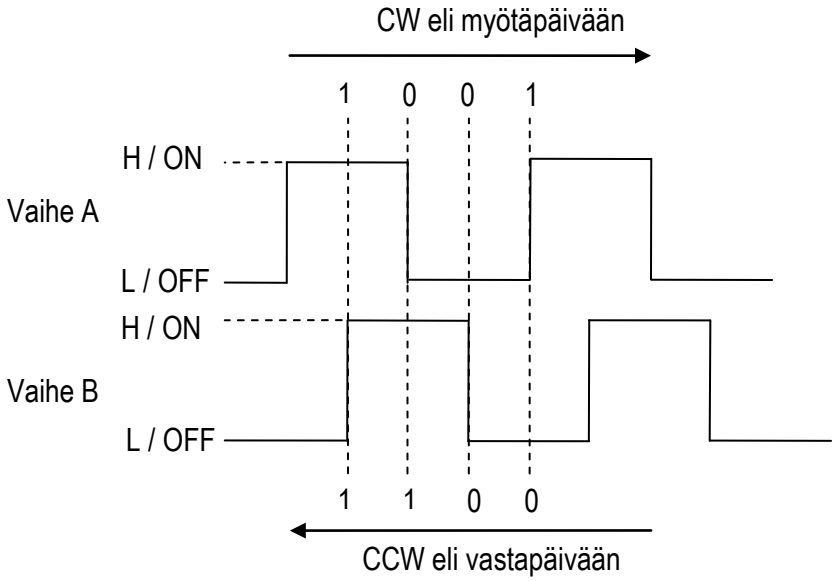
KUVA 2. Pulssianturi ja quadrature-koodaus (2 ; 3)

2.2 Quadrature-koodauksen tulkinta

Pulssianturilta saatava quadrature-koodaus voidaan tulkita erilaisilla logiikoilla ja ohjelmointitavoilla. Käytetty laskentalogiikka vaikuttaa lopulliseen pulssien määrään. Jos laskenta suoritetaan aina, kun jommankumman signaalin tila vaihtuu, saadaan anturilta laskettua nelinkertainen määrä pulsseja verrattuna siihen, mitä sille on ilmoitettu pulssimääräksi kierrosta kohti. Laskettaessa pulssit esimerkiksi A- tai B-signaalin nousevalta reunalta, saadaan yhdeltä kierrokselta valmistajan anturille ilmoittama pulssimäärä. (4.)

Tässä työssä tarkastellaan kahta esimerkkiä kuinka laskenta voidaan suorittaa. Molemmat laskentamallit ovat toteutettavissa digitaalielektroniikan peruskomponenteilla tai ohjelmistolla. Jos halutaan saada nelinkertainen pulssimäärä ja tämän seurauksena parempi erottelukyky, voidaan käyttää esimerkiksi XOR-operaatiota apuna, jolloin jokainen A- ja B-signaalin tilanvaihto huomioidaan.

Anturin akselin pyörintäsuunta voidaan siis määrittää, kun tiedetään anturin A-signaalin edellinenja B-signaalin nykyinen tila. Näiden kesken suoritetaan XOR-operaatio. Kuvasta 3 nähdään, kuinka signaalien tilat vaihtuvat anturin akselin pyörityssuunnan mukaan. (4.)



KUVA 3. Signaaleiden tilamuutokset (4)

XOR eli *exclusive or* on digitaalitekniikassa yleisesti käytetty bittitason operaatio, jonka totuustaulu on esitetty taulukossa 1. (5.)

TAULUKKO 1. XOR-operaation totuustaulu

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

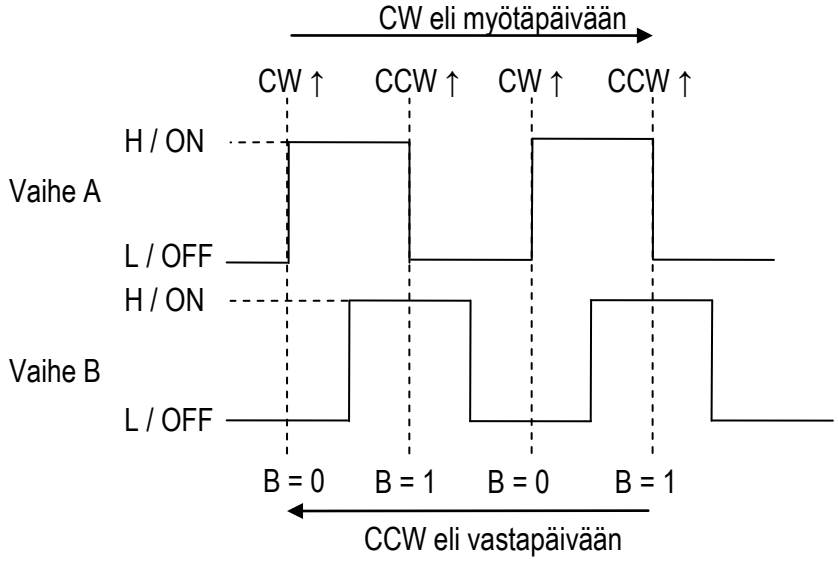
XOR-operaatio suoritetaan A signaalin edellisen tilan ja B signaalin nykyisen tilan kesken. Taulukon signaalien edelliset ja nykyiset tilat on määritetty kuvan 3 perusteella. (4.)

TAULUKKO 2. Pyörityssuunnan määrittäminen XOR-operaation avulla (4.)

$A_{Edellinen}$	$B_{Edellinen}$	$A_{Nykyinen}$	$B_{Nykyinen}$	$A_{Edel.}$ XOR $B_{Nyk.}$	Suunta
1	1	0	1	0	Myötäpäivään
0	1	0	0	0	Myötäpäivään
0	0	1	0	0	Myötäpäivään
1	0	1	1	0	Myötäpäivään
1	1	1	0	1	Vastapäivään
1	0	0	0	1	Vastapäivään
0	0	0	1	1	Vastapäivään
1	1	1	0	1	Vastapäivään

Jos laskentalogiikka ei ole riittävän nopea tai anturin nimellispulssimäärä ja siitä määräytyvä tarkkuus on riittävä, voidaan käyttää yksinkertaisempaa laskentaa. Laskenta tapahtuu ainoastaan A-signaalin nousevalta reunalta. B-signaalin tila määritetään jokaisella A-signaalin nousevalla reunalla ja B-signaalin tilan perusteella tiedetään, kumpaan suuntaan anturin akseli pyörii.

Kuvasta 4 näemme myötäpäivään pyöritettäessä saatavat laskentapulssit tekstillä *CW* ↑ ja vastapäivään pyöritettäessä *CCW* ↑. A-signaalin nousureunojen kohdalle on merkitty B-signaalin sen hetkinen tila.



KUVA 4. Signaaleiden tilamuutokset

Jos halutaan määrittää esimerkiksi anturin pyörimisnopeus, täytyy mukaan ottaa myös ajanotto. Tiedettäessä, montako pulssia anturin yksi kierros sisältää (*Pulssit*$_{max}$), ja laskettaessa tietyn ajan sisällä saatu pulssimäärä (*Pulssit*) yhteen suuntaan voidaan määrittää anturin akselin sen hetkinen kierrosnopeus. (4.)

$$\text{Kierrosnopeus} = \frac{\text{Pulssit}}{\text{Pulssit}_{\max}} / \text{Aika (s)} t\Delta \times \frac{60\ \text{s}}{1\ \text{min}}\ (\text{RPM}) \qquad \textit{KAAVA 1}$$

$$\text{esim. Kierrosnopeus} = \frac{723}{360} / 500 \times 10^{-3}\text{s} \times \frac{60\ \text{s}}{1\ \text{min}} = 241\ \text{kierrosta} / \text{min}\ (\text{RPM})$$

3 SYSTEEMIARKKITEHTUURIN VAIHTOEHTOJA

Mittausjärjestelmä voidaan toteuttaa useilla eri vaihtoehdoilla. Saatavana on täydellisiä kokonaisia mittausjärjestelmiä, jotka näyttävät kalibroinnin jälkeen suoraan mittaustuloksen omalta näytöltään. Järjestelmä voi sisältää anturit, laskentakortit sekä näyttölaitteen että ohjauslaitteiston.

Mittausjärjestelmä voidaan rakentaa myös itse yksittäisistä osista tai osakokonaisuuksista. Pulssiantureilta saatavat signaalit on tulkittava, jotta voidaan määrittää anturin akselin pyörimissuunta sekä pulssien määrä. Rakennettaessa omaa järjestelmää voidaan käyttää valmiita laskentakortteja osana mittausjärjestelmää tai luoda kokonaan uusi laskentaelektroniikka yksittäisiä komponentteja käyttämällä.

3.1 Valmiita kaupallisia laskentakortteja USB-väylään

Kaupallisten USB-väylään saatavien vähintään 4-kanavaisten laskurien hinnat lähtevät noin 60–70 eurosta ja päätyvät useampiin tuhansiin euroihin. Tässä työssä tutkitaan kahta esimerkkiä kaupallisista ratkaisuista laskentajärjestelmän toteuttamiseen ja listaamme niiden muutamia tärkeimpiä ominaisuuksia.

USB-QUAD08 on Measurement Computingin valmistama 8-kanavainen laskuri USB-väylään, joka kykenee laskemaan 10 MHz:n signaalia pulssiantureilta ja sisältää lisäksi 8 digitaalista I/O-linjaa. USB-QUAD08 edustaa selkeästi ammattikäyttöön tarkoitettua laskentalaitetta ja soveltunee teknisten arvojensa perusteella lähes mihin tahansa käyttökohteeseen. Laskennan lukuavaruus on valittavissa 16-, 32- ja 48-bittisenä. Ohjelmistonkehitys voidaan tehdä niin .NET-, C++- kuin Visual Basic -kielillä. Laitteen hinta on noin 430 euroa ja sen malli nähdään kuvasta 5. (6.)

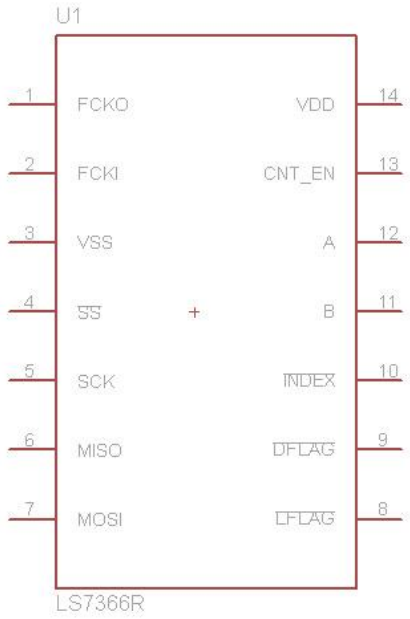
KUVA 5. Measurement Computing - USB-QUAD08 (6.)

1047 - PhidgetEncoder HighSpeed 4-Input on nelikanavainen laskurikortti USB-väylään, joka kykenee laskemaan 250 kHz:n signaalia pulssiantureilta ja sisältää myös 4 digitaalista sisääntuloa. PhidgetEncoder on hintaluokaltaan ja ominaisuuksiltaan enemmän harrastuskäyttöön sopiva, mutta soveltunee tästä huolimatta useisiin käyttökohteisiin. Laskenta on 16-bittinen ja ohjelmistonkehitys voidaan tehdä muun muassa C#-, C++-, Visual Basic- ja Java-kielillä. Kortti maksaa noin 70 euroa ja sen malli nähdään kuvasta 6. (7.)

KUVA 6. 1047 - PhidgetEncoder HighSpeed 4-Input (7.)

3.2 Kaupalliset komponentit laskennan toteuttamiseen

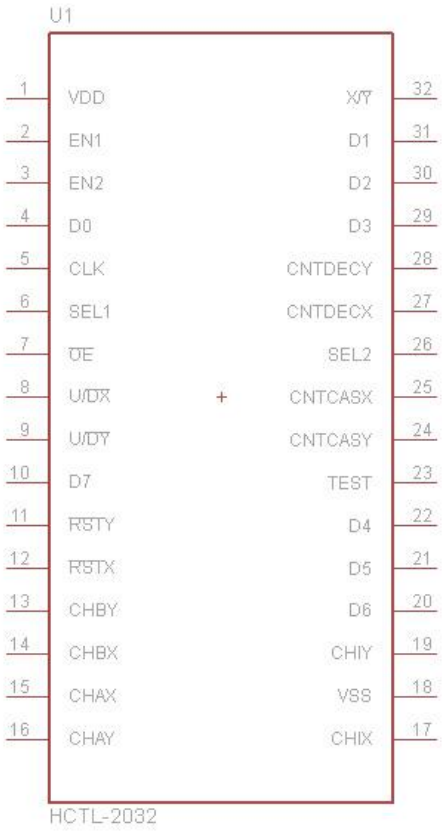
LSI/CSI:n valmistama LS7366R on 32-bittinen laskuripiiri yhdelle pulssianturille. Piiri siirtää laskentatietonsa sarjamuotoisena SPI-väylän kautta esimerkiksi mikrokontrollerille. LS7366R toimii maksimissaan 40 MHz:n kellotaajuudella ja se kykenee laskemaan 9,6 MHz:n taajuista pulssianturin signaalia. Pulssien laskenta voidaan valita 1-, 2- tai 4-kertaisella erottelukyvyllä, joka tarkoittaa että samasta pulssianturista saadaan yhdeltä kierrokselta joko nimellismäärä pulsseja tai vaihtoehtoisesti 2- tai 4-kertainen määrä. Piirissä on myös sisäänrakennettu häiriönsuodatus pulssiantureiden signaaleille. (8.)



KUVA 7. LS7366R-piiri (8.)

Piirin hyvä suorituskyky, sarjamuotoinen siirtomuoto ja monipuoliset ominaisuudet puoltavat valintaa käyttöön, mutta huonona puolena on piirin heikko saatavuus Suomeen. Tilatessa yhdysvaltalaiselta jälleenmyyjältä muutamia kappaleita yksittäisen piirin hinta nousee kohtuuttomaksi suurien toimituskulujen takia ja toimitusaika voi olla hyvinkin pitkä, koska tuote tulee ulkomailta. Piiri maksaa noin 4,00 euroa kappale ja se on saatavilla niin PDIP-, SOIC- kuin TSSOP-koteloinnilla. (8.)

Avago Technologiesin valmistama HCTL-2032 on 32-bittinen laskuripiiri kahdelle pulssianturille ja se siirtää laskentatietonsa rinnakkaismuotoisena 8 bitin annoksina. Piiri toimii maksimissaan 33 MHz:n taajuudella, mutta sille ei ole ilmoitettu maksimia pulssianturin signaalia, jota se kykenee lukemaan. Piiri sisältää digitaalisen häiriönsuodatuksen pulssiantureiden tulosignaaleille. Pulssien laskenta voidaan valita 1-, 2- tai 4-kertaisella erottelukyvyllä. (9.)



KUVA 8. HCTL-2032-piiri (9.)

Valintaa käyttöön puoltaa hyvä suodatus ulkoisille häiriöille ja hyvä suorituskyky. Huonoina puolina ovat piirin suurehko koko ja tiedonsiirtoon vaadittavat 16 eri signaalia kahta anturia kohti sekä piirin korkeahko hinta, n. 10,00 euroa. Kotelointivaihtoehtoina ovat PDIP ja SOIC. Piirin saatavuus on myös heikko ja sen valmistus on lopetettu.

3.3 Laskuri FPGA-piirillä toteutettuna

Ohjelmoitavan FPGA-piirin avulla saataisiin toteutettua hyvällä suorituskyvyllä varustettu laskentalogiikka, koska esimerkiksi pulssianturin quadrature-koodauksen tulkinta voidaan suorittaa yksinkertaisilla loogisilla operaatioilla. Valintaa puoltaa hyvä suorituskyky, mutta sitä vastaan ovat muun muassa piirien suurehko koko ja hinta.

Pienimmät ja edullisimmatkin FPGA-piirit sisältävät suuren määrän liitäntäpinnejä, joten voitaisiin sanoa, että ne ovat jopa liian monimutkaisia tämän mittausjärjestelmän toteuttamiseen. Tosin tehtäessä monimutkaisempia ja suurempia mittausjärjestelmiä FPGA-piirit voisivat olla yksi parhaista vaihtoehdoista laskentatekniikan toteutukseen, koska niillä voidaan toteuttaa hyvin suorituskykyisiä ratkaisuja. Kehitysympäristöt mahdollistavat esimerkiksi prosessorin käyttöönoton samalla piirillä, jolle on kuvattu esimerkiksi digitaalitekniikan porttitasolla tehdyt suodatus- ja laskentakytkennät.

3.4 Laskuri ASIC-piirillä toteutettuna

ASIC-piirillä saataisiin paras mahdollinen suorituskyky ja juuri halutut kytkennät sekä laskentarakenteet, mutta piirien valmistamien on erittäin kallista. Jos mittauskortteja tehtäisiin todella suuri erä, ASIC-piiri nousisi todennäköisesti parhaimpien vaihtoehtojen joukkoon. Vain muutamia mittauskortteja rakennettaessa piirien hinta nousisi todella korkeaksi ja ASIC-piirien valmistaminen yleensäkin on hyvin pitkäaikainen prosessi. ASIC-piiri soveltuu myös erittäin huonosti protolaitteiden kehittämiseen, koska valmista piiriä ei pystytä korjaamaan jälkeenpäin, vaan se on valmistettava kokonaan uudestaan.

3.5 Laskuri mikrokontrollerilla toteutettuna

Ohjelmoitavan mikrokontrollerin avulla prototyyppien tekeminen on nopeaa ja yleensä edullistakin. Edullisimmatkin mikrokontrollerimallit ovat sekä suorituskykyisiä että pienikokoisia ja niihin on sisäänrakennettu useita hyödyllisiä toimintoja valmiiksi, kuten esimerkiksi TWI (Two Wire Interface) eli paremmin I^2C-väylänä tunnettu sarjamuotoinen väylä piirien väliseen kommunikointiin. Nykyiset mikrokontrollerit ovat saatavilla myös USB- ja CAN-väylään yhteensopivina, joten niiden käyttöönotto kytkennässä helpottuu huomattavasti.

Mikrokontrollerilla toteutettaessa on kuitenkin huomioitava, että tiheää pulssiaaltoa antava pulssianturi vaatii nopean laskentalogiikan, joten piiriä valittaessa tulee kiinnittää huomiota erityisesti myös sen suorituskykyyn. Jos mikrokontrollerin kellotaajuus on liian alhainen, se ei ehdi suorittaa laskentaa riittävän nopeasti ja pian anturilta saatu laskentatieto on käyttökelvoton virheiden takia.

3.6 Käytettävän tekniikan valinta

Työssä päädyttiin toteuttamaan niin USB-kommunikointi kuin laskentalogiikkakin mikrokontrollereilla. Edullisimmat valmiit mittauskortit eivät täyttäneet järjestelmälle asetettuja laajennettavuuden vaatimuksia ja monipuolisemmat useampikanavaiset 24- tai 32-bittiset laskurikortit ovat yleensä huomattavan kalliita. Keskihintaisia tai kalliimpia malleja ei ole yleensä suunniteltu laajennettavaksi, vaan lisäkanavia tarvittaessa on hankittava toinen tai kerralla useammalla kanavalla varustettu malli.

Edullisimmat laskurikortit eivät yleensä laske kuin 16-bittisellä lukuavaruudella eli lukualue on -32768 – +32767, joka tarkoittaa, että anturi, jonka erottelukyky on 2000 pulssia kierrokselle, saa laskurin pyörähtämään ympäri jo 16 kierroksen jälkeen.

Lähes kaikki laskentakortit vaativat myös laitekohtaisten ajureiden tai valmistajan omien kirjastojen käyttöä ohjelmistonkehityksessä, mikä saattaa rajoittaa kortin käyttämistä erilaisilla laitealustoilla ja käyttöjärjestelmillä. Työssä haluttiin kehittää mittauskortti, jonka ohjelmisto ja laitteisto ovat mahdollisimman vähän alustariippuvaisia.

Laskentalogiikka päädyttiin toteuttamaan myös mikrokontrollereilla, koska kaupallisten komponenttien saatavuus oli heikohko ja samalla päästiin tutkimaan, kuinka hyvin edulliset mikrokontrollerit sopivat laskentalogiikan toteutukseen.

4 MITTAUSJÄRJESTELMÄSSÄ KÄYTETYT VÄYLÄTEKNIIKAT

4.1 USB-väylä

USB eli Universal Serial Bus on sarjamuotoinen väylä oheislaitteiden liittämiseksi tietokoneeseen. Sen kehityksen aloittivat vuonna 1994 Compaq, DEC, IBM, Intel, Microsoft, NEC ja Nortel. USB-väylän ensimmäinen versio 1.0 julkaistiin vuonna 1995.

USB tukee Plug and Play -ominaisuutta, joka mahdollistaa laitteiden automaattisen asennuksen sekä Hot-Plug -teknologiaa, jonka ansiosta laite voidaan kytkeä tietokoneeseen ilman virtojen katkaisua. USB-porttiin voidaan liittää 127 laitetta samaan aikaan ja sen suurin nopeus uusimmalla 3.0-versiolla on 4,8 Gbit/s. (10.)

Väylästä on julkaistu kolme versiota. Ensimmäisen 1.0-version tiedonsiirron maksiminopeus on 12 Mbit/s. Vuonna 2000 julkaistiin USB 2.0, jonka maksimi nopeus on 480 Mbit/s ja se on taaksepäin yhteensopiva alkuperäisen 1.0-version kanssa. Vuonna 2008 julkaistu USB 3.0 nosti nopeuden 4,8 gigabittiin sekunnissa ja se on taaksepäin yhteensopiva.

Väylälle on luotu valmiita laiteluokkia esimerkiksi ääni-, kuva-, video- ja osoitelaitteille. Yksi luokista on HID eli Human Interface Device class, joka käsittää esimerkiksi näppäimistön ja peliohjaimien kaltaiset ohjainlaitteet. HID-laite voi tästä huolimatta muun tyyppinen, kuten esimerkiksi äänimikserin tai lämpötila-anturin kaltainen laite. HID-laite tarkoittaa siis yleensä laitetta, joka ottaa vastaan ihmisen tekemiä syötteitä ja antaa takaisin sopivia vasteita. Hiirtä käytettäessä ohjauksesta muodostuva data on syöte ja kursorin liikkuminen näytöllä vaste. (11.)

4.2 I^2C-väylä

I^2C eli Inter-Integrated Circuit on sarjamuotoinen väylä mikropiirien väliseen tiedonsiirtoon, joka käyttää tiedonsiirtoon vain kahta johdinta. Väylän kehitti Philips Semiconductors 80-luvun alkupuolella. Väylä voidaan toteuttaa sekä master-slave-, että master-master-tyyppisenä eli sitä voi hallita yksi tai useampi piiri.

Ensimmäisen version tiedonsiirtonopeus oli 100 kbit/s ja 1992 standardin nopeudeksi määriteltiin 400 kbit/s. 1998 I²C:n määrittely päivitettiin ja maksiminopeudeksi määritettiin 3,4 Mbit/s. Väylällä olevien piirien määrää ei ole rajoitettu, mutta rajoittavana tekijänä toimii väylän kapasitanssi, joka saa olla enintään 400 pF. (12.)

I²C-väylää tukeva slave-piiri eli ohjattava piiri voi olla esimerkiksi lämpötila-anturi, jonka lämpötila-lukema voidaan lukea I²C-väylän kautta. I²C-väylä tunnetaan myös nimillä TWI eli Two Wire Interface ja TWSI eli Two-Wire Serial Interface. Tässä työssä väylästä käytetään sekä I²C- että TWI-nimiä, mutta käytännössä ne ovat täysin sama asia.

5 MITTAUSLAITTEISTON KEHITTÄMINEN

5.1 Laitteiston määrittely

5.1.1 Pääkomponenttien valinta

Mikrokontrollerit päätettiin valita Atmelin AVR-sarjasta, koska valmistajan piireistä oli jo entuudestaan kokemusta sekä niille oli hankittuna tarvittava ohjelmointilaite. Ohjelmointiin käytettäisiin Atmelin AVRISP mkII -ohjelmointilaitetta.

Laskentapiirien valinnassa vaatimuksiksi asetettiin, että piiri on fyysisesti pienikokoinen, tukee I^2C-väylää ja on riittävän suorituskykyinen laskentalogiikan toteuttamiseen sekä mahdollisimman edullinen. Piirien välinen kommunikointi haluttiin I^2C-pohjaiseksi, koska mittauskortin seuraaviin versioihin olisi tällöin helpompi lisätä laajennuspaikka eli liitin, johon voitaisiin yhdistää lisämoduuleita.

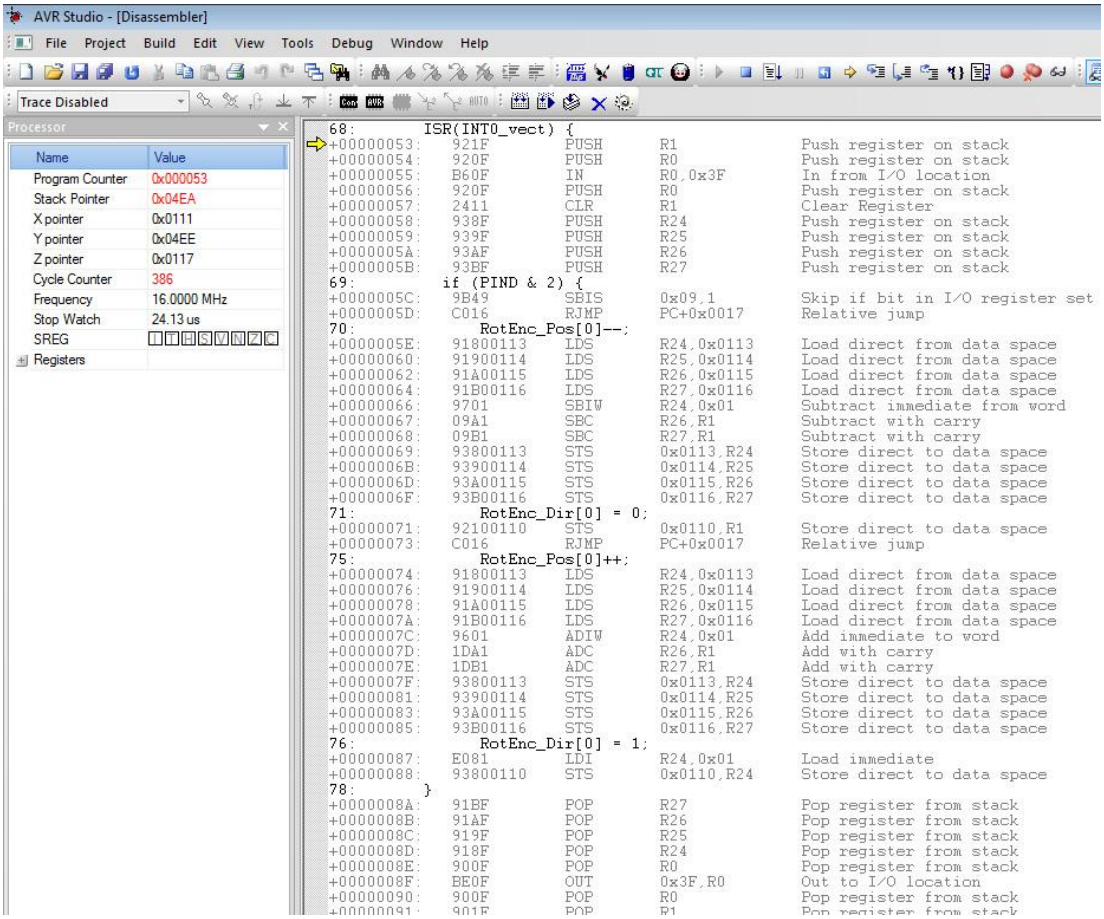
Rajaavaksi vaatimukseksi asetettiin myös, että yhteen piirin tulee saada liitettyä vähintään kaksi pulssianturia ja että piirilevylle juottaminen ei vaatisi puhallusjuottimen tai juotosuunin käyttöä. Tarvittava suorituskyky määritettiin laskennallisesti käytettävien antureiden maksimitaajuuden ja tarvittavan laskentaohjelman vaatiman ajan mukaan.

Tietokoneen ja mittauskortin väliseen kommunikointiin valittiin USB-väylä, joten siihen tarkoitukseen valittavaan piiriin tulisi olla sisäänrakennettuna USB:n vaatima tekniikka. Piirin tulisi olla myös fyysisesti pienikokoinen ja tukea I^2C-väylää sekä olla mahdollisimman edullinen. Piirin suorituskyky ei olisi niinkään kriittinen, mutta asennusta silmällä pitäen sen tulisi olla juotettavissa ilman erikoistyökaluja kuten laskentapiirikin. Näillä kriteereillä parhaaksi vaihtoehdoksi todettiin ATmega32U4-mikrokontrolleri Atmelin valikoimasta. (13.)

5.1.2 Vaadittavan suorituskyvyn määrittäminen

Piirin suorituskyky tietyn tehtävän suorittamiseen voidaan määrittää kellotaajuuden ja tehtävän suorittamiseen kuluvien prosessorikäskyjen määrän perusteella. Käskyjen määrä voidaan laskea suoraan käännetystä ohjelmasta käyttämällä niin sanottua disassembler-työkalua, joka näyttää

ohjelman suorituksen yksittäisten konekäskyjen tarkkuudella. Tehtävän suorittamiseen kulunut aika saadaan laskettua kertomalla kellotaajuus ja prosessorikäskyjen määrä keskenään.



KUVA 9. Keskeytysohjelman suoritusajan mittaus simulaattorin avulla

Laskennassa huomioitiin, että mikrokontrollerin prosessori joutuu myös tallentamaan muistiin senhetkisen suorituskohdan, jolloin keskeytys on tapahtunut ja palauttamaan prosessorin rekistereiden tilat takaisin keskeytysohjelman päätyttyä. Nämä vaiheet nähdään kuvasta 9 kommenteilla *Push register on stack* ja *Pop register from stack*. Mukaan sisältyvät myös rivien läheisyydessä olevat prosessorikäskyt.

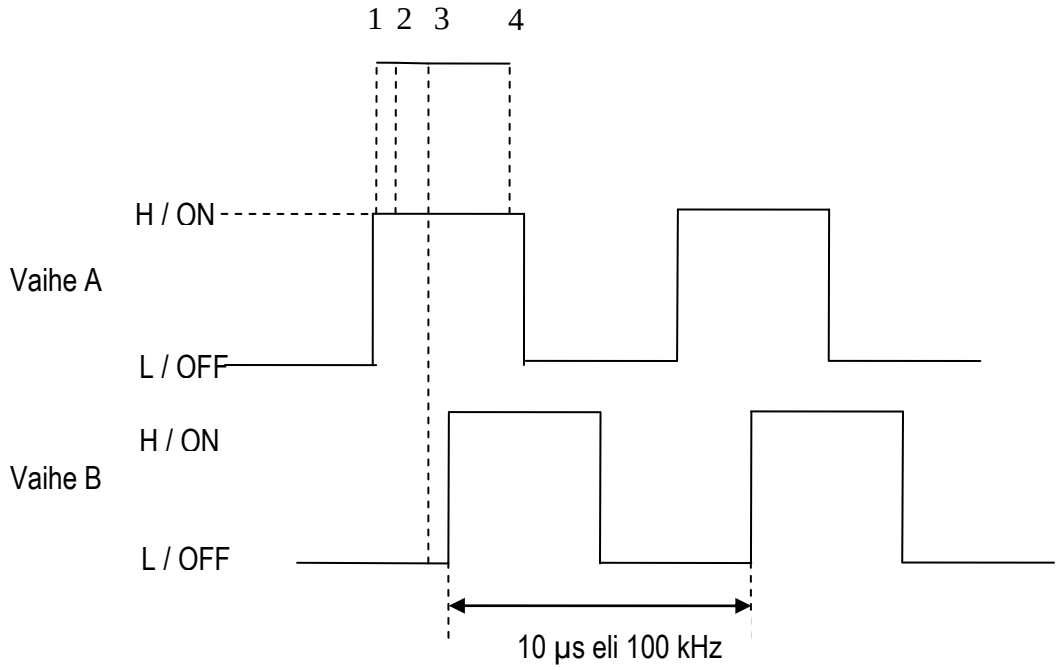
Piiri suorittaa lisäksi vähintään yhden prosessorikäskyn ennen kuin se siirtyy mahdollisen keskeytyksen aiheuttamaan keskeytysohjelmaan. ATmega168-piirillä sekä suoritustilanteen tallennus että palautus vaativat molemmat 35 prosessorikäskyä, jos käytetään alkuperäistä keskeytysohjelman perusrakennetta. Kääntäjä tarjoaa mahdollisuuden myös muokata vektorin ohjelmaa pienemmäksi, mutta tällöin voidaan kohdata ongelmia itse ohjelman suorituksessa. Ongelmien välttämiseksi keskeytysohjelman rakenne päädyttiin pitämään normaalina. (14.)

AVR Studion dissambler-työkalun ja simulaattorin avulla määritettiin, että pulssien laskenta sekä suunnan määritys vaatii 35 prosessorikäskyä eli yhteensä koko keskeytysohjelman suoritus vaatii 70 prosessorikäskyä. Kaavan 2 avulla voidaan määrittää, että 16 MHz kellotaajuudella keskeytysohjelman suoritus kestää tällöin 4,375 µs.

$$\text{Kokonaisaika} = \frac{1}{\text{Kellotaajuus}} \times \text{Käskyjen määrä} \qquad \text{KAAVA 2}$$

$$\text{eli Kokonaisaika} = \frac{1}{16*10^6\text{Hz}} \times 70 = 4{,}375 \times 10^{-6}\text{s}$$

Vaihe A:n nouseva reuna (0 prosessorikäskyä kulunut = 0 s)
1 = Prosessori reagoi keskeytykseen (7 prosessorikäskyä kulunut = 437,5 ns)
2 = Keskeytysohjelma alkaa (10 prosessorikäskyä kulunut = 625 ns)
3 = Keskeytysohjelma määrittää B signaalin tilan (27 käskyä kulunut = 1,69 µs)
4 = Keskeytysohjelma päättyy (70 käskyä kulunut = 4,375 µs)



KUVA 10. Keskeytysohjelman suoritus aikakaaviona

Näillä rajauksilla saatiin karsittua suurin osa valmistajan vaihtoehdoista pois ja laskentalogiikan toteuttamiseksi jäljelle jäivät ATmega48-, ATmega88- ja ATmega168-piirimallit. Piirit ovat fyysisesti samankokoisia ja samalla pinnijärjestyksellä, mutta ne eroavat toisistaan muistin ja keskeytysvektorin koon suhteen. Koska hintaero oli pieni, käytettäväksi piiriksi valittiin ATmega168.

5.1.3 Oheiskomponenttien valinta

Pulssiantureita on saatavilla erilaisilla sähköisillä toteutuksilla. Anturin signaalia muodostava lähtö on voitu toteuttaa esimerkiksi avolähtönä tai jännitelähtönä. Avolähtö toimii kuten kytkin, joka vaihtelee signaalia korkeaimpedanssisen tilan ja maatason välillä, jolloin se vaatii ylösvetovastuksen ylemmän tilan muodostamiseksi. Jännitelähdön signaali vaihtelee pulssianturin käyttöjännitteen ja maan välillä.

Mittauskortista haluttiin yhteensopiva molemmille anturityypeille, joten tuloliitännät varustettiin niin 10 kΩ:n ylösvetovastuksilla kuin jännitesovituksella 4049-piirin avulla. 4049 on niin sanottu HEX-buffer-piiri, jonka päätarkoituksena on toimia sekä puskurina signaaleille että jännitetasojen sovittajana. 4049-piirin tulot on myös suojattu sähköstaattisia purkauksia vastaan. Jännitelähtöisille pulssiantureille voidaan tuoda 4049-piirin ansiosta jopa 16 V.n käyttöjännite, koska 4049-piiri sovittaa tulosignaalin 5 V:n signaaliksi, joka soveltuu laskentapiirille tuotavaksi. Tällaisella toteutuksella voidaan käyttää myös antureita, jotka vaativat mittauskortilla saatavaa 5 V:n jännitettä suuremman käyttöjännitteen.

Oheiskomponentit määräytyivät pitkälti piirinvalmistajien suositusten mukaan. Kaikkien IC-piirien yksittäiset jännitetulot suodatettiin 100 nF:n ohituskondensaattoreilla ja kaikille piireille yhteiseksi jännitteen tasaajaksi valittiin 1000 µF:n kondensaattori. Iso kondensaattori suodattaa myös mahdollisia häiriöpiikkejä pois käyttöjännitteestä.

Pulssiantureiden käyttöjännitelinjalle valittiin suodatukseksi kaksi 1000 µF:n kondensaattoria 10 V:n jännitekestolla, jotka on kytketty sarjaan. Nämä kaksi kondensaattoria vastaavat yhtä 500 µF:n kondensaattoria 20 V:n jännitekestolla, jolloin myös kondensaattorit kestävät mahdollisesti korkeamman jännitetason. Mikäli kohteen käyttöjännite vaatii parempaa suodatusta, käytetään ulkoista suodatuskytkentää.

I^2C-linja vaatii ylösvetovastukset, joten sen SDA- ja SCL-linjoille lisättiin 4,7 kΩ:n ylösvetovastukset. Mikrokontrollerit vaativat myös 10 kΩ:n ylösvetovastuksen reset-linjalle, jotta piirit lähtevät suorittamaan niille luotua ohjelmaa sen sijaan että piiri ei lähtisi ollenkaan suorittamaan ohjelmaa tai nollautuisi satunnaisesti käytön aikana.

Kaikille mikrokontrollereille haluttiin yhteinen oskillaattori, jotta vältyttäisiin ylimääräisiltä komponenteilta. Kellosignaalin tuottajaksi valittiin 16 MHz:n oskillaattori DIL-14-koteloinnilla, koska ATmega32U4-piirin maksimi kellotaajuus on 16 MHz ja ATmega168-piireillä 20 MHz.

ATmega168-piirien ohjelmoimiseksi tarvitaan liittimet ohjelmointilaitteelle, joten molemmille lisättiin piikkirimaliittimet. Pulssiantureiden ja niiden käyttöjännitteen kytkemiseksi levylle lisättiin ruuviliittimet. Jokaiselle pulssianturille varattiin 5 ruuvipaikkaa, jotta jokaisen anturin signaaleille (A, B ja Z) ja käyttöjännitteelle (5–16V ja maa) olisi omat ruuvipaikat. Tällä pyrittiin välttämään tilanne, jossa saman ruuvin alle jouduttaisiin laittaan useita johtimia ja kiinnityksen kontakti voisi huonontua. Tämä mahdollistaa myös antureiden irrottamisen ja kiinnittämisen ilman muiden antureiden johtimiin koskemista.

5.2 Mittauslaitteiston suunnittelu

Mittauskortin piirikaavio- ja layout suunnittelu tehtiin Eagle 5.1.0 -ilmaisversiolla, joka sisältää tiettyjä rajoituksia. Piirilevyn maksimikoko on 100 x 80 mm, levy voi sisältää vain kaksi signaalikerrosta ja piirikaaviosuunnitteluun on käytössä vain yksi sivu. Rajoituksista ainoastaan piirilevyn maksimikoko vaikutti komponenttien sijoitteluun, mutta muuten rajoitusten ei todettu aiheuttavan ongelmia mittauskortin kehityksessä.

Mittauslaitteiston kehittäminen aloitettiin mittauskortin piirikaavion suunnittelulla. USB-liitintä ja ATmega32U4-piiriä lukuun ottamatta kaikille komponenteille oli tarjolla valmiit piirikaavio- ja layoutkuvat. USB-liittimelle löytyi valmis malli SparkFunin luomasta kirjastosta ja ATmega32U4:lle piirikaavio- sekä layout-kuva piirrettiin itse.

Piirikaavio suunniteltiin komponenttimäärittelyjen pohjalta. Piirikaavio jaettiin lohkoihin, jolloin tiettyä tehtävää suorittavat komponentit erottuisivat selvästi toisista komponenteista. Eri lohkoja yhdistävät johdotukset piirrettiin niin, että samat signaalit on nimetty sen sijaan että piirrettäisiin pitkiä johdotuksia, minkä seurauksena kuvan lukeminen vaikeutuisi. Laitteen piirikaavio on työn liitteessä 2.

Piirilevyn layout-suunnittelu aloitettiin sijoittelemalla USB-, ohjelmointi- ja ruuviliittimet kortin laidoille, jonka jälkeen loput komponentit pyrittiin sijoittelemaan niin että signaalivedoista saataisiin

mahdollisimman lyhyitä ja suoria. Komponentit pyrittiin myös sijoittelemaan pääosin levyn yläpuolelle.

Kiinnitysruuveille varattiin tyhjää tilaa kortin reunoilta, mutta reikiä ei tehty, koska sopivaa koteloa ei ollut vielä valittu. Kortin seuraavasta versiosta tehdään todennäköisesti hieman suurempi ja kotelostakin tulisi tällöin erikokoinen.

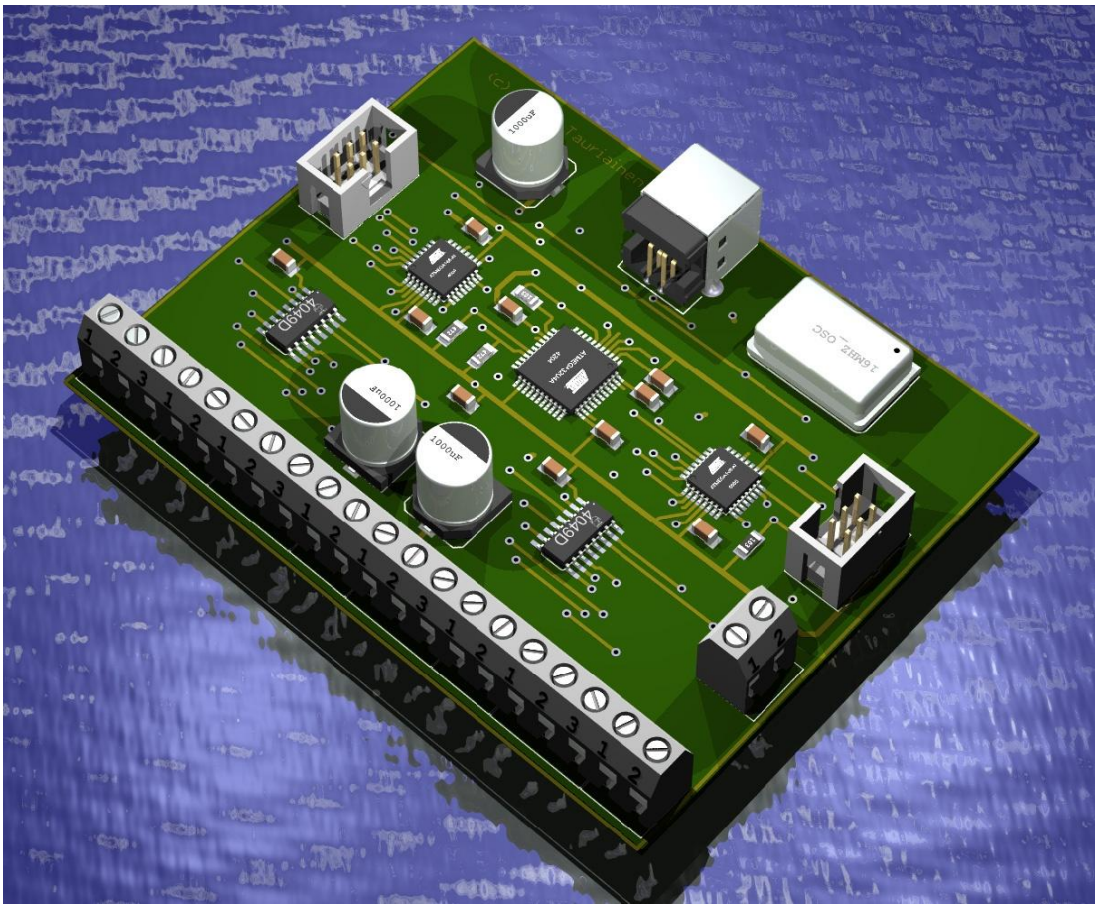
Signaalivedot pyrittiin tekemään levyn yläpuolella vaakasuunnassa ja alapuolella pystysuunnassa, jotta kaikki signaalivedot saataisiin tehtyä levylle. Suunnittelussa pyrittiin myös minimoimaan maalenkit sekä kelluvat maapisteet sijoittelemalla sopiviin kohtiin ylimääräisiä läpivientejä maasignaalille.

Layout-suunnittelua sekä komponenttisijoittelua tarkasteltiin myös 3D-mallinnuksen avulla, jolla nähdään miltä oikea fyysinen kortti tulisi näyttämään valmiina, riippuen siitä kuinka hyvin mallinnuksen komponentit vastaavat ulkonäöltään oikeita vastineitaan. 3D-malli luotiin ilmaisella 3D Eagle -kirjastolla, joka luo Eaglen layout-kuvan ja komponenttien pohjalta tiedoston, joka sisältää kuvauksen 3D-mallista. Tiedosto renderöitiin kuvaksi POV-Ray -ohjelmistolla.

3D-mallinnetusta kuvasta havaittiin mm. joidenkin läpivientien sekä komponenttien huonosti valitut paikat ja ne korjattiin ennen kuin piirilevy todettiin valmiiksi.

Määrittelyn mukaista laajennuspaikkaa ei kortin ensimmäiseen versioon otettu mukaan, koska opinnäytetyön aikataulu on kuitenkin rajallinen ja laajennuspaikan testaamiseksi olisi tarvittu vielä erillisen piirilevyn suunnittelu ja valmistus.

Laajennusmahdollisuutta ei kuitenkaan ole jätetty pois työstä, koska sen vaatima kommunikointi on täysin vastaavaa kuin levyllä olevien piirien välillä ja laajennusmahdollisuudet huomioidaan ohjelmiston puolella tulevaisuutta silmällä pitäen. Tuleviin versioihin tarvitaan lisäksi vain kolminapainen liitäntäpaikka osaksi I^2C-väylää, jossa siirtyvät väylän vaatimat SCL- ja SDA-signaalit sekä signaalimaa.



KUVA 11. 3D-mallinnettu mittauskortti

5.3 Mittauskortin toteutus

Mittauskortin piirilevy valmistettiin Oulun seudun ammattikorkeakoulun piirilevytuotannossa. Levyyn tilattiin lisäksi juotosestopinnoite, jotta erityisesti tiheäjalkaisten piirien juottamista saataisiin helpotettua ja signaalijohtimet saataisiin samalla suojattua paremmin esimerkiksi mahdollisilta tinaroiskeilta, jotka saattaisivat aiheuttaa oikosulkuja levyllä.

Komponentit juotettiin levylle korkeusjärjestyksessä ja sen mukaan minkä kokoisella juottimen kärjellä ne olisivat juotettavissa. Ensimmäiseksi juotettiin mikrokontrollerit pienellä juotoskärjellä fluksia eli tinan juoksutetta apuna käyttäen, jonka jälkeen keskikokoisella kärjellä vastukset ja kondensaattorit ja lopuksi isot kondensaattorit ja liittimet sekä oskillaattori.

5.4 Laitteiston testaus

Ennen käyttöönottoa kortti testattiin huolellisesti, jotta vältyttäisiin esimerkiksi tietokoneen USB-liitännän vioittumiselta. Mittauskortin testaus aloitettiin käyttöjännitelinjan resistanssin mittauksella, jossa selvisi, että käyttöjännite on jostakin oikosulussa maahan. Tutkimuksien jälkeen selvisi, että oskillaattorin jalkaan oli jäänyt pieni tinaroiske, joka johti käyttöjännitteen maahan. Tinaroiskeen poistamisen jälkeen muita oikosulkuja ei löytynyt.

Tämän jälkeen mitattiin vastusmittauksella, että kaikkien komponenttien signaalit saavat hyvän kontaktin, ja lopuksi mitattiin oskilloskoopilla, että oskillaattorilta saadaan 16 MHz:n kellosignaali kaikille mikrokontrollereille. Kuvan 12 mittauksessa yksi ruutu pystyakselilla vastaa 1 volttia ja vaaka-akselilla 0,2 mikrosekuntia. Käytössä ollut oskilloskooppi ei tarjonnut enää pienempää aikaskaalaa mittaukseen.

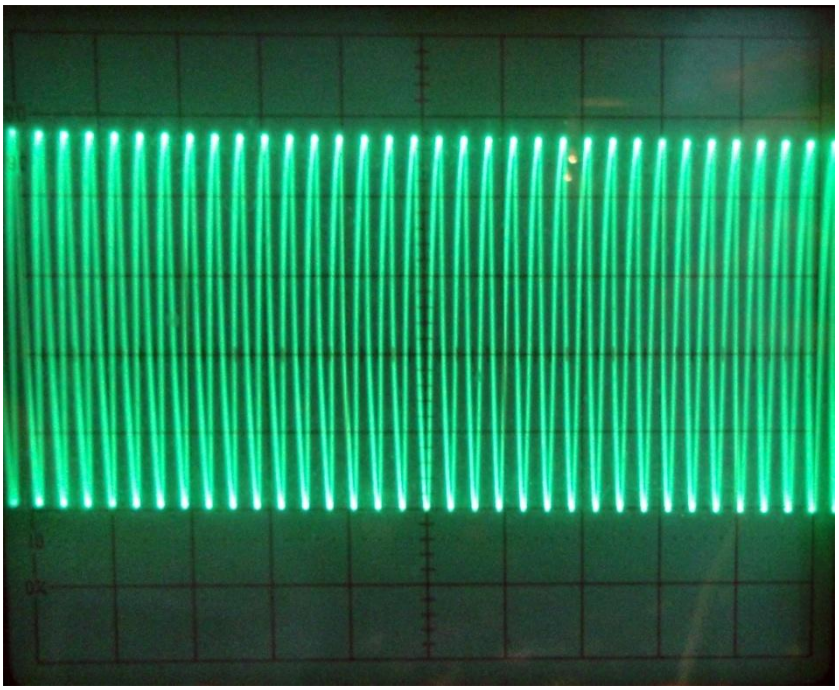
Kuvasta 12 nähdään, että signaali vaihtaa tilaansa 23 kertaa 7 ruudun aikana, josta voidaan laskea oskillaattorin taajuus:

Kokonaisaika 23 tilanvaihdolle on $7 \times 0{,}2 \times 10^{-6}s = 1{,}4 \times 10^{-6}s$

Yhden jakson aika on tällöin $\frac{1{,}4\times10^{-6}s}{23} = 60{,}869\ldots \times 10^{-9}s$

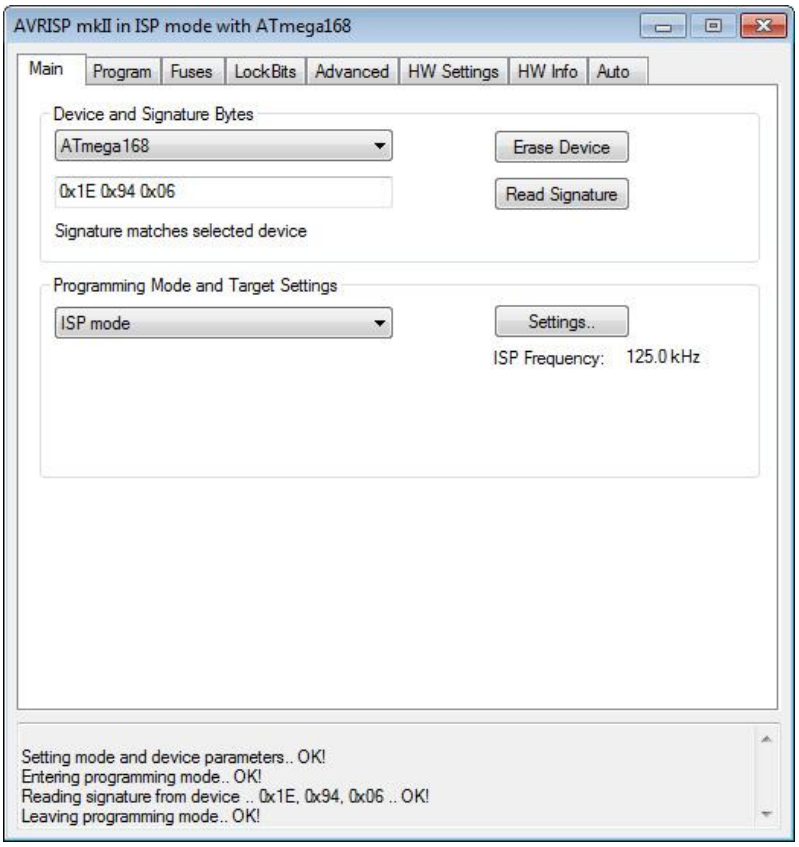
Taajuus on jaksonajan käänteisluku eli oskillaattorin taajuus on

$$\frac{1}{60{,}869\ldots \times 10^{-9}s} \approx 16{,}43\ \text{MHz}$$



KUVA 12. Oskillaattorin signaalin mittaus oskilloskoopilla

Seuraavaksi kortti kytkettiin tietokoneen USB-väylään ja kortin käyttöjännite mitattiin oskilloskoopilla, jotta mahdolliset häiriöt saataisiin näkyviin. Käyttöjännite todettiin puhtaaksi, joten lopuksi testattiin vielä AVRISP mkII -ohjelmointilaitteella, että mikrokontrollereihin saadaan yhteys. Ohjelmointilaitteella voidaan lukea piirin ns. signature eli allekirjoitus, jolla varmennetaan että piiri tunnistetaan oikein. Kuvasta 13 nähdään, että ATmega168-piirin allekirjoitus on *0x1E 0x94 0x06*.



KUVA 13. Piirin allekirjoituksen lukeminen AVRISP mkII -ohjelmointilaitteella

ATmega32U4-mikrokontrolleri oli alun perin tarkoitus ohjelmoida USB-liitännän kautta, mutta siihen tarkoitettu ohjelmisto todettiin jälkeenpäin riittämättömäksi, koska sillä ei pystytty muokkaamaan piirin ns. fuseja, joilla voidaan valita esimerkiksi käytettävä kellosignaali ja monia muita piirin asetuksia. Niinpä ohjelmoinnin ajaksi piirin jalkoihin juotettiin hyppylangat, jotka liitettiin 2x3-piikkirimaliittimeen, jonka kautta ohjelmointi tehtiin AVRISP mkII -ohjelmointilaitteella. Kortin seuraavaan versioon ohjelmointiliitin tultaisiin lisäämään myös USB-kommunikointia hoitavalle piirille.

Testauksien jälkeen kortti todettiin toimivaksi ja sille luotuja ohjelmia päästiin testaamaan käytännössä.

6 OHJELMISTON KEHITYS

6.1 Määrittely

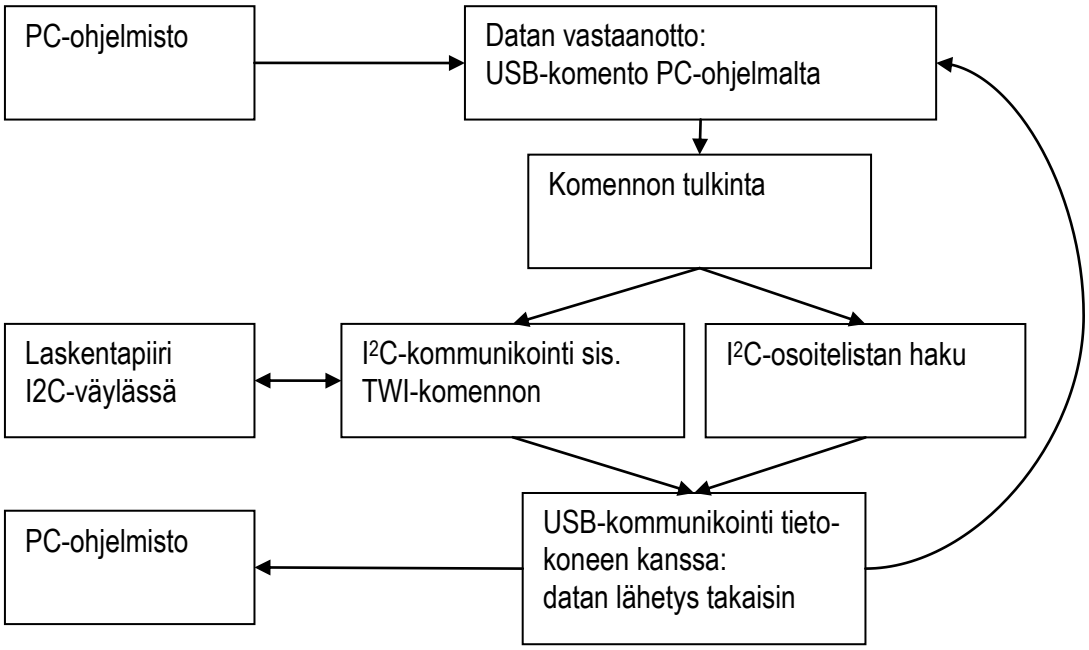
Mittausjärjestelmän vaatimuksiin kuului laajennettavuus, joten mikrokontrollerien ja tietokoneen ohjelmistoista pyrittiin kehittämään helposti laajennettavia. Tiettyjä tehtäviä hoitavat funktiot pyrittiin määrittelemään sellaisiksi, että ne on helppo muokata toimimaan erikokoisissa järjestelmissä ja eri anturimäärillä. Esimerkiksi funktio, jonka tehtävänä on nollata piirin kaikki laskurit, toimii sen mukaan, montako anturia ohjelman alkupään määrityksissä on ilmoitettu.

6.1.1 USB-mikrokontrolleri

USB-kommunikointia hoitavan mikrokontrollerin ohjelmiston määrittelyt luotiin mittausjärjestelmän yleisten vaatimusten pohjalta. Tietokoneen ja laskentapiirien välistä kommunikointia hoitavan piirin tulisi kyetä kommunikoimaan piirien kanssa I^2C-väylän kautta ja välittämään tiedot mittausohjelmistolle USB-väylän kautta. PC-ohjelmiston antamien käskyjen mukaan USB-mikrokontrolleri voisi lukea laskentapiirin antureiden määrän ja antureiden pulssimäärät sekä nollata laskurit tarvittaessa.

Mikrokontrollerin muistin määrä ei saisi toimia rajoittavana tekijänä, joten antureiden tuloksia ei piiriltä lukemisen jälkeen säilytettäisi piirin muistissa vaan väliaikainen data korvautuisi uudella mittaustiedolla uuden lukupyynnön myötä.

Mittauksen ohjaus ja järjestelmän käyttöönotto tapahtuisi kokonaan PC-ohjelmiston kautta, joten piirin tehtävänä olisi toimia mittaustietojen välittäjänä sekä PC-ohjelmiston pyynnöstä välittää tiedot laskentapiirien määrästä ja niiden I^2C-osoitteista. Nykyisessä versiossa I^2C-osoitteet ja piirien määrät on tallennettu piirin muistiin kiinteästi. Tulevaisuudessa määritykset voitaisiin muokata enemmän automaattiseksi, mutta aikarajan puutteissa tätä ei lähdetty tutkimaan.



KUVA 14. USB-mikrokontrollerin ohjelman rakenne

Piirien väliseen kommunikointiin I²C-väylällä määritettiin komennot, joiden avulla USB-piiri voisi lukea ja ohjata mittauspiirejä.

TWI_CMD_GET_COUNTER_COUNT (0x10)

TWI_CMD_READ_COUNTER_BYTE0 (0x20)

TWI_CMD_READ_COUNTER_BYTE1 (0x21)

TWI_CMD_READ_COUNTER_BYTE2 (0x22)

TWI_CMD_READ_COUNTER_BYTE3 (0x23)

TWI_CMD_READ_COUNTER_DIRECTION (0x24)

TWI_CMD_RESET_COUNTER (0x30)

TWI_CMD_RESET_COUNTERS (0x40)

6.1.2 Laskuripiirit

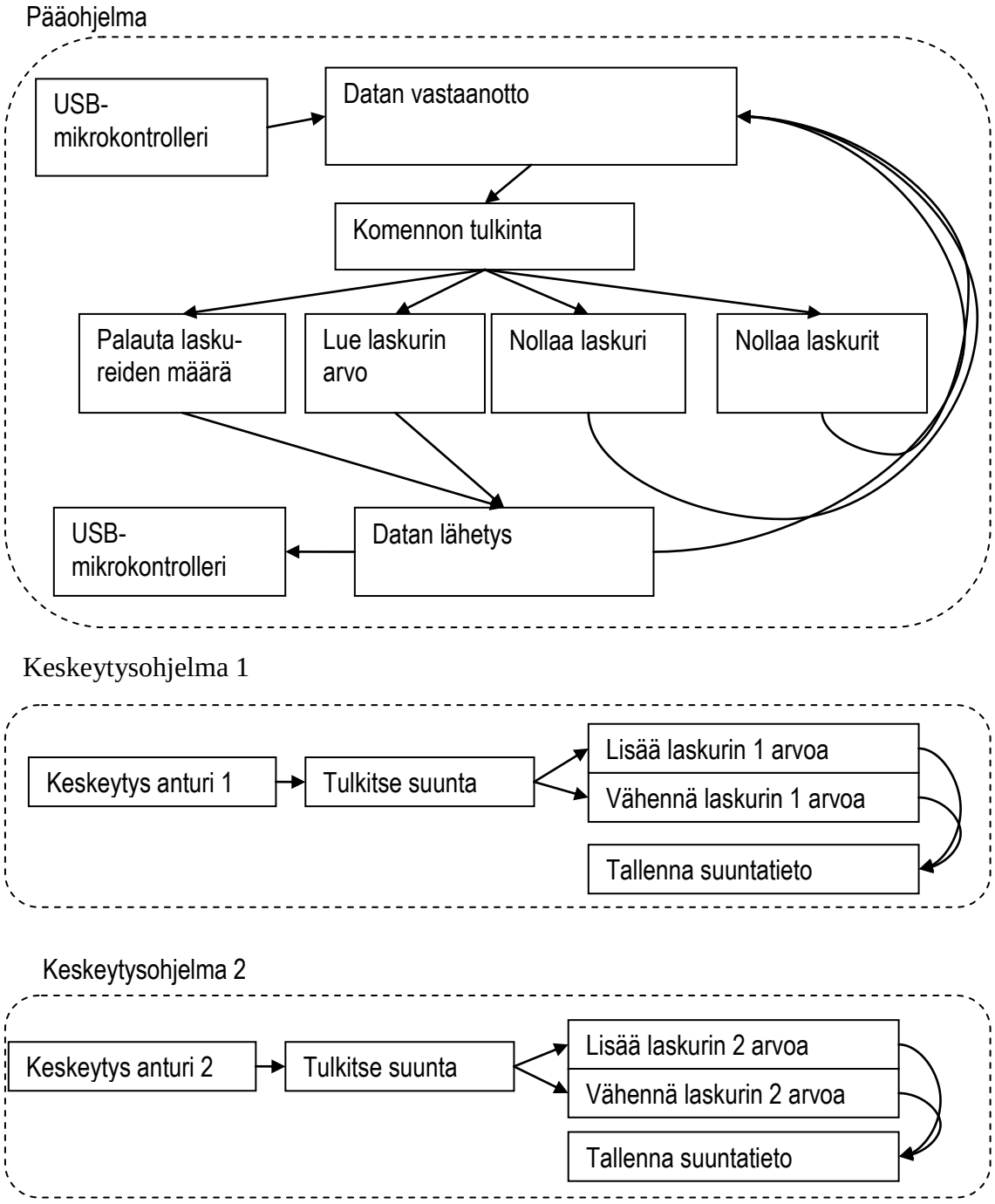
Laskuripiirien ohjelmiston vaatimukset luotiin yleisten vaatimusten pohjalta. Ohjelmiston tulisi kyetä tulkitsemaan antureilta saatava quadrature-koodaus ja määrittämään sen perusteella anturin akselin pyörimissuunnan sekä kokonaispulssimäärän. Piirin tulisi myös kyetä kommunikoimaan ohjaavan USB-mikrokontrollerin kanssa I^2C-väylän kautta.

Ohjelman tulisi siis sisältää myös anturikohtainen nollausmahdollisuus ja vastata oikealla tavalla USB-kommunikointia hoitavan mikrokontrollerin kyselyihin. Laskentapiireillä tulisi siis olla käytössä samat edellä mainitut TWI-komennot kuin USB-piirillä, joiden perusteella piirien toimintaa voitaisiin ohjata.

Laskenta määriteltiin toimivaksi anturin nimelliserotteluky vyllä eli käytössä olisi nousureunaherkkä liipaisu, jotta prosessorille aiheutuvien keskeytysten määrä saataisiin mahdollisimman pieneksi. Tällä pyrittiin välttämään tilannetta, jossa mikrokontrolleri ei ehdi käsitellä kaikkia keskeytyksiä riittävän nopeasti ja laskentaan tulee virhettä mukaan.

Laskentaan käytettäisiin piirin korkeimmilla prioriteeteilla toimivia keskeytyksiä, jotta laskenta toimisi mahdollisimman tehokkaasti ja esimerkiksi I^2C-väylän keskeytykset olisivat alemmalla prioriteetilla. Korkeamman prioriteetin keskeytys on voittavassa osassa, joten käytännössä toinen laskureista olisi prioriteetiltaan korkeampi kuin toinen.

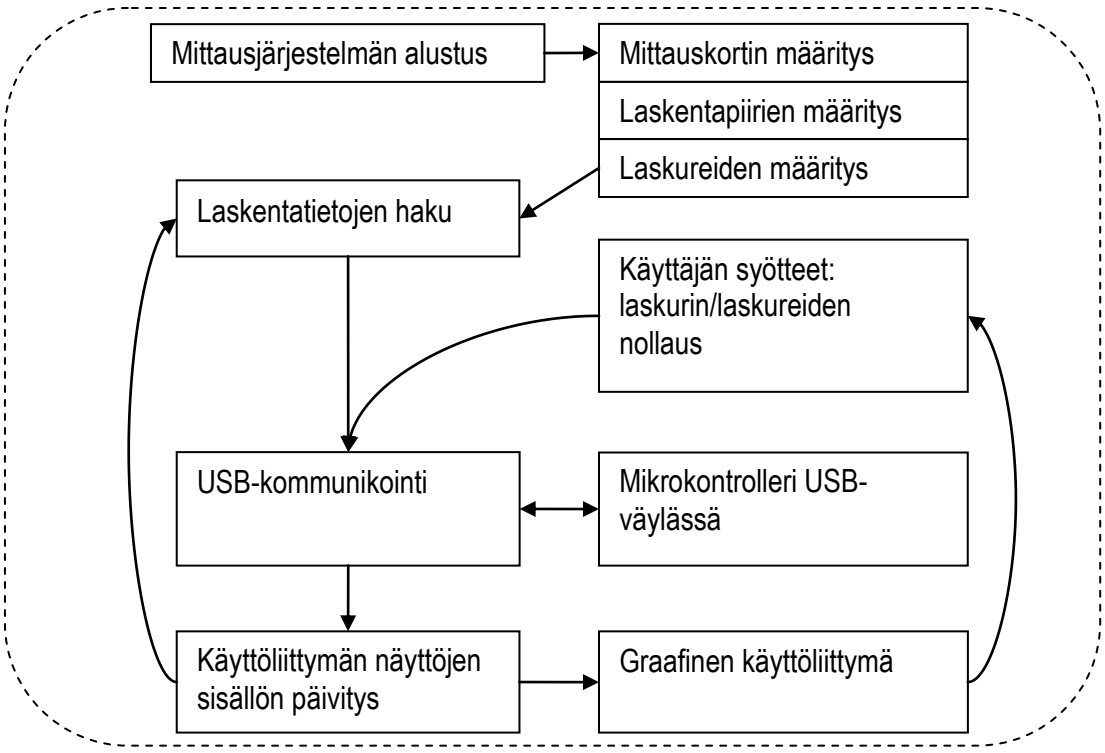
Laskuripiireille luodun ohjelman rakenne on esitelty kuvassa 15.



KUVA 15. Laskuripiirin ohjelma rakenne

6.1.3 Tietokoneen mittausohjelmisto

Tietokoneen mittaussovellus määriteltiin mittausjärjestelmän yleisten määritelmien pohjalta. Sovelluksen tulisi kommunikoida mittauspiirien kanssa USB-piirin välityksellä, mutta käyttäjän näkökulmasta tämän tulisi olla mahdollisimman huomaamatonta ja käyttäjälle näytettäisiin vain ja ainoastaan mittauksen kannalta oleellista tietoa.



KUVA 16. PC-ohjelmiston rakenne

Ohjelmiston tulisi myös hoitaa mittauskortin käyttöönotto ja kommunikointi mahdollisimman automaattisesti. Käyttöliittymän tulisi olla mahdollisimman yksinkertainen ja selkeä, jotta oikeassa käyttötilanteessa käyttäjän ei tarvitsisi huolehtia laitteiston sisäisestä toiminnasta, vaan hän voisi keskittyä puhtaasti omaan työsuoritukseensa.

Mittaussovelluksen määritelmiksi asetettiin myös, että sen kääntäminen muille alustoille olisi helppoa eli ohjelmointikielen ja käytettävien rajapintojen tulisi olla mahdollisimman vähän alustariippuvaisia, jotta samaa mittausohjelmistoa voitaisiin käyttää erilaista prosessoria ja käyttöjärjestelmää käyttävillä alustoilla.

Tietokoneen ja mittauspiirien välille määriteltiin kommunikointikomennot, joiden perusteella eri ohjelmistot tietäisivät mitä ja mistä dataa halutaan lukea.

Tietokoneen ja USB-piirin väliseen kommunikointiin määriteltiin komennot, joita vastaava datapaketti on suluissa.

USB_NO_COMMAND (0x00)

USB_CMD_GET_COUNTER_COUNT (0x10)

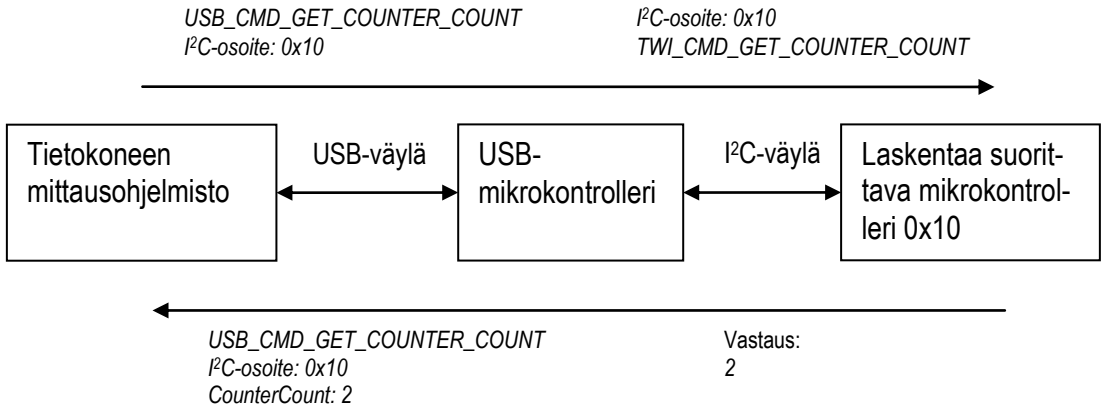
USB_CMD_READ_COUNTER (0x20)

USB_CMD_RESET_COUNTER (0x30)

USB_CMD_RESET_COUNTERS (0x40)

USB_CMD_GET_COUNTER_IC_ADDRESSLIST (0x50)

Komentojen avulla USB-piirille saataisiin välitettyä haluttu toiminto, jonka mukaan se kommunikoisi I²C-väylän kautta laskentapiirien kanssa. Komentojen avulla mittaussovellus pystyisi määrittämään käytettävän mittausjärjestelmän rakenteen. Kuvassa 17 on esitelty, kuinka esimerkiksi mittausohjelmisto voi tehdä kyselyn laskentapiirin laskureiden määrästä.



KUVA 17. Mittaussovellus määrittää laskentapiirin laskureiden määrän

6.2 Suunnittelu

6.2.1 USB-mikrokontrolleri

USB-kommunikointi suunniteltiin toimimaan niin, että mittauskortti näkyisi tietokoneelle HID-yhteensopivana laitteena, jolloin laitteelle ei tarvitsisi kirjoittaa erillistä laiteajuria.

USB-kommunikointia hoitavan mikrokontrollerin ohjelman suunniteltiin pohjautuvan kahteen Atmelin malliohjelmaan, joita täydennettäisiin niin, että ohjelmalle tehdyt määritelmät saataisiin toteutettua. USB/HID-kommunikoinnin toteuttamiseen käytettäisiin EVK527-ATmega32U4-usbdevice_hid-mallia, jolla ATmega32U4-piiri saadaan näkymään tietokoneelle HID-yhteensopivana laitteena.

I^2C-väylän kommunikoinnin toteuttamiseksi käytettäisiin AVR315: Using the TWI module as I^2C master -esimerkkiä. AVR315-mallikoodin avulla piiri saadaan toimimaan I^2C-väylän isäntänä. Mallikoodi on luotu IAR EWAAVR -kääntäjälle, joten sen toimivuus AVR GCC -kääntäjällä tulee varmistaa ennen lopullista käyttöönottoa. (15.)

Ohjelman päärakenne pohjautuisi ATmega32U4-usbdevice_hid-malliin, koska USB-kommunikointi on ohjelman eniten koodia sisältävä osa.

6.2.2 Laskuripiirit

Laskuripiirien suunnittelu tehtiin vaatimusten sekä suorituskyvyn ja yleisten määrittelyjen pohjalta. Laskuripiirien ohjelma suunniteltiin pohjautuvaksi Atmelin kehittämään I^2C-esimerkkiin AVR311: Using the TWI module as I^2C slave, johon laskentaa ja muita laskureiden ohjaukseen tarvittavia funktioita voitaisiin liittää mukaan.

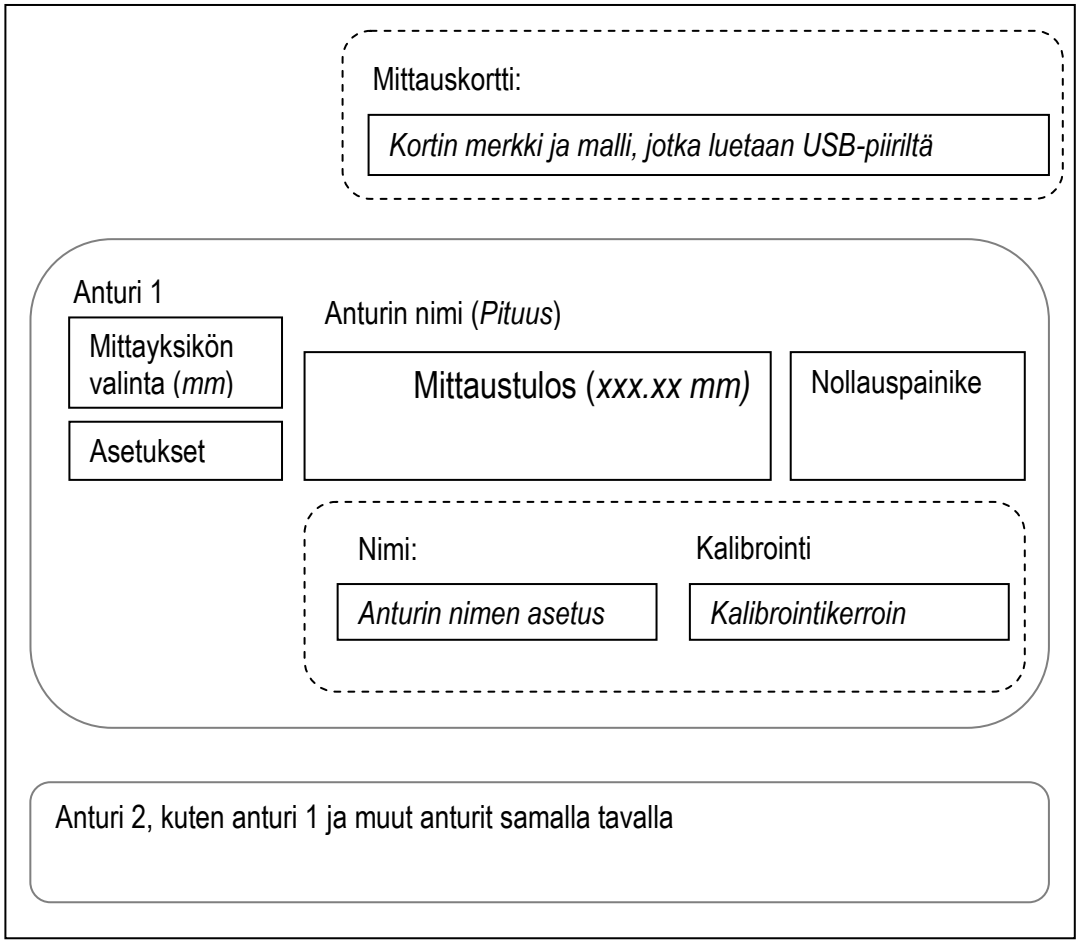
Kuten AVR315, myös AVR311-malli on myös luotu alun perin IAR EWAAVR -kääntäjälle, joten sen toimivuus AVR GCC -kääntäjällä tulisi varmistaa ennen käyttöönottoa. (16.) Ennen AVR311-mallin käyttöä keskeytysohjelman funktiot sekä alustukset testattaisiin puhtaassa projektissa simuloimalla, jonka jälkeen projektit yhdistettäisiin yhdeksi ohjelmistoksi.

6.2.3 Tietokoneen mittausohjelmisto

Mittausohjelmiston käyttöliittymä suunniteltiin luotujen määritelmien pohjalta. Käyttöliittymä pyrittiin kehittämään mahdollisimman yksinkertaiseksi ja selkeäksi. Painikkeista tehtäisiin riittävän suuret, jotta sovelluksen käyttö onnistuisi helposti myös esimerkiksi kosketusnäytön avulla.

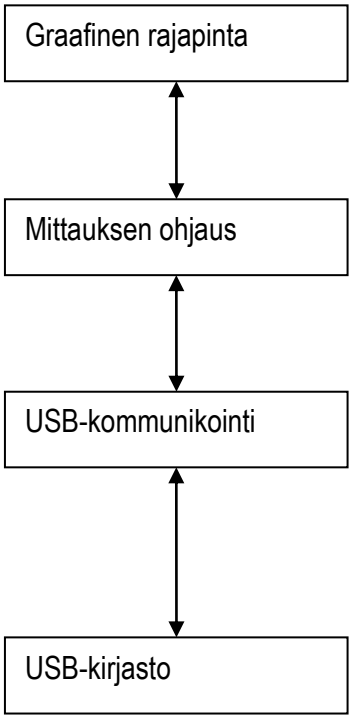
Sovelluksen käyttöliittymän suunniteltu perusrakenne on kuvassa 18. Katkoviivalla merkityt alueet piilotetaan käyttöliittymästä, mikäli niitä ei tarvitse näyttää käyttäjälle.

Esimerkiksi kortin merkki ja malli voidaan piilottaa, jos mittausjärjestelmä saadaan toimintaan, koska ne eivät sisällä mittauksen kannalta oleellista tietoa. Ne voidaan tuoda näkyviin, jos mittauskorttia ei havaita järjestelmässä, jolloin käyttäjä voi kokeilla yhdistämistä uudelleen esimerkiksi mahdollisen kommunikointihäiriön takia.



KUVA 18. Mittaussovelluksen käyttöliittymän suunnitelma

Mittausohjelmiston rakenne suunniteltiin lohkottavaksi pienempiin osa-alueisiin, jotta ohjelman koodin kehittäminen ja ylläpito selkeytyisi. Ohjelman päälohkot on esitelty kuvassa 19.



Graafinen rajapinta tarjoaa käyttäjälle antureiden näytöt ja ohjauspainikkeet, kalibrointivalinnat sekä asetukset.

Mittauksen ohjaus hoitaa mittausjärjestelmän määrittämisen ja käyttöönoton. Mittauksen ohjaus välittää laskentatiedot graafiselle käyttöliittymälle.

USB-kommunikointi luo yhteyden laitteen ja mittauksen ohjauksen välille. USB-kommunikointi suorittaa myös laitteiden hakemisen ja yhdistämisen pyyntöjen mukaan USB-kirjaston avulla.

USB-kirjasto toimii rajapintana ohjelmiston ja käyttöjärjestelmän USB-rajapinnan välisessä kommunikoinnissa.

KUVA 19. Ohjelman rakennelohkot

6.3 Toteutus

6.3.1 USB-mikrokontrolleri

USB-kommunikointia hoitavan mikrokontrollerin ohjelmiston toteutus aloitettiin ATmega32U4-usbdevice_hid -mallisovelluksen kääntämisellä. Malliohjelmaa muokattiin aluksi vain hieman, että nähtäisiin sen toimivan oikein käytettävällä piirillä. Kun mallisovellus todettiin toimivaksi, projektiin lisättiin mukaan AVR315 Using the TWI module as I^2C master -mallin lähdekoodi.

Sovelluksen käännöksen aikana havaittiin, että IAR EWAAVR -kääntäjälle tehty AVR315-malli ei ole suoraan yhteensopiva AVR GCC -kääntäjän kanssa, joten se vaati muutoksia toimiakseen. AVR GCC käyttää mm. erinimisiä kirjastotiedostoja ja keskeytysvektoreita. (17.)

IAR EWAAVR -kääntäjälle tarkoitetut rivit, joilla määritetään esimerkiksi TWI-keskeytysohjelma ovat muotoa

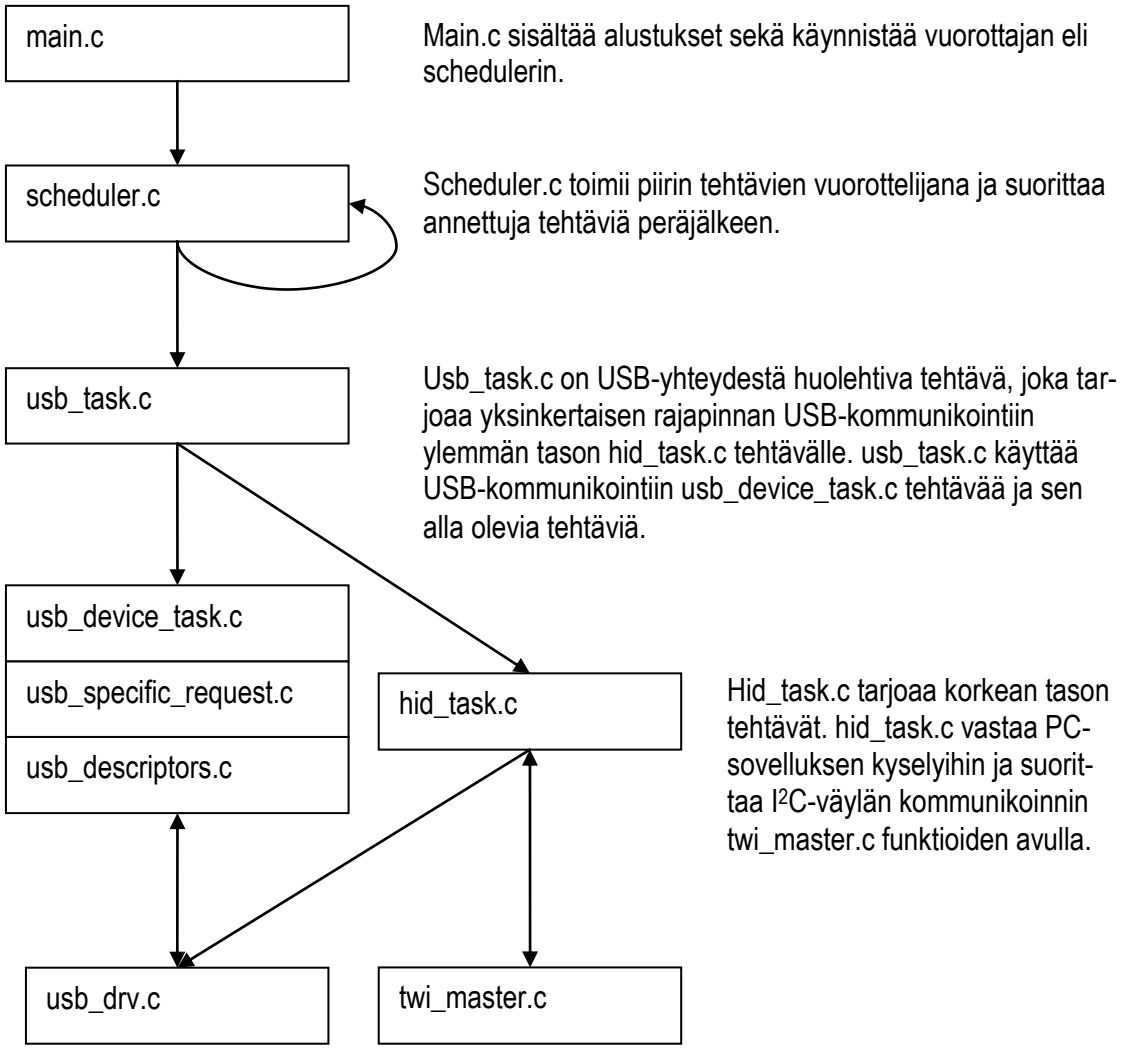
#pragma vector=TWI_vect

__interrupt void TWI_ISR(void);

AVR GCC -kääntäjälle vastaava merkintä on

ISR(TWI_vect)

Mikrokontrollerin ohjelmiston rakenne on selvitetty kuvassa 20.



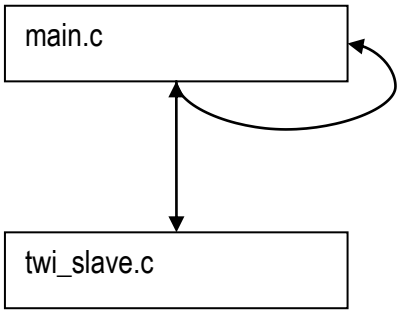
KUVA 20. USB-mikrokontrollerin ohjelman rakenne

6.3.2 Laskuripiirit

Laskuripiirien keskeytysfunktiot ja alustuskoodit luotiin aluksi puhtaaseen ohjelmapohjaan, jotta ne saatiin testattua kunnolla. Kun keskeytysfunktiot todettiin toimiviksi, ne liitettiin AVR311 : Using the TWI module as I2C slave -esimerkin projektiin mukaan.

AVR311-esimerkki on luotu IAR EWAAVR -kääntäjälle ja se ei ole suoraan yhteensopiva käyttämämme AVR Studion AVR GCC -kääntäjän kanssa.

AVR311 vaatii toimiakseen samat muutokset kuin AVR315-malliin tehtiin. (17.) Laskuripiirien ohjelma jaettiin kahteen osaan kuvan 21 mukaan.



Main.c sisältää laskentaan käytettävien keskeytysfunktioiden ja muuttujien alustukset, keskeytysfunktiot, anturien nollausfunktiot sekä I²C-viestien käsittelyt ja vastaukset.

Twi_slave.c sisältää I²C-väylän kommunikointiin vaadittavat alustukset sekä tiedonsiirtoon käytettävät alemman tason funktiot ja keskeytysfunktion, joka reagoi I²C-väylän muutoksiin.

KUVA 21. Laskuripiirin ohjelman rakenne

6.3.3 Tietokoneen mittausohjelmisto

Mittausohjelmiston toteutus tehtiin Qt Creator 2.0.1 -ohjelmistonkehitystyökalulla, jolla luodut sovellukset ovat Qt 4.7.0 -yhteensopivia. Qt valittiin ohjelmiston ohjelmointikieleksi, koska se on suunniteltu alustariippumattomaksi ja pohjautuu työn tekijälle ennestään tuttuun C++-kieleen.

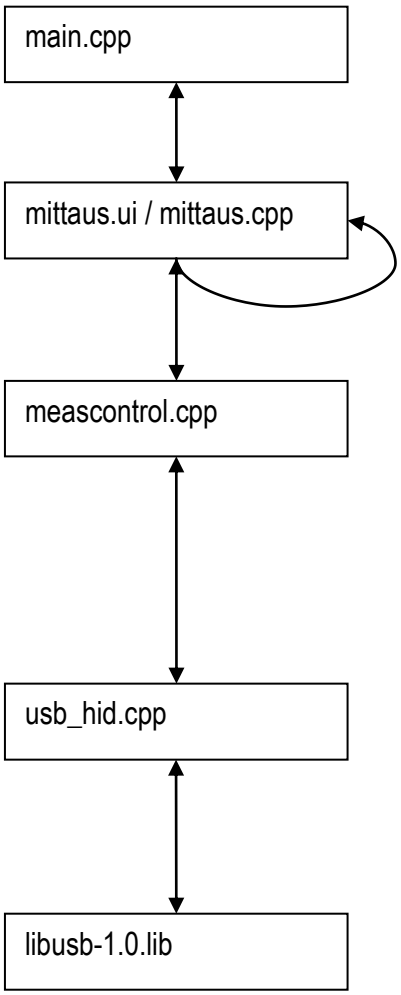
Ohjelmiston toteutus aloitettiin käyttöliittymän rakentamisella, joka toteutettiin Qt Creatorin Design -työkalun avulla. Design-työkalulla graafinen käyttöliittymä voidaan rakentaa ilman koodin kirjoittamista sijoittelemalla halutut elementit oikeille paikoilleen. (18, s. 23.) Sovelluksen varsinainen toiminnallisuus luodaan ohjelmoimalla. Käyttöliittymä kehitettiin suunnitelman mukaiseksi, mutta alussa testin aikana käytössä oli vain yhdelle laskurille näyttö sekä ohjauspainikkeet.

Kun käyttöliittymä saatiin valmiiksi, siirryttiin painikkeiden toiminnallisuuksien lisäämiseen. Painikkeille luotiin alustavat slotit eli aliohjelmat, joihin siirrytään painiketta painettaessa. Kalibrointi- ja nimikentälle luotiin slotit joihin siirrytään, jos kenttien sisällön arvo muuttuu.

Testauksen ajaksi käyttöliittymään lisättiin väliaikaisia muuttujia, joiden data vastasi todellisessa käyttötilanteessa käytettävää laskentadataa. Väliaikaisten muuttujien avulla varmistettiin, että lopullinen mittaustieto saadaan näkyville oikeassa muodossa näytöille mittayksiköt mukaan lukien. Kun käyttöliittymän perustoiminnot todettiin toimiviksi, aloitettiin USB-kommunikointia hoitavan luokan toteutus.

USB-kommunikoinnin toteuttamiseksi valittiin libusb-1.0-kirjasto, josta saatavilla on libusbWin-Backend -käännös, joka itsessään käyttää kommunikointiin käyttöjärjestelmän omaa usb-kirjastoa. Libusb:n avulla oman sovelluksen USB-rajapinnasta saadaan käyttöjärjestelmäriippumaton ja sen kääntäminen eri alustoille helpottuu huomattavasti, koska sovelluksen rakennetta ei välttämättä tarvitse muuttaa ollenkaan kohdealustaa vaihdettaessa. (19.)

Mittaussovellus jaettiin eri osa-alueisiin ohjelmiston suunnitelman mukaisesti. Ohjelmiston hierarkkinen rakenne noudattaa kuvan 22 mukaista mallia.



Main.cpp sisältää Qt:n vaatimat alustukset sekä käynnistää mittaus.uin ja mittaus.cpp:n sisältämän graafisen sovelluksen.

Mittaus.ui ja mittaus.cpp sisältää sovelluksen graafisen kuvauksen sekä ohjauksen vaatimat funktiot
Graafinen sovellus saa mittaustiedot meascontrol.cpp funktioiden avulla.

Meascontrol.cpp tarjoaa graafiselle käyttöliittymälle yksinkertaisen kommunikointirajapinnan laskureiden lukemiseksi ja ohjaamiseksi. Meascontrol käyttää USB-kommunikointiin usb_hid.cpp:n tarjoamia funktioita ja tulkitsee siltä saadut viestit. Meascontrol hoitaa myös mittausjärjestelmän määrittelyn sekä käyttöönoton.

Usb_hid.cpp luo meascontrolin viesteistä HID-viestejä sekä huolehtii viestien vastaanottamisesta sekä välittämisestä meascontrol.cpp:n funktioille
usb_hid etsii ja yhdistää laitteet meascontrollin pyyntöjen mukaan.

Usb_hid.cpp kommunikoi käyttöjärjestelmän USB-rajapinnan kanssa libusb-1.0-kirjaston avulla.

KUVA 22. Mittausohjelmiston rakenne

Eri luokat välittävät tietoa Qt:n ns. signaaleiden ja slottien avulla. Signaali on eräänlainen heräteviesti ja slot toimii kuin aliohjelma, joka reagoi heräteviestiin ja suorittaa sille tehdyn toiminnon. (18, s. 20-22.)

Kun esimerkiksi usb_hid luokka havaitsee uuden HID-viestin saapuvan USB-laitteelta se lähettää signaalin HIDMessageReport, joka sisältää viestin. Vastaavasti Meascontrol-luokka reagoi signaaliin ohjaten sen MessageParser-slottiin eli aliohjelmaan, joka käsittelee viestin sisällön. Jos MessageParser tulkitsee laskentatiedon muuttuneen, se lähettää uuden signaalin CounterChanged, joka vastaanotetaan Mittaus-luokassa, jonka UpdateCounterScreen slot päivittää kyseisen laskurin näytön.

6.4 Testaus

6.4.1 USB-mikrokontrolleri

USB-mikrokontrollerin toimintaa testattiin Atmelin AVR153 -esimerkin UsbHidSmallDemoCode-mallia apuna käyttäen. AVR153 sisältää esimerkkejä miten USB-HID pohjainen kommunikointisovellus toteutetaan tietokoneelle Atmelin HID-ajuria käyttäen. (20.)

Sovelluksen avulla seurattiin, että piirin ohjelmisto noudatti sille suunniteltua toimintaa. Piirille lähetettiin määritelmien mukainen USB-pyyntö ja vastauksen perusteella tulkittiin oliko piiri suorittanut oikean ohjelmahaaran. Testauksen avulla saatiin jäljitettyä ohjelmakoodin virheitä, jotka aiheuttivat piirin ohjelman jumittumisen ikuiseen silmukkaan.

USB-piirin testauksessa käytiin läpi myös kaikki viestipaketit ja tutkittiin, että esimerkiksi laskentadata saadaan välitettyä oikeassa muodossa. Testauksen ajaksi USB-piiriin tallennettiin kuviteltu laskentatieto, jonka siirtoa testattiin USB-piirin ja PC-ohjelmiston välillä. Viestipaketin sisältö luettiin ja tarkistettiin käsin, minkä perusteella voitiin todeta vastaako paketin sisältö odotettua dataa.

6.4.2 Laskuripiirit

Laskuripiirien pulssien laskentaa testattiin aluksi AVR Studiossa simulaattorin avulla. Simulaation avulla saatiin varmistettua, että piiri laskee myötäpäivään pyöritettäessä ylöspäin ja vastapäivään pyöritettäessä alaspäin. Simuloimalla testattiin myös, että 32-bittinen laskentatieto saadaan pilkottua 8 bitin annoksiksi, koska I^2C-väylällä ja USB-väylällä tietoannos koostuu 8 bitin tavuista, joten data on lohkottava osiin ennen lähetystä.

I^2C-väylän kommunikointia testattiin USB-mikrokontrolleria apuna käyttäen. Testaukseen käytettiin Atmelin AVR153-esimerkin UsbHidSmallDemoCode-mallia, joka on komentorivipohjainen sovellus USB-kommunikointiin AVR-piirien kanssa.

USB-piirille luotu sovellus välittää tietokoneelta I^2C-väylään lähetettävät paketit ja välittää I^2C-väylältä paluutietona saadun datan takaisin USB-väylään. Kommunikointiin käytettiin määritelmien mukaisia komentoja.

Testin aikana USB-piiriin ohjelman rakennetta muutettiin niin, että lähetettäessä tietoa takaisin tietokoneelle, datapaketin mukaan liitettiin myös I²C-väylän kommunikointia hoitavan funktion virhekoodit. Näin tietokoneelle saatiin välitettyä myös tiedot väylällä tapahtuvista mahdollisista virhetilanteista.

Piirien välisessä kommunikoinnissa esiintyi ongelmia. Virhekoodien perusteella todettiin, että väylä on siirron aikana väärässä tilassa ja lisäksi slave-piiri reagoi väylän muutoksiin väärin. Vianselvityksen jälkeen ilmeni, että AVR GCC -kääntäjä vaatii lisäalustuksia väylän käyttöönottoon toisin kuin IAR EWAAVR -kääntäjä. Kääntäjän loputkin yhteensopivuusongelmat saatiin ratkaistua, kun mallikoodista löydettiin AVR GCC -kääntäjälle muokattu malli, josta voitiin etsiä eroavaisuuksia käyttämämme koodin kanssa. (21.)

I²C-väylän kommunikointi saatiin toimimaan, mutta useamman tavun siirto vaatii, että yhteys avataan joka kerta kokonaan uudestaan. Tarkempien tutkimusten jälkeen selvisi, että käytetty TWI-kirjasto ei tue ainakaan AVR GCC -kääntäjällä toistuvia aloituksia vaan vaatii erilaista funktiota datan käsittelyyn. Testien perusteella haettiin mahdollisimman nopea yhteys I²C-väylälle ja 320 kHz:n todettiin toimivan luotettavasti. Maksimilla 400 kHz:n taajuudella yhteys onnistui useimmiten, mutta ajoittain siinä ilmeni virheitä.

Kun yhteys saatiin toimimaan luotettavasti, siirryttiin testaamaan piirien laskentalogiikkaa todellisessa käyttöympäristössä simulaation sijaan. Laskureiden laskentasuunnat tarkistettiin yksinkertaisesti kytkemällä tulosignaaleita vuorotellen nollaan, millä matkittiin oikean anturin toimintaa. Kun laskenta todettiin toimivaksi, aloitettiin laskennan testaaminen oikeiden antureiden kanssa.

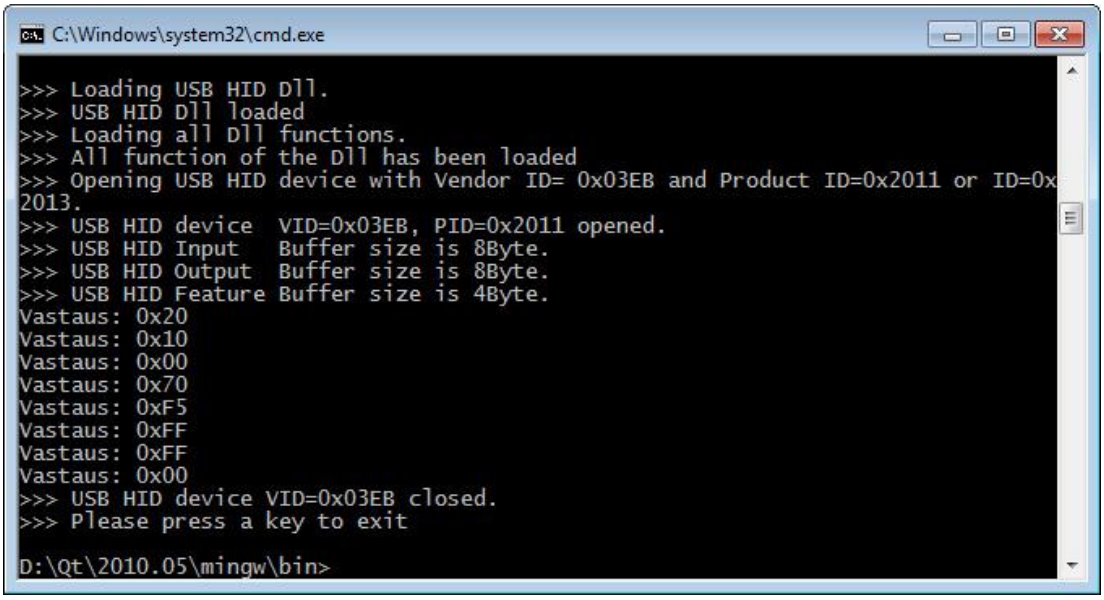
USB-mikrokontrolleri vastaa takaisin samalla komennolla ja tiedoilla, jotka sille on lähetetty. Testauksessa toiminta voidaan varmistaa lukemalla vastauksena saatu tieto.

Kuvasta 23 voidaan määrittää, että

- pyyntö on *USB_CMD_READ_COUNTER* (0x20) eli laskurin arvon lukeminen
- kohdepiirin I²C-osoite on 0x10 (0x10)
- kohdelaskuri, jonka arvo on haettu: 0 (0x00)

- laskurin arvo -2704 pulssia (0x70 (LSB), 0xF5, 0xFF, 0xFF (MSB))

- laskurin suunta on vastapäivään (0x00)



KUVA 23. Laskennan ja kommunikoinnin testaus komentorivisovelluksella

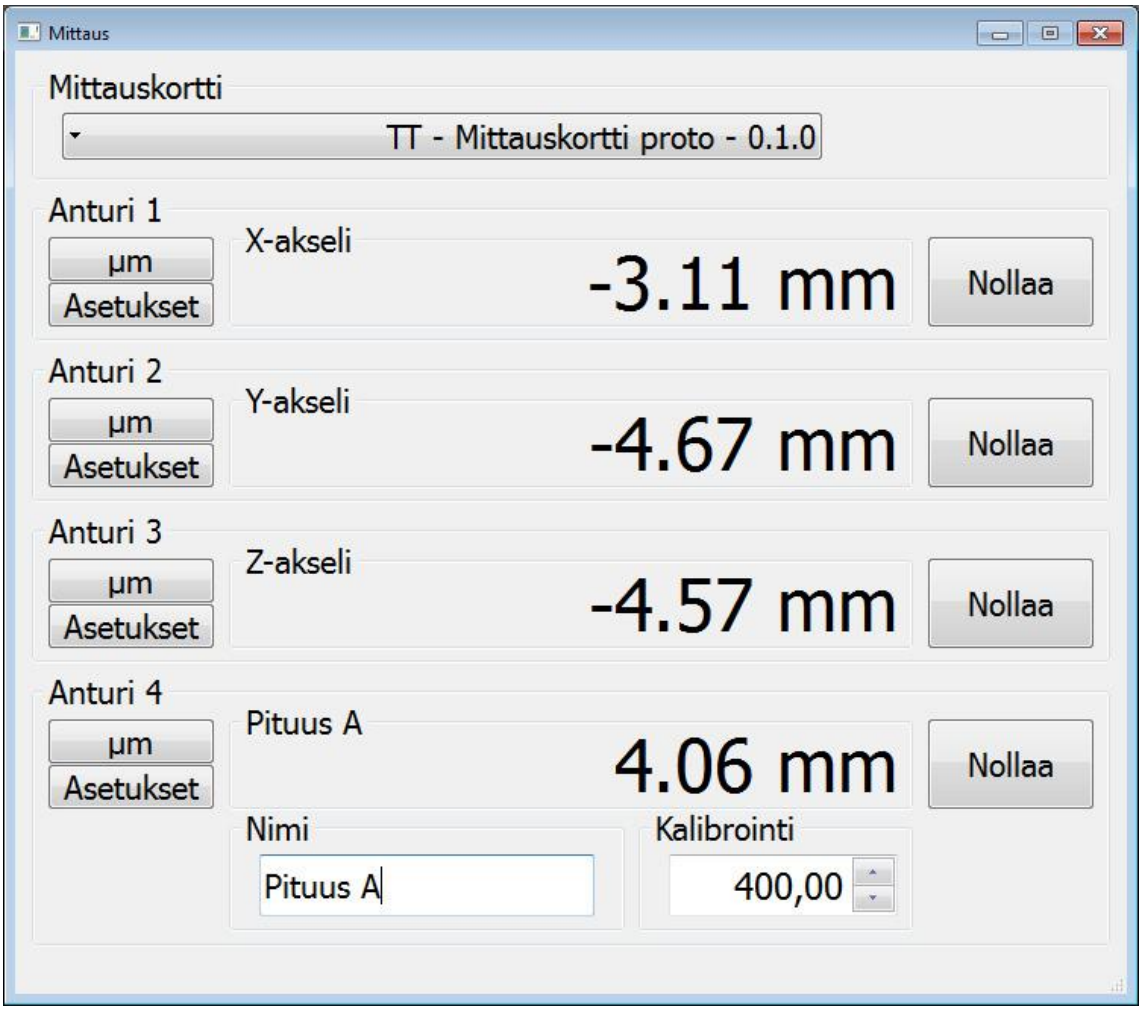
6.4.3 Tietokoneen mittausohjelmisto

Qt-pohjaista mittaussovellusta testattiin kehityksen eri vaiheissa. USB-väylän HID-kommunikointia suorittavan usb_hid-luokan toiminta testattiin läpi huolellisesti ennen muiden luokkien käyttöönottoa. HID-luokan toimintaa testattiin virhetilanteiden varalle muun muassa irrottamalla kortti USB-väylästä kesken käytön ja seuraamalla ohjelman toimintaa. Testien perusteella ohjelman rakenne saatiin muokattua sellaiseksi, että alinta kommunikointia suorittavan libusb:n toiminta saadaan päätettyä oikein myös virhetilanteissa. Jos kirjastoa ei esimerkiksi lopeteta oikein tai sille annetaan virheellisiä syötteitä, koko ohjelmisto kaatuu, jos virhetilanteita ei ole huomioitu. Testaus suoritettiin ajamalla ohjelmaa rivi kerrallaan debug-työkalun avulla ja seuraamalla muuttujien arvoja.

Mittausjärjestelmän käyttöönotosta huolehtivaa meascontrol-luokkaa testattiin myös debug-työkalun avulla ja testien perusteella luokan rakenne muokattiin sellaiseksi, että ohjelmassa ei yritetä edetä seuraavaan vaiheeseen, jos edellistä kohtaa ei ole suoritettu oikein.

Laskennan toimivuutta testattiin kytkemällä anturit mittauskortin tuloihin ja laskentaa seurattiin, kuten kohdassa 6.4.2, mutta käytössä olivat nyt lisäksi graafinen sovellus ja mittausjärjestelmän määrittävät funktiot. Antureiden mittaustuloksista seurattiin erityisesti sitä, että ne säilyttävät saman lukeman, kun anturin akseli pyöritetään takaisin lähtöpaikkaansa. Testien perusteella Qt-sovelluksen eri osa-alueet todettiin toimivaksi.

Tietokoneelle luotu mittaussovellus nähdään kuvasta 24. Anturin 4 nimeäminen sekä kalibrointi on avattu ”Asetukset”-painikkeella näkyviin. Mittauskortin nimi on määritetty piirin sisälle ja se luetaan pudotusvalikkoon valittavaksi.



KUVA 24. Mittausohjelmisto testissä

7 YHTEENVETO

Tämän työn tavoitteena oli kehittää yleiskäyttöinen mittausjärjestelmä, joka voitaisiin integroida mihin tahansa käyttökohteeseen. Vaatimuksiin kuuluivat mm. edullisuus, tuki usealle anturille ja mahdollisuus käyttää 100 kHz:n pulssiaaltoa tuottavia pulssiantureita. Kommunikointi tuli hoitaa USB-väylän kautta tietokoneen mittaussovelluksen kanssa.

Kehitetty järjestelmä voidaan integroida useisiin käyttökohteisiin, koska mittausjärjestelmän ohjelmistoja ei ole suunniteltu vain tiettyä mittaustehtävää suorittamaan, vaan niihin on sisällytetty mahdollisuudet muokata mittausjärjestelmä käyttöympäristöön sopivaksi. Esimerkkinä mainittakoon mittaussovelluksen anturit, jotka voidaan nimetä ohjelmassa käyttöpaikan mukaan esimerkiksi ”Pituus” tai ”X-akseli”. Nykyisen mittauskortin käyttöön erityisen häiriöllisessä ympäristössä tulisi suhtautua kuitenkin varauksella, koska nykyinen versio ei suodata antureilta saatavista signaaleista esimerkiksi yhteismuotoista häiriötä pois.

Laitteiston kokonaishinta on kilpailukykyinen verrattuna mihin tahansa kaupalliseen vastineeseen. Laitteiston komponenteista muodostuvat kustannukset ovat 31,83 euroa ja lista osista on liitteessä 1. Piirilevyn hinta riippuu valmistajasta ja tilatun erän suuruudesta. Yksittäisiä levyjä valmistettaessa esimerkiksi siihen tarkoitetulla protolevyjen työstölaitteella hinta nousee lähelle 100:aa euroa kappaleelta. Valmistettaessa esimerkiksi 20 kappaleen erä PCBCART yrityksellä, hinta on enää noin 4–5 euroa levyltä.

Ensimmäisen piirilevyn teko on luonnollisesti kallista, mutta seuraavat ovat edullisia. Suurissa erissä piirilevyn hinta jää muutaman euroon kappaleelta. Koteloiden hinnat lähtevät kymmenistä euroista ylöspäin, joten yksittäisen laitteen kokonaiskustannukset olisivat noin 57 euroa valmistettaessa 20 kappaleen erä, jos oletetaan, että piirilevy maksaa noin 5 euroa ja kotelo noin 20 euroa.

Kehitetty järjestelmä tukee nykyiseltään neljää anturia ja ohjelmisto on kehitetty niin, että lisäantureiden käyttöönotto on vaivatonta. Laitteistotasolla laajennusmahdollisuus on huomioitu tulevaisuuden versioita varten I^2C-väylän käyttömahdollisuudella.

Laskennallisesti laitteisto tukee antureita 100 kHz:n asti, mutta tämän todentaminen käytännössä on hankalaa, koska se vaatisi todennäköisesti tarkkaa moottorinohjausjärjestelmää anturin akse-

lin pyörittämiseksi sekä toista laskentajärjestelmää, johon saatua tulosta voitaisiin verrata. Karkeiden testien perusteella laskenta kuitenkin tuntuu säilyttävän saman nollakohdan nopeankin akselin edestakaisen pyörittämisen jälkeen ja käytettäessä useita antureita yhtä aikaa. Laskennallisesti laitteiston tulisi kyetä 100 kHz:n taajuuteen, jos käytetään yhtä anturia laskentapiiriä kohti. Kaksi anturia aiheuttaa kaksinkertaisen määrän keskeytyksiä, joten piiri ei välttämättä ehdi laskea kaikkia tilanmuutoksia virheittä. Käytettävä suorituskyky riittää tästä huolimatta useaan käyttökohteeseen, joten voidaan todeta, että työn tavoitteeseen on päästy. Työn viimeinen vaatimus oli laitteiston kommunikointi mittaussovelluksen kanssa USB-väylän kautta ja työn tavoitteeseen päästiin myös tältä osalta.

Mikrokontrollerit sopivat pulssiantureiden laskureiksi tietyin varauksin. Kohteessa, jossa anturin tuottama pulssiaalto pysyy alle 100 kHz:n, vähintään 16 MHz:n kellotaajuudella toimivat mikrokontrollerit sopivat kohteeseen. Yli 100 kHz:n antureiden kanssa on järkevämpää käyttää tarkoitukseen suunniteltua piiriä, koska reilusti suuremmalla kellotaajuudella varustetut mikrokontrollerit nousevat hinnaltaan kalliimmiksi kuin samalla tai paremmalla suorituskyvyllä varustetut kaupalliset laskentapiirit. Joistain Microchip Technologyn valmistamista PIC-mikrokontrollereista löytyy tuki quadrature-koodaukselle laitteistotasolla, mutta yleensä paikka on ainoastaan yhdelle anturille, joten toteutus ei pärjää hintavertailussa kaupalliselle laskentapiirille.

Kehitetylle mittausjärjestelmälle jatkokehitysmahdollisuuksia löytyisi monesta eri osa-alueesta, koska kyseessä on kuitenkin vasta ensimmäinen versio. Tärkeimpänä kehityskohteena itse pitäisin suorituskyvyn kasvattamista, koska nykyinen kortti on suunniteltu maksimissaan 100 kHz:n pulssiaaltoa tuottaville antureille. Suorituskykyä tarvitaan, jos mittausjärjestelmää halutaan käyttää esimerkiksi nopeiden liikkeiden tai kiihtyvyyden mittaamiseen.

Nykyisestä versiosta puuttuu myös anturien signaaleiden häiriönsuodatus, joka löytyy mm. kaupallisista laskentakorteista ja komponenteista. Häiriönsuodatuksen merkitys kasvaa, jos anturia käytetään häiriöisessä ympäristössä, kuten sähkömoottorien läheisyydessä.

Graafinen käyttöliittymä on nykyisessä versiossa hieman hidas toiminnaltaan, koska USB-kommunikointia ei ole eristetty omaan säikeeseensä. Tämä tarkoittaa käytännössä siis sitä, että ohjelman graafista puolta ja USB-kommunikointia ei suoriteta rinnakkain vaan peräkkäin, mikä aiheuttaa mm. ikkunan nykimisen sitä siirrettäessä. Säikeistystä ei lähdetty tekemään ensimmäiseen versioon opinnäytetyölle varatun ajan puitteissa, koska hyvin moniajoa tukevan sovelluksen

toteutus vaatii lisäsuunnittelua ja aikaa. Parempi tuki moniajolle on kuitenkin tarkoitus toteuttaa seuraavassa versiossa, jotta käyttöliittymästä saataisiin ripeämpi ja paremmin käyttäjän toimiin reagoiva.

Jos mittausjärjestelmän mittauskorttiin haluttaisiin vakiona paikat useammalle kuin kahdeksalle anturille, voisi FPGA-piirillä toteutettu ratkaisu osoittautua hinnaltaan ja suorituskyvyltään parhaaksi vaihtoehdoksi. FPGA:lla voitaisiin toteuttaa myös signaaleiden suodatus ja todennäköisesti USB-väylän kommunikointi, joten osamäärä jäisi vähäiseksi.

Jos antureita olisi yhdestä kahdeksaan kappaletta, voisi USB-kommunikointia tukeva mikrokontrolleri yhdessä anturikohtaisilla LS7366R IC-piireillä osoittautua hinnaltaan edullisimmaksi ja suorituskyvyltään parhaimmaksi vaihtoehdoksi. Tällaisella ratkaisulla saataisiin suorituskyvyltään ja ominaisuuksiltaan lähes USB-QUAD08 mittauslaitetta vastaava tuote, jonka osien kokonaiskustannukset olisivat karkealta arviolta 65–100 euron tietämillä kotelointiratkaisun mukaan. USB-QUAD08-laitteelle on ilmoitettu 10 MHz:n maksimi pulssitaajuuden tulkinta ja LS7366R-piirille 9,6 MHz eli suorituskyky olisi jonkin verran huonompi, mutta ei selvästi pienempi.

Arviossa on otettu huomioon laskentapiirien lisäksi myös oheiskomponentit, piirilevy, kotelo ja pientarvikekustannukset. Lisäksi jos nykyisen protolaitteiston ohjelmisto sovitettaisiin uuteen tuotteeseen, siitä saataisiin monipuolisempi kuin kaupallisesta vastineestaan esimerkiksi laajennettavuudeltaan. Ongelmia tuottaa kuitenkin edelleen komponenttien heikko saatavuus, joten tilauserästä pitäisi tehdä heti kerralla suurempi.

Jos mittauskortille ei tarvita suurta suorituskykyä, soveltuvat mikrokontrollerit myös mainiosti laskennan toteuttamiseen edullisen hintansa, ominaisuuksien ja hyvän saatavuuden puolesta. Mikrokontrollereja on toki saatavilla myös isommalla kellotaajuudella, mutta hinta noussee korkeaksi.

8 POHDINTA

Työ osoittautui haastavaksi ja mielenkiintoiseksi. Uutta asiaa työn aikana tuli vastaan monelta osa-alueelta. Pintaliitoskomponenttien ja erityisesti tiheäjalkaisten piirien juottaminen ei ollut aiemmin tuttua, mutta osoittautui työn aikana odotettua huomattavasti helpommaksi.

USB- ja I^2C-väylän kommunikoinnin toteutus laitteisto- sekä ohjelmistotasolla olivat täysin uusia tuttavuuksia työn tekijälle. Molemmat tuottivat aluksi enemmän tai vähemmän ongelmia kommunikoinnissa ohjelmistoa toteutettaessa, mutta useamman päivän ongelmanratkonnan jälkeen yhteydet alkoivat toimia niin piirien välillä, kuin tietokoneen ja mittauskortin välillä. Qt oli vaikuttanut jo pidemmän aikaa mielenkiintoiselta ohjelmointikieleltä ja tutustumisen jälkeen se osoittautui todella hyväksi vaihtoehdoksi erityisesti mahdollisimman alustariippumatonta sovellusta kehittäessä.

Työn tuloksena voidaan todeta, että hyvin suorituskykyisen ja monipuolisen laskentakortin toteutus vie aikaa, mutta työhön tarvittavat osat ovat suhteellisen edullisia. Kun yhden kortin ohjelmiston ja laitteiston on saanut suunniteltua, ovat seuraavat versiot huomattavasti edullisempia tuottaa.

LÄHTEET

1. Rotary Encoders Technical Guide. 2011. Omron. Saatavilla: http://www.ia.omron.com/data_pdf/guide/52/rotary_tg_e_4_1_4-11(further_info).pdf. Hakupäivä 3.1.2011.

2. Rotary Encoder. 2008. Omron. Saatavilla: http://www.ia.omron.com/data_pdf/data_sheet/e6c2-c_dsheet_csm493.pdf. Hakupäivä 13.1.2011.

3. Pulssianturi E6C2-C. 2011. Omron. Saatavilla: http://downloads.industrial.omron.eu/IAB/Products/Sensing/Rotary%20Encoders/Incremental/E6C2-C,%20E6C3-C/Q109/Q109-FI2-01+E6C2-C+Datasheet.pdf. Hakupäivä 13.1.2011.

4. Quadrature Encoder Velocity and Acceleration Estimation with CompactRIO and LabVIEW FPGA. 2010. National Instrument. Saatavilla: http://zone.ni.com/devzone/cda/tut/p/id/3921. Hakupäivä 23.3.2011.

5. XOR. 2011. Germundsson, R. & Weisstein, E. W. Saatavilla: http://mathworld.wolfram.com/XOR.html. Hakupäivä 26.4.2011.

6. USB-QUAD08. 2011. Measurement Computing. Saatavilla: http://www.mccdaq.com/usb-data-acquisition/USB-QUAD08.aspx. Hakupäivä 23.3.2011.

7. 1047 - PhidetEncoder HighSpeed 4-Input. 2010. Phidgets. Saatavilla: http://www.phidgets.com/products.php?category=7&product_id=1047. Hakupäivä 23.3.2011.

8. LS7366R 32-bit Quadrature Counter with Serial Interface. 2009. LSI/CSI. Saatavilla: http://www.lsicsi.com/pdfs/Data_Sheets/LS7366R.pdf. Hakupäivä 3.1.2011.

9. Quadrature Decoder/Counter Interface ICs. 2007. Avago Technologies. Saatavilla: http://www.avagotech.com/docs/AV02-0096EN. Hakupäivä 3.1.2011.

10. Universal Serial Bus. 2007. Intel. Saatavilla: http://www.intel.com/technology/usb/. Hakupäivä 23.3.2011.

11. Device Class Definition for Human Interface Devices (HID). 2001. USB Implementers' Forum. Saatavilla: http://www.usb.org/developers/devclass_docs/HID1_11.pdf. Hakupäivä 24.3.2011.

12. Industrial Reference Design Platform I²C. 2011. NXP. Saatavilla: http://ics.nxp.com/support/boards/ird/pdf/training.i2c.pdf. Hakupäivä 26.3.2011.

13. ATmega32U4. 2010. Atmel. Saatavilla: http://www.atmel.com/dyn/resources/prod_documents/doc7766.pdf. Hakupäivä 31.3.2011.

14. ATmega168. 2010. Atmel. Saatavilla: http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf. Hakupäivä 31.3.2011.

15. AVR315: Using the TWI module as I²C master. 2010. Atmel. Saatavilla: http://www.atmel.com/dyn/resources/prod_documents/doc2564.pdf. Hakupäivä 26.4.2011.

16. AVR311: Using the TWI module as I²C slave. 2009. Atmel. Saatavilla: http://www.atmel.com/dyn/resources/prod_documents/doc2565.pdf. Hakupäivä 26.4.2011.

17. Porting From IAR to AVR GCC. 2011. AVR Libc. Saatavilla: http://www.nongnu.org/avr-libc/user-manual/porting.html. Hakupäivä 26.3.2011.

18. Blanchette, J. & Summerfield, M. 2009. C++ GUI programming with Qt 4. 4. painos. USA: Prentice Hall.

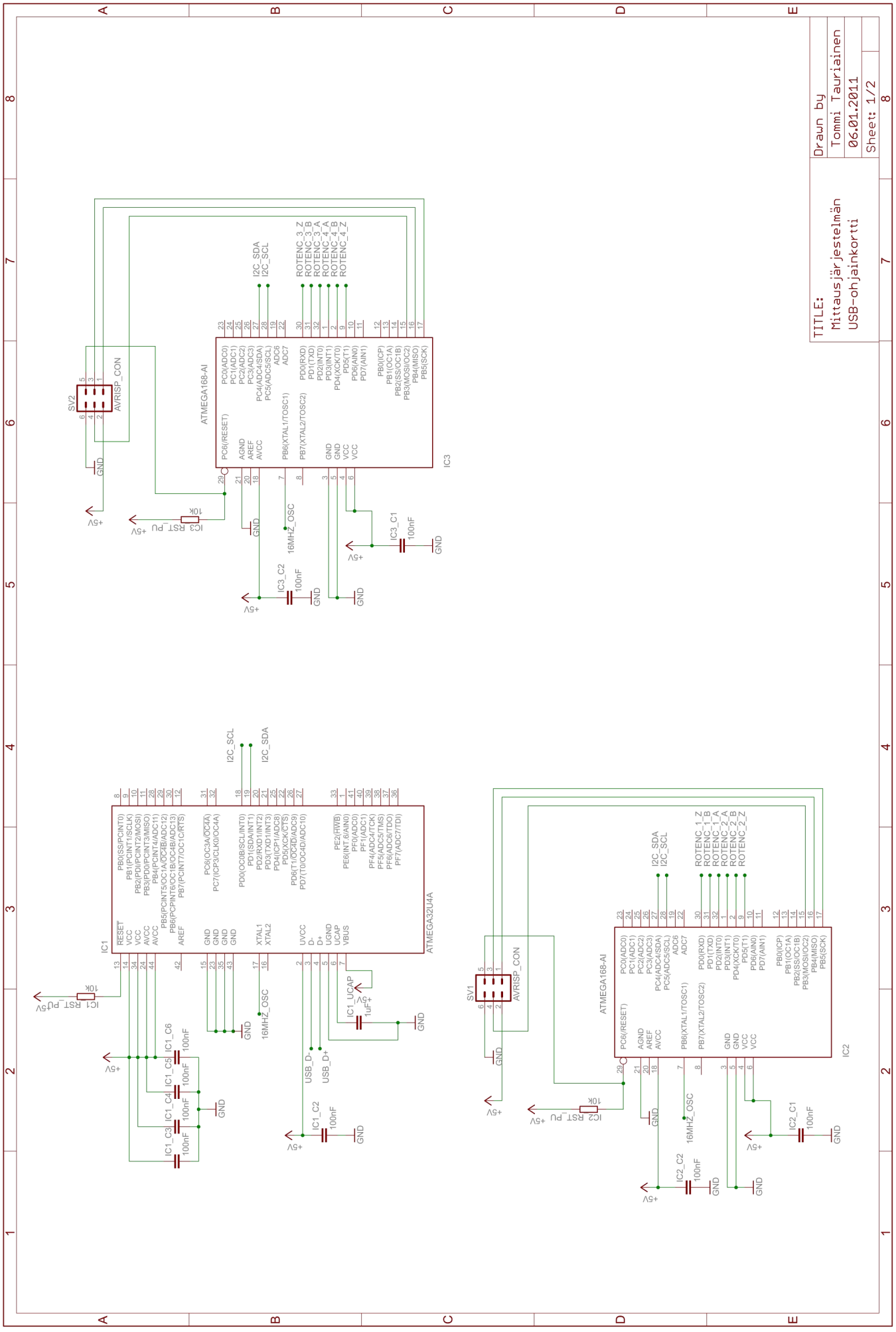
19. Windows backend. 2011. Libusb. Saatavilla: http://www.libusb.org/wiki/windows_backend. Hakupäivä 26.3.2011.

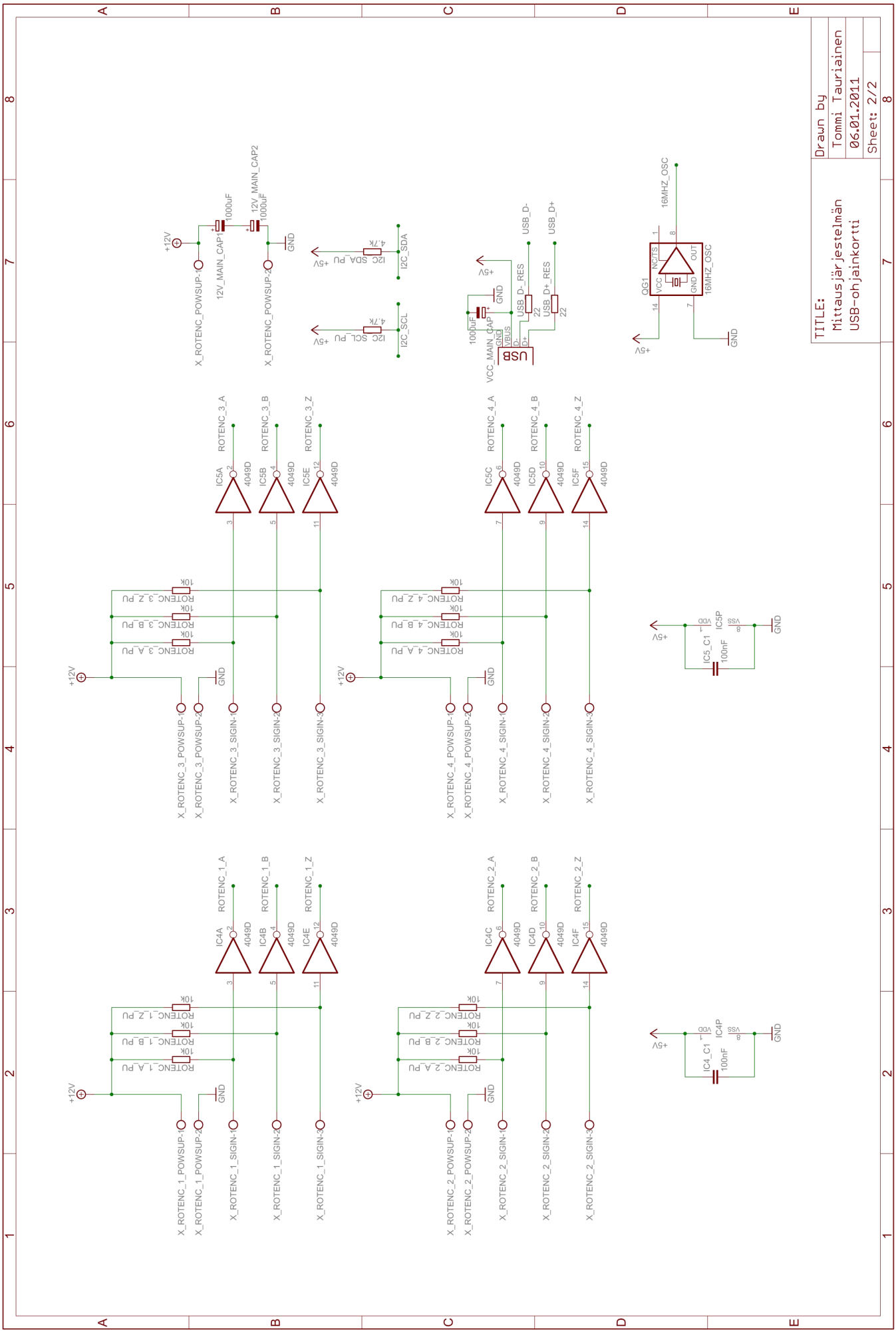
20. USB PC Drivers Based on Generic HID Class. 2008. Atmel. Saatavilla: http://www.atmel.com/dyn/resources/prod_documents/doc7645.pdf. Hakupäivä 26.3.2011.

21. Ming, Zhou 2009. Beijingcode, Quadcopter. Saatavilla: http://beijingcode.org/projects/quadcopter/browser/people/chowming/hw_library_demo?rev=6eb76170680e7e87eee1e3fc0f2acb4e06bb318b. Hakupäivä 15.3.2011.

MITTAUSKORTIN KOMPONENTIT LIITE 1

Komponentti	kpl	à hinta	yht.	
IC-piiri, ATmega32U4	1	6,00	6,00	
IC-piiri, ATmega168	2	4,94	9,88	
IC-piiri, 4049	2	0,35	0,70	
Oskillaattori, DIL-14, 16 MHz	1	3,50	3,50	
Vastus, 1206, 22Ω	2	0,05	0,10	
Vastus, 1206, 4,7 kΩ	2	0,05	0,10	
Vastus, 1206, 10 kΩ	15	0,05	0,75	
Kondens. 1206, 100 nF	11	0,05	0,55	
Kondens. 1206, 1 uF	1	0,05	0,05	
Kondens. ELKO, 1000 uF	3	0,60	1,80	
Piikirima 2x20	1	0,80	0,80	
Liitin, USB-B piirilevylle	1	1,00	1,00	
Ruuviliitosrima, 2-os	11	0,60	6,60	
Yhteensä			31,83	€


TITLE:
Mittausjärjestelmän
USB-ohjainkortti
Draun by
Tommi Tauriainen
06.01.2011
Sheet: 1/2
ATMEGA168-AI
IC3
SV2
AVRISP_CON
IC3_RST_PU
10k
16MHZ_OSC
IC3_C1
IC3_C2
100nF
I2C_SDA
I2C_SCL
ROTENC_3_Z
ROTENC_3_B
ROTENC_3_A
ROTENC_4_A
ROTENC_4_B
ROTENC_4_Z
ATMEGA32U4A
IC1
SV1
IC1_RST_PU
IC1_C1
IC1_C2
IC1_C3
IC1_C4
IC1_C5
IC1_C6
IC1_UCAP
1uF
USB_D-
USB_D+
ATMEGA168-AI
IC2
IC2_RST_PU
IC2_C1
IC2_C2
+5V
GND


TITLE:
Mittausjärjestelmän
USB-ohjainkortti
Drawn by
Tommi Tauriainen
06.01.2011
Sheet: 2/2
+12V
12V_MAIN_CAP1
1000uF
12V_MAIN_CAP2
1000uF
GND
X_ROTENC_POWSUP-1
X_ROTENC_POWSUP-2
I2C_SDA_PU
4.7k
I2C_SDA
I2C_SCL_PU
I2C_SCL
+5V
100uF
VCC_MAIN_CAP
USB
VBUS
D-
D+
GND
USB_D-
USB_D+
USB_D-_RES
USB_D+_RES
22
16MHZ_OSC
QG1
NC/TS
VCC
GND
OUT
IC5_C1
IC4_C1
100nF
IC5P
IC4P
VDD
VSS
IC5A
IC5B
IC5C
IC5D
IC5E
IC5F
IC4A
IC4B
IC4C
IC4D
IC4E
IC4F
4049D
ROTENC_3_A
ROTENC_3_B
ROTENC_3_Z
ROTENC_4_A
ROTENC_4_B
ROTENC_4_Z
ROTENC_1_A
ROTENC_1_B
ROTENC_1_Z
ROTENC_2_A
ROTENC_2_B
ROTENC_2_Z
10K
ROTENC_3_A_PU
ROTENC_3_B_PU
ROTENC_3_Z_PU
ROTENC_4_A_PU
ROTENC_4_B_PU
ROTENC_4_Z_PU
ROTENC_1_A_PU
ROTENC_1_B_PU
ROTENC_1_Z_PU
ROTENC_2_A_PU
ROTENC_2_B_PU
ROTENC_2_Z_PU
X_ROTENC_3_POWSUP-1
X_ROTENC_3_POWSUP-2
X_ROTENC_3_SIGIN-1
X_ROTENC_3_SIGIN-2
X_ROTENC_3_SIGIN-3
X_ROTENC_4_POWSUP-1
X_ROTENC_4_POWSUP-2
X_ROTENC_4_SIGIN-1
X_ROTENC_4_SIGIN-2
X_ROTENC_4_SIGIN-3
X_ROTENC_1_POWSUP-1
X_ROTENC_1_POWSUP-2
X_ROTENC_1_SIGIN-1
X_ROTENC_1_SIGIN-2
X_ROTENC_1_SIGIN-3
X_ROTENC_2_POWSUP-1
X_ROTENC_2_POWSUP-2
X_ROTENC_2_SIGIN-1
X_ROTENC_2_SIGIN-2
X_ROTENC_2_SIGIN-3