Jussi Takala

# MONITOROINTILAITE TÄRINÄTESTAUSJÄRJESTELMÄLLE

Tietotekniikan koulutusohjelma

2011

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

# MONITOROINTILAITE TÄRINÄTESTAUSJÄRJESTELMÄLLE

Takala, Jussi
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Marraskuu 2011
Ohjaaja: Peltonen, Kauko
Sivumäärä: 30
Liitteitä: 7

Asiasanat: atmel, mikro-ohjain, tärinätesti, monitorointi

---

Opinnäytetyö toteutettiin Rauman EMC-laboratoriolle, joka tilasi työn helpottaakseen tärinätestauksessa käytettävää visualisointikeinoa komponenttien resonanssipisteiden havainnollistamiseksi. Tarkoituksena oli kehittää laite, jonka avulla saadaan testattavien laitteiden heikoimmat komponentit paikannettua nykyistä systeemiä helpommin. Alkuperäisen systeemin ongelmana oli, että sitä ei voitu käyttää sulavasti vaihtuvilla taajuuksilla, vaan ainoastaan yksittäisillä taajuuksilla kerrallaan.

Työn lähtökohtana oli kehittää mikro-ohjaimeen pohjautuva laite tärinäohjausjärjestelmän ja stroboskoopin välille. Laitteen tulisi analysoida ohjausjärjestelmältä tulevaa signaalia ja lähettää haluttua signaalia stroboskoopille.

Laitetta suunnitellessa kävi ilmi, että se kannattaa rakentaa kahden mikro-ohjaimen ympärille, jotta se toimisi reaaliajassa vaihtelevilla taajuuksilla. Laitteen ytimeksi valittiin kaksi ATmega32-mikro-ohjainta. Master-ohjain analysoi tärinätestausjärjestelmältä tulevaa signaalia ja lähettää taajuustiedon slave-ohjaimelle, joka lähettää tarvittavan signaalin eteenpäin stroboskoopille. Päätettiin, että laite sisältäisi LCD-näytön, joka näyttäisi tulevan taajuuden ja muuta informaatiota. Laitteeseen haluttiin myös kaksi kytkintä, joilla pystyttäisiin muuntamaan ulostulosignaalia.

EMC-laboratorio sai lopullisen laitteen, ja se tulee olemaan tärkeä väline tärinätestauksessa. Monitorointilaite testattiin laboratorion oikeissa olosuhteissa ja sen todettiin toimivan halutulla tavalla. Vaikka laite toimi hyvin ja täytti sille määritellyt vaatimukset, jäi siihen kuitenkin kehittämismahdollisuuksia tulevaisuutta varten, mikä ei ole ihme, koska kyseessä on prototyyppi.

A MONITORING DEVICE FOR THE VIBRATION TEST SYSTEM

Takala, Jussi
Satakunta University of Applied Sciences
Information Technology
November 2011
Supervisor: Peltonen, Kauko
Number of pages: 30
Appendices: 7

Keywords: atmel, microcontroller, vibration test, monitoring

---

This thesis was made for Rauma EMC laboratory, which ordered the thesis to facilitate the visualization for illustration of components resonant points in the vibration test. The purpose of the thesis was to develop a device which helps you to find the weakest components of tested device more easily than with the current system. The problem of the original system was that it could not be used on the run with variable frequencies, but only with one frequency at a time.

The starting point of the project was to develop a microcontroller based device between the vibration test system and the strobe. The device should analyze the signal from the vibration control system and send desired signal to strobe.

When designing the device it came out that it is better to build the device round two microcontrollers, so that the device works in real time with variable frequencies. Two ATmega32 microcontrollers were selected to be the core of the device. One controller analyzes incoming signal from vibration test system and sends the data to the other controller, which sends the required signal to the strobe. It was decided that the device would include a LCD display, which would display the incoming frequency and other information. It was also desired that the device would contain two switches, which could modify the output signal.

EMC laboratory implemented the final device and it will be an important instrument in vibration test. The monitoring device was tested in the real circumstances of the laboratory and it was found to work the desired way. Even though the device worked well and met the requirements, there remained it opportunities for development for the future, which is not surprising because it is a prototype.

LYHENNELUETTELO

| | |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| BNC | Bayonet Neill-Concelman |
| COLA | Constant Output Level Adapter |
| EMC | Electromagnetic compatibility |
| ETA | Euroopan talousalue |
| FPM | Flashes Per Minute |
| I/O | Input/Output |
| LCD | Liquid Crystal Display |
| MLF | MicroLeadFrame |
| PDIP | Plastic Dual Inline Package |
| SPI | Serial Peripheral Interface |
| TQFP | Thin Quad Flat Pack |
| TTL | Transistor-transistor logic |

SISÄLLYS

# 1 JOHDANTO

Työn tarkoituksena oli kehittää tärinätestauslaitteen ohjausjärjestelmän ja stroboskoopin välille elektroninen laite, jolla voidaan muuttaa ja säätää stroboskoopille menevää taajuutta muutamilla hertseillä.

Tärinätestauksessa on käytettävissä vain kaksi anturia, joista toisella mitataan alustan ja toisella halutun komponentin liikettä. Kun verrataan näiden kahden anturin arvoja, saadaan selville kuinka hyvin mitattava komponentti kestää rasitusta ja minkä verran se elää alustalla. Työn tilaaja Rauman EMC-laboratorio käyttää tärinätestauksessa visualisointikeinona strobovaloa, jonka avulla testattavien laitteiden komponenttien resonanssipisteet löydetään paljaalla silmällä. Tämä visualisointikeino on tärkeä menetelmä, sillä normaalisti testattavat laitteet sisältävät satoja ellei jopa tuhansia komponentteja, joiden jokaisen erillinen testaaminen antureiden avulla on käytännössä mahdotonta.

Laboratoriolla on käytössään Monarch Palm Strobe -stroboskooppi, jossa on TTL-tason 3,5 mm stereoplugiliitin, jonka kautta stroboskoopille voidaan syöttää suoraan haluttua taajuutta, jonka tahdissa strobovalo vilkkuu. Ongelmana on, että tutkittaessa tärinätestauslaitteessa olevaa tuotetta, tulisi stroboskoopin taajuuden olla muutaman hertsin poikkeavaa tärinätestilaitteeseen menevästä taajuudesta, jotta laite nähdään eri tasoissa ja näin ollen epävakaiden komponenttien resonanssipisteet saataisiin näkyviin paljaalla silmällä.

Stroboskoopissa on mahdollisuus säätää manuaalisesti toimintataajuutta, mutta se on hidasta ja hankalaa, koska stroboskoopissa taajuus määritellään FPM-yksikkönä, kun taas LDS Dactronista lähtevä signaali määritellään hertseinä. Tämän työn tarkoituksena on siis kehittää laite, joka seuraa tulevaa taajuutta ja poikkeuttaa lähtevää taajuutta automaattisesti.

# 2 RAUMAN EMC-LABORATORIO

Tämän opinnäytetyön tilaajana toimii Rauman EMC-laboratorio, joka tarjoaa yrityksille monipuolisia elektroniikan tuotekehitys- ja testauspalveluja. Standardien mukaisten EMC-testien lisäksi laboratorion palveluihin kuuluvat EMC-koulutus, laitteistojen ja tilojen vuokraus sekä elektroniikan tuotekehityspalvelut. Monipuolinen tarjonta mahdollistaa tuotteen korjauttamisen välittömästi mahdollisten virheiden osalta. Lisäksi laboratoriossa on mahdollista suorittaa olosuhdetestauksia, joihin kuuluvat säätesti, shokkitesti sekä tärinätesti. /1/

## 2.1 Yleistä EMC:stä

Sähkömagneettinen yhteensopivuus (EMC) tarkoittaa elektronisen laitteen tai järjestelmän kykyä toimia luotettavasti luonnollisessa toimintaympäristössään. /2/ Käytännössä tämä tarkoittaa, että laitteen on kestettävä tietty määrä sähkömagneettista häiriötä ja vastaavasti laite ei itse saa aiheuttaa liikaa sähkömagneettista säteilyä ympäristöön. /3/

## 2.2 EMC-direktiivi

EMC-direktiivin päätarkoitus on taata laitteiden vapaa liikkuvuus ja saada aikaan hyväksyttävä sähkömagneettinen ympäristö ETA:n sisällä. Käytännössä tämä koskee laitteita, jotka saattavat aiheuttaa sähkömagneettista häiriöitä tai alistua sellaisille. Mikäli sähkö- tai elektroniikkalaitetta halutaan myydä ETA:n sisällä, sen on ensin läpäistävä EMC-direktiivissä asetetut EMC-standardit. EMC-direktiivin päätavoitteena on varmistaa, että sähkö- tai elektroniikkalaitteiden aiheuttamat häiriöt eivät vaikuta muiden EMC-direktiivin artiklan 1.1 mukaisten laitteiden, eivätkä radio- ja televerkkojen, niiden laitteiden ja sähkönjakeluverkkojen virheettömään toimintaan. /4/

## 2.3 Tärinätestaus

Tärinätestauksessa on tarkoitus löytää materiaalien ja rakenteiden mahdolliset heikot kohdat jo tuotekehitysvaiheessa, jotta tuotteeseen voidaan tehdä tarvittavat korjaukset, ennen kuin se menee asiakkaalle. Tärinätestaus tuo luotettavuutta laitteen kestävyyteen, sillä testauksessa todennetaan laitteen kestävyys sille tarkoitetussa ympäristössä.

Tärinätestaukseen sisältyy usein useita erilaisia mekaanisia rasitusmenetelmiä. Näitä menetelmiä ovat esimerkiksi resonanssihaku, sinipyyhkäisy, satunnaistärinä ja iskutestaus. Tähän työhön liittyvää silmin havaittavaa komponenttien liikettä, joka toteutetaan stroboskoopilla, hyödynnetään sinipyyhkäisytestauksessa, jolloin tärinän taajuus on tiedossa, minkä avulla säädetään stroboskoopin taajuutta.

## 2.4 Laboratoriossa käytettävät laitteet tärinätestauksessa

Kuten jo aiemmin mainittua, monitorointilaite suunniteltiin tärinätestausjärjestelmän ja stroboskoopin välille. Seuraavassa on esitelty nämä laitteet.

### 2.4.1 Tärinätestauslaite LDS V830T

Tärinätestauslaite LDS V830T (kuva 1.) on laite, jolla voidaan testata esimerkiksi komponenttien tai lopputuotteiden laatua, kulutuskestävyyttä ja rasituskestävyyttä. Testit voidaan tehdä standardien mukaisesti tai tapauskohtaisesti, jolloin saadaan mahdollisimman hyvin testattua testattavan laitteen toimivuus halutuissa olosuhteissa. Laboratorion testiympäristö sisältää itse tärinätestauslaitteen lisäksi SPA 16K 8kW -tehovahvistimen, Laser Shaker Control System -täristinohjaimen sekä tiedonkeruulaitteet ja -ohjelmat.

Käytännössä testejä hallitaan tietokoneella, jossa on tärinätestauksen hallinta- ja tiedonkeruuohjelmat. Hallintaohjelmilla määritellään testien tyypit, laajuudet, kestot ja muut asetukset. Tiedonkeruuohjelmilla voidaan seurata esimerkiksi testauksessa käytettävien antureiden liikettä ja liikkeiden poikkeamaa todellisesta tärinätaajuudesta.

Tietokoneelta siirretään dataa Laser Shaker Control System -täristinohjaimelle, joka ohjaa täristintä SPA16K tehovahvistimen kautta. Täristinohjaimelta lähettää COLA-signaalia, joka on puhdasta sinisignaalia ja samaa taajuutta täristimelle menevän taajuuden kanssa, mutta pienemmällä amplitudilla. Tätä COLA-signaalia monitorointilaite hyödyntää analysoidessaan täristimelle menevää signaalia.


Kuva1. Tärinätestauslaite LDS V830T /1/

LDS V830T:n tekniset tiedot:

Taajuusalue 0-53000 Hz
Suurin sinivoima 6,54 kN (peak)
Suurin satunnaisvoima 7,60 kN (rms)
Suurin sinikiihtyvyys 55.3 (peak)
Suurin satunnaiskiihtyvyys 60 g (rms)
Suurin sininopeus 2,0 m/s (peak)
Suurin amplitudi 50,8 mm (peak to peak)
Suurin testikuorma 160 kg /4/

### 2.4.2 Stroboskooppimittalaite Monarch Palm Strobe

Monarch Palm Strobe (Kuva 2.) on helppokäyttöinen ja kevyt stroboskooppi, jonka välähdystaajuus toimii 100 – 12500 FPM alueella ±0,01 % tarkkuudella. Se sisältää 6-numeroisen LCD-näytön ja sen muistiin voidaan tallentaa 8 eri välähdysnopeutta. TTL-yhteensopiva I/O-liitin mahdollistaa stroboskoopin liittämisen muihin laitteisiin, mikä mahdollistaa esimerkiksi stroboskoopin taajuuden säätämisen ulkoisesti erillisellä laitteella. /5/

Kuva 2. Monarch Palm Strobe -stroboskooppi /5/

Stroboskoopin avulla voidaan niin sanotusti jäädyttää pyörivät koneet eli visualisoida siten, että kone näyttäisi olevan paikallaan, jolloin koneen tutkiminen on mahdollista pysäyttämättä konetta. Tärinätestauksen resonanssipisteiden visualisoinnissa halutaan hiukan poikkeutettua taajuutta testattavan laitteen tärinätaajuudesta. Koska sellaista toimintoa ei löydy stroboskoopista, Rauman EMC-laboratorio tilasi tämän työn.

# 3 VAATIMUSMÄÄRITTELY

## 3.1 Toiminnalliset vaatimukset

Tärinätestausjärjestelmä toimii noin 1-53000 Hz:n alueella, mutta tämän työn toiminta-alueeksi määriteltiin 1-300 Hz, koska suuremmat taajuudet ovat liian nopeita silmälle havaittaviksi. Laitteen piti pystyä analysoimaan LDS Dactronilta tulevaa sinisignaalia sekä lähettämään kanttisignaalia eteenpäin stroboskoopille.

Koska tässä työssä kehiteltävän monitorointilaitteen tarkoitus ei ollut suoranaisesti olla taajuusmittari eikä taajuusgeneraattori, puolen hertsin tarkkuus oli riittävä sekä taajuuden mittauksessa että lähettämisessä. Tämä mahdollisti sen, ettei ohjelmiston eikä laitteiston tarvinnut olla kovin monimutkainen, mikä tarkoitti sitä, että laitteen käytettävyys, toimivuus sekä virrankulutus saatiin paremmaksi.

## 3.2 Mekaaniset vaatimukset

Laitteessa tulisi olla kaksi kytkintä, joilla voidaan poikkeuttaa lähtevää taajuutta tarpeen mukaan joko korkeammaksi tai matalammaksi. Päätettiin, että taajuutta pitäisi voida poikkeuttaa ±1 Hz yhdellä painalluksella, jolloin taajuutta saatiin askellutettua yhden hertsin sykleissä haluttuun arvoon asti. Lisäksi laitteessa tuli olla reset-kytkin, jolla laite saadaan nopeasti alkutilaan.

Laitteeseen tulevan sini-aallon taajuutta piti monitoroida LCD-näytöltä, josta kävisi ilmi tärinätestauslaitteelle syötettävä taajuus. Laitteen käyttäjä näkisi heti näytöltä millä taajuuksilla testattava laite olisi epävakaa. Lisäksi sovittiin, että näytölle tulisi muuta tarpeellista informaatiota.

## 3.3 Sähköiset vaatimukset

Laitteen tuli toimia viiden voltin käyttöjännitteellä, jolla laitteen kaikki komponentit toimivat ilman jännitteregulointia. Sisään tuleva COLA-signaali oli oltava vahvuudel-

taan 1 V (peak-to-peak), joten laitteen oli vahvistettava tätä signaalia ennen mikro-ohjaimelle siirtämistä, koska kyseessä ei ollut TTL-tason signaali. Lisäksi ulos tulevan signaalin vahvuus oli oltava TTL-tasoista, jotta stroboskooppi ymmärsi sitä.

Standardiset TTL-tason logiikkapiirit toimivat 5 V:n käyttöjännitteellä. TTL-tason mikropiireissä sisääntulosignaali on määritelty loogiseksi nollaksi, kun jännite on 0-0,8 V, ja ykköseksi, kun jännite on 2,2-5V. Ulostulosignaalit ovat rajoitettu tiukemmin, jolloin looginen nolla on määritelty 0-0,4 V:n välille ja ykkönen 2,6-5 V:n välille.

# 4 LAITTEISTON TOTEUTUS

## 4.1 Monitorointilaitteen kokoonpano

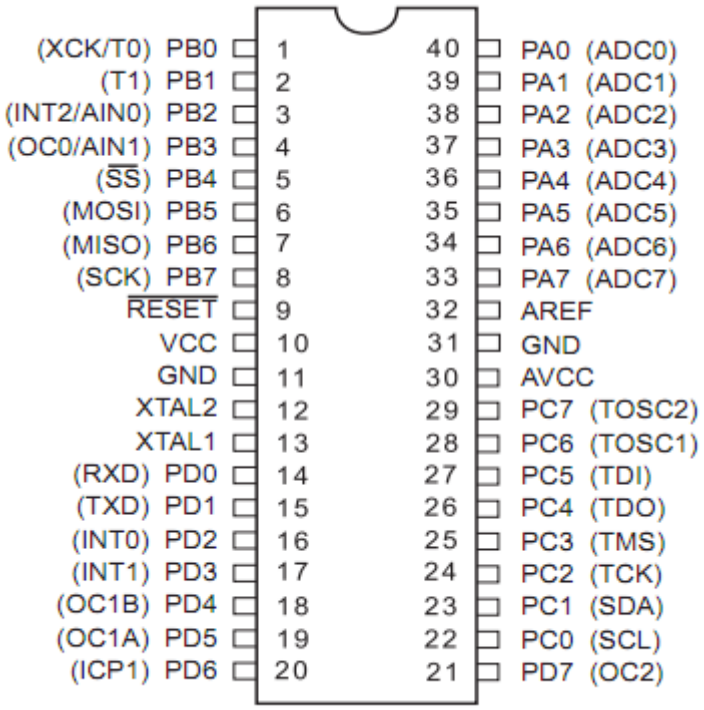
Koska laitetta suunnitellessa kävi ilmi, että yksi mikro-ohjain ei riitä sekä taajuuden reaaliaikaiseen mittaamiseen että lähettämiseen, oli työssä käytettävä kahta mikro-ohjainta. Laitteen ytimenä toimivat kaksi Atmelin ATmega32-ohjainta, joista toinen toimii master- ja toinen slave-ohjaimena. Master-ohjain analysoi ja laskee laitteeseen tulevan taajuuden sekä lähettää sen eteenpäin slave-ohjaimelle. Master-ohjaimeen liitetään sisääntuloksi BNC-liitin sekä PORTB:stä lähtevä SPI-väylä, joka menee suoraan slaven PORTB:hen. Slave-ohjaimeen liitetään SPI-väylän lisäksi kaksi kytkintä, BNC-liitin sekä LCD-näyttö. Laite ottaa vastaan dataa kytkimiltä sekä ohjausjärjestelmältä sini-signaalin muodossa ja vastaavasti lähettää dataa LCD-näytölle sekä stroboskoopille. Molempiin ohjaimiin lisätään yhteinen reset-kytkin, jolla saadaan koko järjestelmä asetettua alkutilaan.

## 4.2 Mikro-ohjain

Monitorointilaitteen mikro-ohjaimeksi valittiin Atmelin ATmega32-ohjain, koska siitä löytyy riittävä määrä I/O-liitäntöjä sekä tarpeelliset ajastimet. Mikro-ohjaimiksi olisi varmasti löytynyt lukuisia muitakin vaihtoehtoja, mutta tähän päädyttiin, koska

kyseessä on monipuolinen sekä kohtuuhintainen ohjain, joka on varmasti riittävä ominaisuuksiltaan.

Ohjaimen pakkaustyyppi on PDIP, joka tarkoittaa sitä, että I/O-nastat sijaitsevat piirin kahdella sivulla, ja että kotelon materiaalina on muovi. Ohjaimessa on 40 signaalilinjaa, joiden merkitykset ja järjestyksen näkee kuvasta 3. Prosessorissa on kaikkiaan neljä 8-bittistä porttia (PORTA, PORTB, PORTC ja PORTD), jotka voidaan määritellä joko tuloiksi tai lähdöiksi. Muut signaalilinjat ovat sähkönsyöttöön, resetointiin sekä ulkoisen kellokiteen käyttöön tarkoitettuja nastoja /6/. Mikro-ohjaimen tarkemmat tekniset tiedot ja ominaisuudet ovat nähtävissä ATmega32:n datalehden ensimmäiseltä sivulta (LIITE1).




Kuva 3. ATmega32:n signaalilinjojen järjestys PDIP-kotelossa. /6/

## 4.3 LCD-näyttö

Laitteiston näytöksi valittiin merkkipohjainen 4x20 LCD -näyttö. Näyttö kytketään laitteeseen 16 piikkisen liittimen välityksellä. Taulukossa 1 on esitetty LCD-näytön liittimien toiminnat.

LCD-näytön kytkennässä otettiin huomioon se, että näytön 8-bittisestä rinnakkaismuotoisesta dataväyläliitännästä käytetään vain neljää ylintä bittiä. ASCII-merkit kirjoitetaan näytölle kahdessa osassa. Ensin kirjoitetaan neljä alinta bittiä ja sen jälkeen neljä ylintä bittiä. Tällä tavoin näyttöä säädetään neljällä I/O-linjalla.

Taulukko 1. LCD-näytön signaalilinjojen järjestys ja symbolien kuvaukset

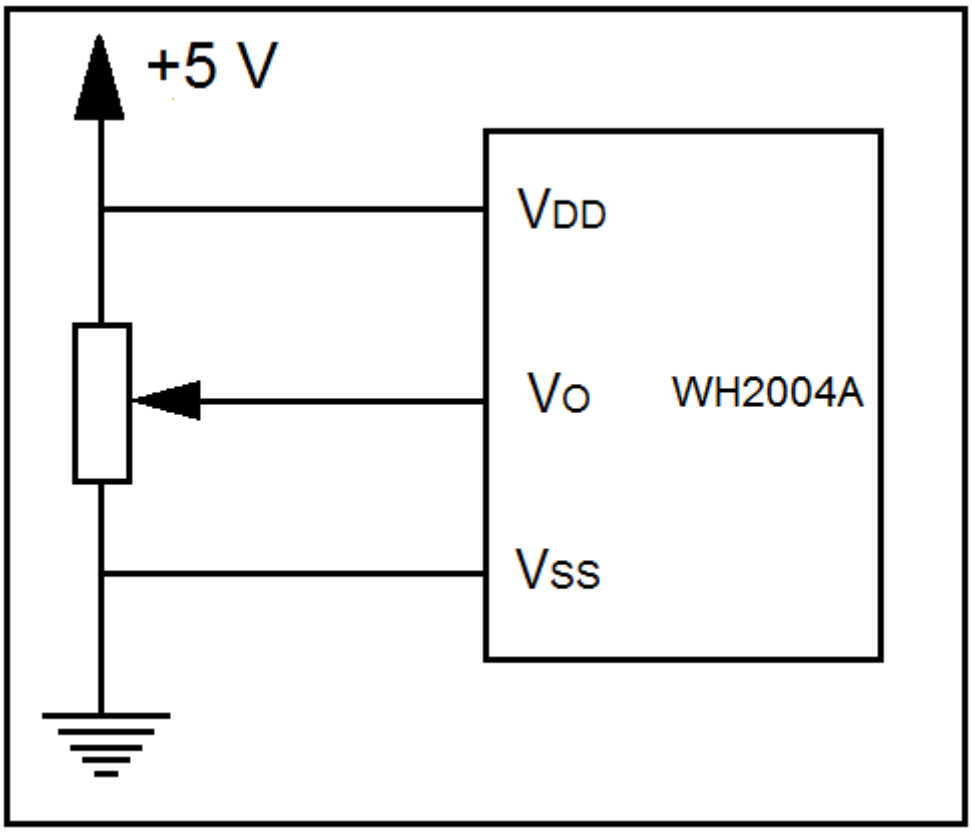
| Signaalilinja | Symboli | Kuvaus |
|---|---|---|
| **1** | VSS | Maadoitus |
| **2** | VDD | Virtalähde logiikkapiirille |
| **3** | V0 | Virtalähde LCD:lle |
| **4** | RS | Rekisterivalinta |
| **5** | R/W | Luku/Kirjoitus |
| **6** | E | Aktivointisignaali |
| **7** | DB0 | Databitti 0 |
| **8** | DB1 | Databitti 1 |
| **9** | DB2 | Databitti 2 |
| **10** | DB3 | Databitti 3 |
| **11** | DB4 | Databitti 4 |
| **12** | DB5 | Databitti 5 |
| **13** | DB6 | Databitti 6 |
| **14** | DB7 | Databitti 7 |
| **15** | A | Virtalähde LED:eille |
| **16** | K | Maadoitus LED:eille |

LCD-näyttö kytketään mikro-ohjaimeen yhden 8-bittisen portin välityksellä. Näyttöä ohjataan CodeVision AVR C-kääntäjän valmiiden kirjastofunktioiden avulla. Kirjastofunktioita käytettäessä LCD-näyttö kytketään mikro-ohjaimen PORTA:han taulukon 2 mukaisesti.

Taulukko 2. LCD-näytön kytkentä mikro-ohjaimen PORTA:han

| LCD-näytön signaalit | PORTA:n signaalilinjat |
|---|---|
| **RS(pin4)** | 0 |
| **R/W(pin5)** | 1 |
| **EN(pin6)** | 2 |
| | 3 |
| **DB4(pin11)** | 4 |
| **DB5(pin12)** | 5 |
| **DB6(pin13)** | 6 |
| **DB7(pin14)** | 7 |

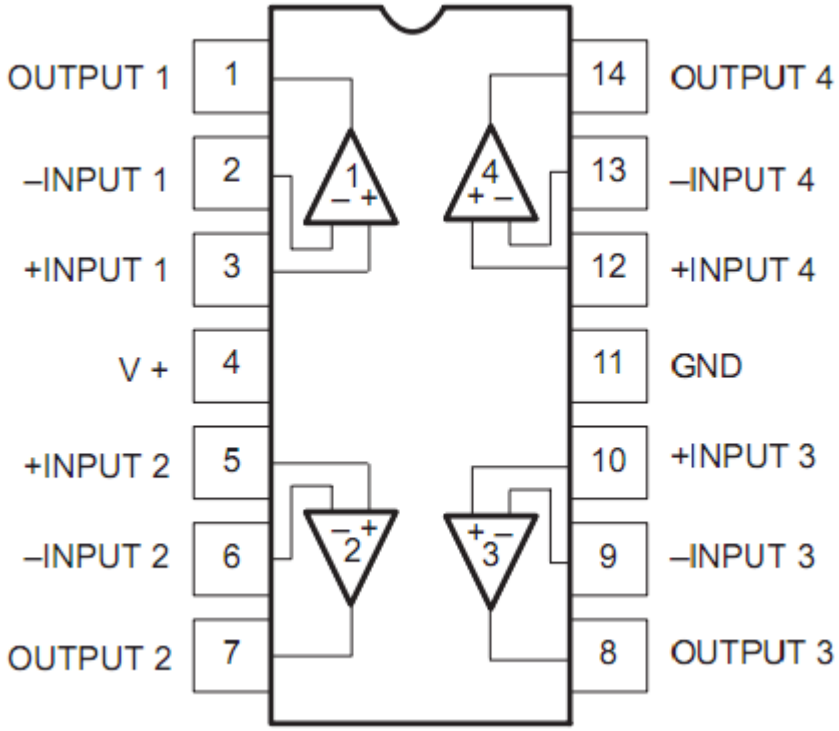
Näytön kontrastia saadaan säädettyä 10 kΩ:n potentiometrillä. Potentiometri liitetään näytön V0-linjaan, maadoituslinjaan sekä +5 V -linjaan. Kuvassa 4 on kuvattuna potentiometrin kytkentä LCD-näyttöön.




Kuva 4. Potentiometrin kytkentä LCD-näyttöön

## 4.4 Komparaattori

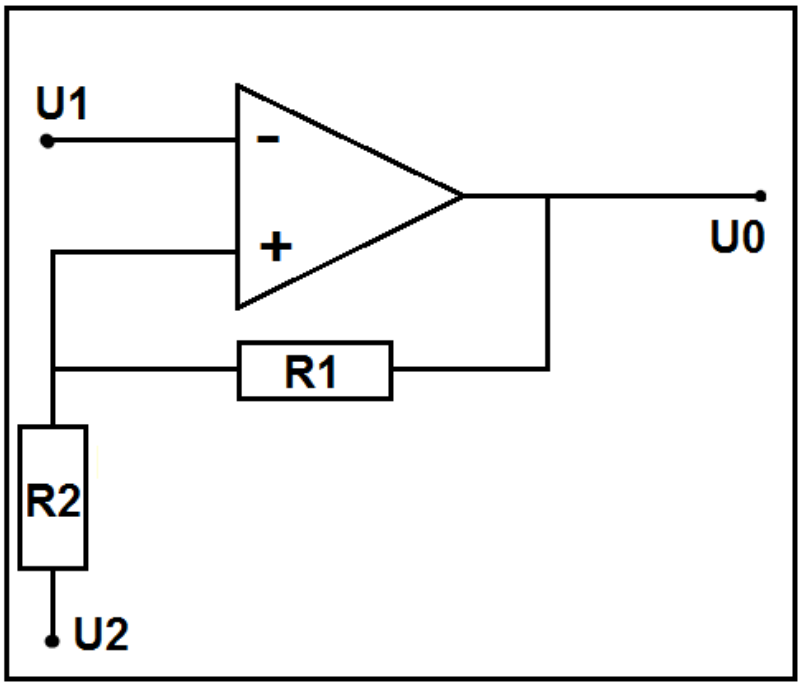
Ensimmäisellä laboratoriotestauskerralla kävi ilmi, että COLA-signaali oli liian heikko, jotta mikro-ohjain pystyisi havaitsemaan signaalin. Ongelmaa tutkiessa selvisi, että COLA-signaalin toiminta alue on 1 V nollan molemmin puolin, joka ei ole tarpeeksi vahva mikro-ohjaimelle, joka vaatii TTL-tason signaalin. Tästä johtuen valittiin LM324-operaatiovahvistin vahvistamaan signaalia. Signaalin vahvistamisen lisäksi se kytkettiin komparaattoriksi. Kytkentään lisättiin takaisinkytkentä, jolla saatiin aikaan hystereesi. Hystereesin ansiosta komparaattori ei reagoi pieniin häiriöihin tai kohinaan. Kuten kuvasta 5 ilmenee, LM324-piiri sisältää neljä vahvistinta, joista kuitenkin vain yhtä käytetään tässä työssä.




Kuva 5. LM324:n signaalilinjat /7/

Komparaattori vertailee kahden jännitteen tasoa toisiinsa. Ulostulona komparaattori antaa joko loogisen ykkösen tai nollan, sen mukaan kumman jännitelinjan jännite on suurempi. Koska tässä työssä ei olla tekemisissä kovin suurten taajuuksien kanssa, voidaan hyvin käyttää operaatiovahvistimella toteutettua komparaattoria. Mikäli toimittaisiin paljon suuremmilla taajuuksilla, olisi järkevämpää käyttää pelkästään komparaattorikäyttöön tarkoitettua komponenttia. /8/

Kun komparaattorista tehtiin takaisinkytkentä vertailujännitteeseen, saatiin aikaan hystereesi. Hystereesissä lähtöjännite vaikuttaa vertailujännitteeseen sen mukaan, kummassa tilassa lähtö on, jolloin komparaattorin tilojen tulkinta ei ole niin häiriöaltista. Kytkennässä käytettävien vastusten suuruuksilla on mahdollista määrittää hystereesin suuruus eli se, kuinka paljon jännitteen on noustava tai laskettava vertailuarvon yli ennen kuin komparaattori muuttaa tilaansa. Kuvassa 6 on yksinkertaistettu esimerkkikytkentä komparaattorin takaisinkytkennästä.

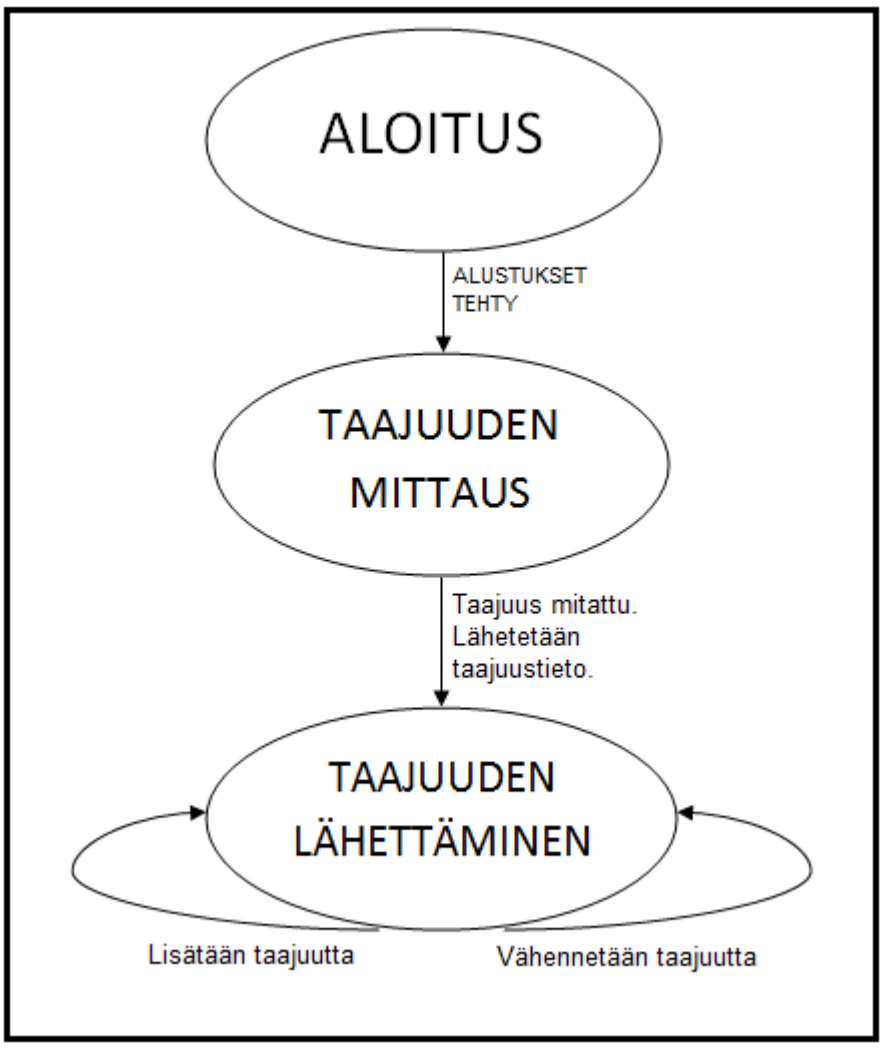



Kuva 6. Esimerkkikytkentä komparaattorin takaisinkytkennästä, jossa U1 on tulojännite, U2 on vertailujännite, U0 lähtöjännite sekä R1 ja R2 ovat käytettäviä vastuksia

# 5 OHJELMISTO

## 5.1 Ohjelmiston suunnittelu

Lähtökohtana ohjelman rakenteelle oli, että sen oli oltava mahdollisimman yksinkertainen ja suoraviivainen. Kuvasta 7 käy ilmi suunnitelma pelkistetystä ohjelmiston rakenteesta tilakaavion muodossa, josta ei kuitenkaan käy ilmi master- ja slave-ohjaimen erillisiä toimintoja vaan kokonaisuus laitteiston toiminnasta.




Kuva 7. Pelkistetty tilakaavio laitteiston kokonaistoiminnasta

Kun ohjelmiston toiminnot ja toiminnalliset tarpeet oli kartoitettu, alettiin tutkia mikro-ohjaimen ominaisuuksia erityisesti ajastimien ja liitäntäporttien kannalta. Sulautettujen järjestelmien prosessoreihin on haluttu tehdä paljon erilaisia toimintoja ilman, että prosessoripiirin pinnimäärä kasvaa suuremmaksi. Tästä johtuen prosessorin porttien (PORTA, PORTB, PORTC ja PORTD) pinnejä käytetään useampaan tarkoitukseen. Kaikkia portteja voidaan käyttää joko tuloina tai lähtöinä, minkä ansiosta prosessoriin saadaan kytkettyä esimerkiksi kahdeksan tuloa, joissa esiintyvät vain tasot 0 ja 1. Oleellista tämän työn kannalta oli kuitenkin se, että porttiin D on sijoitettu ajastimen ohjauksia, sarjaliikenteen tarvitsemat liitännät TXD ja RXD sekä kaksi ulkoisen keskeytyksen tuloa. /9/ Taulukosta 3 on nähtävissä PORTD:n vaihtoehtoiset toiminnot.

Taulukko 3. PORTD:n signaalilinjojen vaihtoehtoiset toiminnot /6/

| **Portin signaalilinja** | **Vaihtoehtoinen toiminto** |
|---|---|
| PD7 | OC2 (Timer/Counter2 Output Match Output) |
| PD6 | ICP1 (Timer/Counter1 Input Capture Pin) |
| PD5 | OC1A (Timer/Counter1 Output Compare A Match Output) |
| PD4 | OC1B (Timer/Counter1 Output Compare B Match Output) |
| PD3 | INT1 (External Interrupt 1 Input) |
| PD2 | INT0 (External Interrupt 0 Input) |
| PD1 | TXD (USART Output Pin) |
| PD0 | RXD (USART Input Pin) |

Ohjainten PORTD-linjoja käytettiin ajastimien ohjauksia ja keskeytyksiä varten, koska ne ovat PORTD:n vaihtoehtotoimintoja. Signaalilinja PD5 toimii slave-ohjaimessa OCR1A-ulostulona eli laitteesta lähtevän kanttisignaalin linjana. Master-ohjaimessa PD2 toimii INT0-keskeytyksen sisääntulona, joka käytännössä tarkoittaa LDS Dactronilta tulevan signaalin vastaanottoporttia. Slave-ohjaimessa PORTA valittiin LCD-näyttöä ohjaavaksi portiksi ja PORTC jätettiin kytkimien käyttöön. Sekä masterissa että slavessa PORTB:tä käytetään SPI-väylänä.

## 5.2 Ohjelmiston toteutus

Koska tärinätestausjärjestelmän monitorointilaitteella on kaksi päätehtävää: mitata tulevan siniaallon taajuus ja lähettää eteenpäin määrätyn taajuista kanttiaaltoa, jaettiin ohjelmisto näihin kahteen moduuliin kahdelle eri mikro-ohjaimelle. Kaksi mikro-ohjainta otettiin käyttöön, koska taajuuden mittaus ja lähetys toimivat ohjaimessa samoilla ajastinrekistereillä, minkä takia niitä ei voida ajaa yhtäaikaisesti.

Toiminnot on jaettu kahdelle mikro-ohjaimelle, joista toinen toimii masterina ja toinen slavena. Master-ohjaimen ohjelmisto perustuu tulevan signaalin taajuuden mittaamiseen ja tämän taajuustiedon lähettämiseen SPI-väylän kautta slave-ohjaimelle. Slave-ohjaimen tehtäviin kuuluu taajuustiedon vastaanottaminen masterilta SPI-väylän kautta, LCD-näytön ohjaus, taajuuden lähettäminen ja -poikkeutus. Tässä kappaleessa käydään läpi funktioiden toiminnot pääpiirteittäin, mutta ohjelmien C-

koodit ovat nähtävissä kokonaisuudessaan liitteissä 3 ja 4, joista löytyvät funktiot sekä muut tapahtumat kommentoituna yksityiskohtaisemmin.

### 5.2.1 SPI-tiedonsiirto

SPI on kaksisuuntainen synkroninen sarjasiirtoprotokolla, joka on alun perin Motorolan kehittämä. Kyseessä on erittäin laajalti käytetty menetelmä mikro-ohjaimissa ja niiden oheispiireissä. SPI:ssä ei käytetä kuittauksia eikä osoitteita, minkä takia se ei ole kaikkein luotettavin tiedonsiirtomenetelmä, mutta sopiva taajuustietojen lähettämiseen kahden mikro-ohjaimen välillä. /10/

SPI-tiedonsiirtomenetelmään päädyttiin, koska tarvittiin nopea ja yksinkertainen tiedonsiirtomenetelmä. Laiteessa siirretään tietoa sekunnin välein kahden ohjaimen välillä, minkä vuoksi tiedonsiirto ei saa olla liian hidas tai raskas, jotta ohjelmistojen muut toiminnot eivät keskeytyisi.

Kummassakin piirissä on kahdeksan bittinen siirtorekisteri, jonka kautta tieto siirretään ohjaimien välillä. Käytännössä tämä tarkoittaa sitä, että kun master-ohjain on saanut laskettua taajuuden, on sen muutettava tämä int-muotoinen muuttuja taulukkomuotoon. Kun taajuustieto on masterilla taulukkomuodossa, lähettää se tämän tiedon siirtorekisteriä käyttäen merkki kerrallaan slavelle, joka siirtää vastaanotetut tiedot omaan taulukkoonsa. Slave muuntaa vastaanotetun taulukkotiedon int-muuttujaksi, minkä jälkeen taulukko tyhjennetään seuraavaa vastaanottoa varten.

### 5.2.2 Taajuuden mittaus

Taajuuden laskeminen perustuu yksinkertaisesti siihen, että master-ohjelma rekisteröi ulkoisesti havaittavan signaalin laskevat reunat ja käy sekunnin välein laskemassa kuinka monta laskevaa reunaa ohjelma on rekisteröinyt. Tästä saadaan suoraan LDS Dactronilta tulevan sini-aallon taajuus.

Testauksessa kävi ilmi, että jos ohjelma käy laskemassa ulkoiset keskeytykset ainoastaan kerran sekunnin aikana, se antaa virheellisen taajuuden. Tästä johtuen ohjel-

maa muutettiin siten, että ohjelmassa käydään kahdensadan sadasosasekunnin välein laskemassa havaitut laskevat reunat, ja viiden jakson jälkeen, kun sekunti on kulunut, lähetetään taajuustieto eteenpäin.

Testauksen yhteydessä kokeiltiin myös tiheämpää lähetysväliä, joka perustui siihen, että esimerkiksi kahdensadan sadasosan aikana tulleet keskeytykset kerrotaan viidellä, jolloin saadaan teoreettinen laskelma sekunnin aikana tulleista keskeytyksistä. Tämäkään kokeilu ei kuitenkaan toiminut halutulla tavalla. Laskettu taajuus oli virhealtis, koska mikro-ohjaimen kello ja tuleva taajuus eivät ole synkronoitu, jolloin sekunnin eri jaksoilla saattoi tulla eri määrä keskeytyksiä ja tämä näkyi aika ajoin muutaman hertsin virheenä.

### 5.2.3 Sekunnin toteuttaminen ATmega32:lla

Sekunti toteutettiin mikro-ohjaimen vertailutilalla (Timer 1 Output Compare mode). Vertailutilassa verrataan TCNT1-ajastimen ja vertailurekisterin OCR1A:n arvoja. Kun ajastimen ja vertailurekisterin arvot ovat yhtä suuret, TIFR-lippurekisterin Output Compare Flag -lippu asettuu, ja ohjelma hyppää keskeytysfunktioon, mikäli Output Compare Interrupt -keskeytys on määritetty ja globaali keskeytys on sallittu. /11/

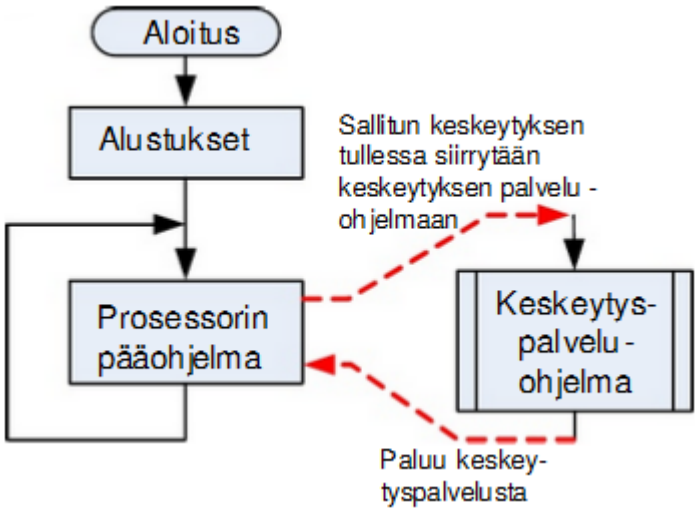
Tässä työssä käytimme mikro-ohjaimen sisäistä kello-oskillaattoria, jonka taajuus on 4 MHz. Tämä kellon taajuusarvo oli kuitenkin määritelty laitteen sisäisissä asetuksissa jaettavaksi neljällä, jolloin kellotaajuus oli käytännössä 1 MHz. Tämä tarkoittaa, että yhden sekunnin aikana kello vaihtaa tilaansa miljoona kertaa. TCCR1B-rekisteristä valittiin asetus, jossa käytettävän kellotahdin esijakajan arvo oli 64, jolloin yhden sekunnin aikana ohjelma rekisteröi 15625 kappaletta kellon tilan muutoksia. Kun tämä arvo vielä jaettiin viidellä, saatiin siitä kahtasataa sadasosasekuntia vastaava arvo. Kun OCR1A-rekisterin arvoksi määriteltiin 3125, ohjelma siirtyi kahdensadan sadasosan välein vertailutilakeskeytykseen ja kun keskeytyksessä oli käyty viisi kertaa, oli kulunut tasan sekunti.

Kun jokin keskeytys on käynnissä, ohjelman muut toiminnot eivät ole sallittuja, minkä takia keskeytysfunktio kannattaa pitää mahdollisimman yksinkertaisena. Tästä

johtuen ohjelman vertailutilakeskeytys toteutettiin niin, että keskeytyksen tapahtuessa askelletaan a-apumuuttujaa yhdellä, jonka jälkeen poistutaan keskeytyksestä. Kun apumuuttujan arvo on 5, ohjelma siirtyy mittaus-fuktioon, jossa lasketaan sekunnin aikana tulleet ulkoiset keskeytykset, jonka jälkeen tämä taajuustieto lähetetään eteenpäin slave-ohjaimelle, ja nollataan laskurit seuraavaa taajuustietoa varten.

### 5.2.4 Ulkoisten tapahtumien laskeminen

Ulkoisiin tapahtumiin reagoiminen on tehokkainta ulkoisella keskeytyksellä. Kun jokin huomiota kaipaava tapahtuma ilmoittaa mikro-ohjaimelle tarpeestaan niin sanotulla keskeytyspyynnöllä, ohjain keskeyttää sen hetkisen varsinaisen työnsä ja siirtyy keskeytysohjelmaan. /9/ Kuvasta 8 on nähtävissä, miten keskeytysmenetelmä toimii teoriassa.




Kuva 8. Mikro-ohjaimen keskeytysmenetelmä /8/

Ohjelma reagoi ulkoiseen tapahtumaan, joka tässä työssä käytännössä tarkoittaa LDS Dactronilta tulevaa signaalia, INT0-keskeytyksellä. MCUCR-rekisterissä asetettiin ohjelma reagoimaan signaalin laskeviin reunoihin, mikä tarkoittaa sitä, että ohjelma reagoi tulevan signaalin muutokseen. Kun ohjelma havaitsee INT0-keskeytyksen, se siirtyy keskeytysohjelmaan, jossa lisätään laskevan reunan laskuria yhdellä, minkä jälkeen ohjelma palaa pääohjelmaan.

### 5.2.5 Kanttiaallon lähettäminen

Kanttiaallon lähettämisessä käytettiin ajastimen vertailutilaa, kuten aikaisemmin tarkan sekunnin määrittämisessä, mutta hiukan toisella tavalla. Tässä toiminnossa käytettiin CTC:tä (Clear Timer on Compare Match Mode). CTC:ssä ohjain laskee normaalisti ylöspäin, kunnes ajastimen arvo on sama kuin OCR1A-verkkorekisteriin määritelty arvo. Tämän jälkeen ajastin nollataan ja laskenta alkaa alusta. Lisäksi määritellään asetetaanko lähtöpinni ykköseksi, nollaksi vai invertoidaanko se. Kyseessä olevan lähtöpinnin (PORTD:n 5-bitti) on asetettava ulostuloksi, mikä määritellään DDRD-suuntarekisterissä. /11/

Mikro-ohjaimesta ulos lähtevän signaalin taajuuden saa laskettua seuraavalla kaavalla:

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 * N * (1 + OCR_n)},$$

jossa $f_{OCn}$ on mikro-ohjaimesta lähtevän signaalin taajuus, $f_{clk\_I/O}$ on systeemin kellon taajuus, N on esijakajan arvo (1, 8, 64, 256 tai 1024) ja $OCR_n$ on OCR-verkkorekisterin (tässä tapauksessa OCR1A) arvo. /6/ Koska tiedetään haluttu taajuus, mutta ei tarvittavaa OCR1A-rekisterin arvoa, muutettiin kaava seuraavanlaiseksi:

$$OCR_n = (\frac{f_{clk\_I/O}}{2 * N * f_{OCn}}) - 1.$$

Kun tiedetään haluttu taajuus, sijoitetaan se lausekkeeseen $f_{OCn}$:n tilalle, jolloin saadaan mikro-ohjaimesta halutun taajuista signaalia.

### 5.2.6 Kytkimien toiminta

Slave-ohjaimella on kaksi kytkimillä toteutettua toimintoa. Kytkimet ovat nimeltään SW1 ja SW2. Molempien kytkintoimintojen toimivuus on varmistettu ohjelmallisesti toteutetulla kytkinvärähtelyn poistolla, jonka toiminta on käyty läpi tarkemmin kommentoidussa ohjelmakoodissa LIITE 4:ssa.

SW1:llä lähtötaajuuden poikkeuttajaa askelletaan suuremmaksi ja SW2:lla pienemmäksi. Kun esimerkiksi taajuutta askelletaan suuremmaksi, poikkeuttajan muuttujan arvoa kasvatetaan yhdellä, minkä jälkeen ohjelma palaa takaisin normaaliin toimintaansa.

# 6 PIIRILEVYN TOTEUTUS

## 6.1 Piirilevyn suunnittelu

Piirilevyn suunnittelu aloitettiin piirikaavion (LIITE 5) luonnilla, joka toteutettiin National Instrumentsin Multisim-ohjelmalla. Piirikaaviosta käy ilmi komponenttien kytkennät toisiinsa, mikä auttaa ymmärtämään laitteen toiminnan komponenttitasolla. Piirikaavion kytkennät eivät kuitenkaan kuvaa, miten komponentit on käytännössä kytketty piirilevylle.

Piirikaavion luonnin jälkeen käännettiin Multisimin avulla luotu kytkentä Ultiboard-ohjelmalle, jossa varsinainen piirilevyn suunnittelu tapahtui. Tässä ohjelmassa suunniteltiin komponenttien sijoittelu ja johdinvedot. Komponenttien ja johdinvetojen asettelujen jälkeen luotiin Ultiboardissa tarvittavat tuotantotiedostot, joita tässä työssä tarvittiin kolme: pohjan johdinvedot, poraukset ja piirilevyn reunat.

Seuraavaksi otettiin käyttöön CircuitCAM-ohjelma, jolla piirilevyn valmistustiedostot käsitellään lopullisesti jyrsinlaitetta varten. Ohjelmassa koottiin yksitellen tarvittavat tuotantotiedostot yhteen, ja luotiin niistä lopullinen valmistustiedosto jyrsinkoneen ohjaamista varten.

## 6.2 Piirilevyn valmistaminen

Lopullinen piirilevy jyrsittiin yksipuoliselle kuparipinnoitteiselle aihiolle. Aihio kiinnitettiin jyrsimeen, jota ohjattiin ohjelmallisesti tietokoneen kautta. Ohjelmalla avattiin piirilevyn valmistustiedosto, minkä jälkeen näytölle ilmestyi luonnos levystä,

jonka sai aseteltua halutulle kohdalle aihiota. Tämän jälkeen jyrsin alkoi porata reiät levylle erikokoisilla poranterillä, jotka vaihdettiin manuaalisesti porausten välissä.

Poraamisen jälkeen ohjelmassa siirryttiin johdinvetojen jyrsimiseen. Jyrsijään asennettiin jyrsimiseen tarkoitettu terä, jolla jyrsittiin kuparipinnoitteeseen johdinvedot sekä padit komponentteja varten. Tämän vaiheen jälkeen aihio käännettiin jyrsijässä, minkä jälkeen piirilevy jyrsittiin irti aihiosta.

Piirilevyn jyrsinnän jälkeen komponentit juotettiin piirilevyyn kiinni ja testattiin laitteen toiminta. Ensimmäinen prototyyppi piirilevystä oli toimiva, mutta sen jälkeen suunniteltiin ja jyrsittiin vähän kapeampi versio piirilevystä, jotta se mahtui sille suunniteltuun koteloon. Piirilevyn komponenttien asettelut ja johdinvedot on nähtävissä 3D-mallinnoksena liitteistä 6 ja 7.

# 7 LAITTEEN TESTAUS

## 7.1 Ohjelmiston testaus

Koko ohjelmiston kehittämisen aikana ohjelmistoa testattiin sitä mukaan, kun se kehittyi. Käytössä oli funktiogeneraattori ja oskilloskooppi, joilla simuloitiin laboratorion oikeita olosuhteita. Funktiogeneraattorilta saatiin tuotettua master-ohjaimelle halutunlaista signaalia, joka vastasi tärinätestausjärjestelmältä tulevaa signaalia. Oskilloskoopilla pystyttiin seuraamaan minkälaista signaalia slave-ohjain lähettää eteenpäin. Nämä laitteet olivat erityisen tärkeitä, kun tutkittiin miten laitteeseen tuleva ja laitteesta lähtevä signaali täsmäävät toisiinsa, jolloin oli helppo tutkia, millä taajuuksilla laite vaati ohjelmallista kalibrointia.

Yleisesti ohjelmiston toimivuutta testattiin yhtäjaksoisilla testeillä, jolloin laite jätettiin tulkitsemaan taajuutta pitkäksi aikaa. Tässä testissä tutkittiin, jääkö laite johonkin solmukohtaan jumiin, mutta tällaista tapahtumaa ei löydetty missään kohtaa testauksia. Toinen yleinen testausmenetelmä oli tulevan taajuuden vaihtelu eri nopeuksilla ja

sen vaikutuksen tutkiminen laitteen toimintaan. Näiden lisäksi testattiin luonnollisesti taajuusarvojen ääripäitä ja raja-arvoja.

### 7.2 Laitteiston testaus

Laitteiston testaus ei ollut niin laaja projekti, kuin ohjelmiston testaus, sillä laitteisto tuli suurelta osin testattua ohjelmiston testauksen yhteydessä. Laitteistoa testattiin sekä laboratoriossa että funktiogeneraattorin ja oskilloskoopin kanssa. Ensimmäisellä laboratoriotestauskerralla huomattiin COLA-signaalin olevan liian heikkoa, jotta TTL-tason komponentti pystyisi sitä tulkitsemaan, joten ennen kuin koko laboratoriotestaus sai alkunsa, laitteisto vaati jo lisäkomponentteja.

Funktiogeneraattorilla ja oskilloskoopilla testattiin lähinnä yksittäisten komponenttien toimivuutta. Kun ohjelmisto oli valmis, asennettiin kaikki komponentit koekytkentälevylle, joka simuloi lopullista piirilevyä. Koekytkentälevyä käytettiin, koska kytkennän ja komponenttien muuttaminen oli helppoa ja nopeaa, mikäli niille oli tarvetta. Kun laitteen toimivuus oli testattu koekytkentälevyllä sekä simuloidussa olosuhteissa että laboratoriossa, alettiin suunnitella lopullista piirilevyä.

## 8 YHTEENVETO

### 8.1 Lopullinen laite

Lopullinen prototyyppi koteloitiin läpinäkyvään siniseen pvc-muoviseen elektroniikkarasiaan. Läpinäkyvyys kotelolle oli tietoinen valinta, koska LCD-näyttö haluttiin suojaan laatikon sisälle, sillä sisällä näyttö oli suojassa ulkoisilta kolhuilta ja muutenkin suojaisemmassa asemassa kuin esimerkiksi upotettuna laatikosta ulos rakenteiden pinnalle. Laitteen oikealla reunalla sijaitsevat BNC-liittimet ja vasemmalla reunalla sijaitsee paikka virtajohdolle. Kuvassa 9 on nähtävissä lopullinen prototyyppi.


Kuva 9. Lopullinen prototyyppi

Monitorointilaitteen lopullisessa testauksessa todettiin, että laite toimi työn tilaajan toiveiden mukaisesti. Laite tulee helpottamaan sekä tehostamaan huomattavasti tärinätestauksessa resonanssipisteiden tutkimista ja näin ollen parantamaan tuotekehityksen laatua paljon.

## 8.2 Kehittämismahdollisuudet

Laite testattiin käytössä perusteellisesti ja se toimi toivotulla tavalla, mutta kehitettävää jäi kuitenkin jonkin verran, koska kyseessä on vain prototyyppi. Kehitettävää löytyy sekä ohjelmiston että laitteiston puolella.

Ohjelmiston puolella parannettavaa jäi master-ohjaimen taajuuden analysoinnissa. Erilaisella ajastintoiminnolla olisi mahdollisuus saada tarkemmin ja nopeammin laskettua tuleva taajuus. Sopivampi ajastintoiminto voisi olla esimerkiksi kaappaustila (capture mode), mutta tiukan aikataulun takia ei ollut mahdollista muokata ohjelmistoa kesken projektin.

Laitteiston puolella parannettavaa voisi olla esimerkiksi ulkoisten kristallikellokiteiden käyttöönotto, jolloin järjestelmän kellotaajuus olisi tarkempi. Piirilevyn kokoa olisi mahdollista pienentää tiheämmällä osa- ja johdinsijoittelulla, joilla ei kuitenkaan kovin suuria muutoksia voi saada aikaan. Sen sijaan erilaisella mikro-ohjaimen

pakkaustyypillä (TQFP/MLF) piirilevyn kokoa saisi huomattavasti pienemmäksi. Myös kokonaan eri mikro-ohjaimella (esimerkiksi ATmega88) olisi mahdollista pienentää levyn kokoa, mutta se vaatisi myös muutoksia ohjelmiston puolelle, mikä saattaa tehdä siitä hyvinkin hankalan operaation.

# LÄHTEET


1. SAMK. 2010. Rauman EMC-laboratorio. [Verkkodokumentti, viitattu 8.7.2011] Saatavissa: http://www.samk.fi/emc

2. Wikipedia. 2011. Sähkömagneettinen yhteensopivuus. [Verkkodokumentti, viitattu 8.7.2011] Saatavissa: http://fi.wikipedia.org/wiki/S%C3%A4hk%C3%B6magneettinen_yhteensopivuus

3. Matti Parkkonen. 1998. EMC. [Verkkodokumentti, viitattu 8.7.2011] Saatavissa: http://www.netlab.tkk.fi/opetus/s38118/s98/htyo/32/index.shtml

4. Jaakko Annanpalo. 1997. EMC-direktiivin. [Verkkodokumentti, viitattu 8.7.2011] Saatavissa: http://www.edilex.fi/tukes/fi/lainsaadanto/pdf/emc-direktiivin_soveltamisopas.pdf

5. Monarch Instrument. 2011. Monarch Palm Strobe. [Verkkodokumentti, viitattu 20.7.2011] Saatavissa: http://www.monarchinstrument.com/product.php?ID=32

6. ATMEL. 2011. ATmega32 datasheet. [Sähköinen dokumentti, viitattu 8.7.2011] Saatavissa: http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf

7. Philips. 2002. LM324 datasheet. [Sähköinen dokumentti, viitattu 28.9.2011] Saatavissa: http://www.datasheetcatalog.org/datasheets/208/62529_DS.pdf

8. Wikipedia. 2011. Komparaattori. [Verkkodokumentti, viitattu 28.9.2011] Saatavissa: http://fi.wikipedia.org/wiki/Komparaattori

9. Dingidom. 2010. AVR-rauta, keskeytysohjelmointia. [Verkkodokumentti, viitattu 18.7.2011] Saatavissa: http://www.scribd.com/doc/36922318/6-4-rauta-Keskeytys-Ohjelmointia

10. Dingidom. 2010. AVR-rauta, SPI-ohjelmointia. [Verkkodokumentti, viitattu 7.9.2011] Saatavissa: http://www.scribd.com/doc/36922321/6-5-AVR-rauta-SPI-ohjelmointia

11. Dingidom. 2010. AVR-rauta, ajastinohjelmointia. [Verkkodokumentti, viitattu 18.7.2011] Saatavissa: http://www.scribd.com/doc/36922351/6-6-1-AVR-rauta-Timer-ohjelmointia

# LIITTEET

## Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32
- Power Consumption at 1MHz, 3V, 25°C
  - Active: 1.1mA
  - Idle Mode: 0.35mA
  - Power-down Mode: < 1µA

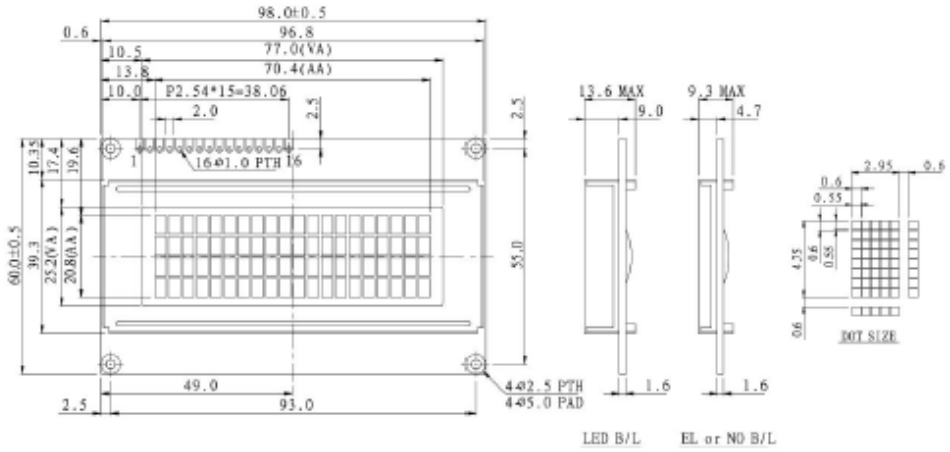
# 8-bit AVR® Microcontroller with 32KBytes In-System Programmable Flash

## ATmega32
## ATmega32L

2503Q–AVR–02/11

20x4dots Character **WH2004A**

## *Dimension drawing*




Character type

## *Feature*

1. 5x8 dots includes cursor
2. Built-in controller (KS 0066 or Equivalent)
3. +5V power supply (Also available for +3V)
4. 1/16 duty cycle
5. LED can be driven by pin1,pin2,pin15,pin16 or A and K
6. N.V. optional for +3V power supply

| Pin NO. | Symbol | Function |
|---|---|---|
| 1 | Vss | GND |
| 2 | Vdd | +3V or + 5V |
| 3 | Vo | Contrast Adjustment |
| 4 | RS | H/L Register select signal |
| 5 | R/W | H/L Read / write signal |
| 6 | E | H→L Enable signal |
| 7 | DB0 | H/L Data bus line |
| 8 | DB1 | H/L Data bus line |
| 9 | DB2 | H/L Data bus line |
| 10 | DB3 | H/L Data bus line |
| 11 | DB4 | H/L Data bus line |
| 12 | DB5 | H/L Data bus line |
| 13 | DB6 | H/L Data bus line |
| 14 | DB7 | H/L Data bus line |
| 15 | A/Vee | +4.2V for LED(RA=0Ω) /Negative Voltage output |
| 16 | K | Power supply for B/L (0V) |

## *Mechanical Data*

| Item | Standard Value | Unit |
|---|---|---|
| Module Dimension | 98.0x60.0 | mm |
| Viewing Area | 77.0x25.2 | mm |
| Mounting hole | 93.0x 55.0 | mm |
| Character Size | 2.95x4.75 | mm |

## *Absolute Maximum Rating*

| Item | Symbol | Standard Value | | | Unit |
|---|---|---|---|---|---|
| | | min. | typ. | max. | |
| Power Supply | VDD-VSS | -0.3 | --- | 7.0 | V |
| Input Voltage | VI | -0.3 | --- | VDD | V |

Note : VSS=0 Volt, VDD=5.0 Volt.

## *Electronical Characteristics*

| Item | Symbol | Condition | Standard Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | min. | typ. | max. | |
| Input Voltage | VDD | VDD=+5V | 4.7 | 5.0 | 5.3 | V |
| | | VDD=+3V | 2.7 | 3.0 | 5.3 | V |
| Supply Current | IDD | VDD=5V | --- | 1.0 | 1.2 | mA |
| Recommended LC Driving Voltage for Normal Temp. Version module | VDD-V0 | -20°C | 5.0 | 5.1 | 5.7 | V |
| | | 0°C | 4.6 | 4.8 | 5.2 | |
| | | 25°C | 4.1 | 4.6 | 4.7 | |
| | | 50°C | 3.9 | 4.2 | 4.5 | |
| | | 70°C | 3.7 | 3.9 | 4.3 | |
| LED Forward Voltage | VF | 25°C | --- | 4.2 | 4.6 | V |
| LED Forward Current | IF | 25°C | --- | 280 | 560 | mA |
| EL Power Supply Current | IEL | Vel=110VAC;400Hz | --- | --- | 5.0 | mA |

## *Display Character Address Code :*

| Display position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | --- | --- | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DD RAM Address | 00 | 01 | | | | | | | | | | | | | | 13 |
| DD RAM Address | 40 | 41 | | | | | | | | | | | | | | 53 |
| DD RAM Address | 14 | 15 | | | | | | | | | | | | | | 27 |
| DD RAM Address | 54 | 55 | | | | | | | | | | | | | | 67 |

```
 /*****************************************************************************************
Project:       Monitorointilaite tärinätestausjärjestelmälle
File:          Master-ohjelma
Date:          12.7.2011
Author:        Jussi Takala
Chip type:     ATmega32
*****************************************************************************************/


/*****************************************************************************************
Käytettävät kirjastot
*****************************************************************************************/
#include <mega32a.h>
#include <delay.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>


/*****************************************************************************************
Funktioiden alustukset
*****************************************************************************************/
void SPI_MasterInit(void);
void SPI_MasterTransmit(unsigned char data);
void syotto(void);
void kalibrointi(void);


/*****************************************************************************************
Muuttujien alustukset
*****************************************************************************************/
unsigned char i=0;                    //apumuuttuja SPI-lähetyksessä
unsigned char a=0;                    //apumuuttuja vertailukeskeytyksessä
unsigned char kalibrointikerta=0;     //kalibrointikertojen muuttuja
unsigned char text[];                 //taulukko, jonka sisältö siirretään SPI:llä
unsigned long laskeva_reuna=0;        //laskevien reunojen laskuri
unsigned long taajuusarvo=0;          //taajuusarvo jota laite lähettää eteenpäin


/*****************************************************************************************
Vertailutila-keskeytys

Pääohjelmassa on annettu ohjausrekisteri-OCR1A:lle arvo 3125, joka vastaa
kahtasataa sadasosaa yhden megahertsin kellossa, kun kellotahdin esijakajan jakoluku on 64
(1MHz -> 1000000Hz/64clk/5=3125). Näin ollen ohjelma käy kahdensadan sadasosan välein tässä
keskeytyksessä, jolloin se kasvattaa a-muuttujan arvoa yhdellä.
*****************************************************************************************/
interrupt [TIM1_COMPA] timer1_compA(void) //timerin keskeytysfunktio
{
    a++;
}


/*****************************************************************************************
Ulkoinen INT0-keskeytys

Ohjelma askelluttaa laskeva_reuna-laskuria yhdellä aina, kun se havaitsee laskevan reunan
PD2-pinnissä.
*****************************************************************************************/
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    laskeva_reuna++;
}


/*****************************************************************************************
Mittaus-funktio

Mittaus-funktio, jossa käydään sekunnin välein. Tässä funktiossa
lasketaan sekunnin aikana kertyneet laskevat reunat, joista saa suoraan arvon hertseinä(Hz=1/s).
*****************************************************************************************/

void mittaus(void)
```

```
{
        taajuusarvo=(laskeva_reuna);  //määritellään taajuus laskevista reunoista

        kalibrointi();                //kutsutaan kalibrointi-funktiota

        itoa(taajuusarvo,text);       //siirretään taajuustieto text-taulukkoon

        syotto();                     //kutsutaan syotto-funktiota

        laskeva_reuna = 0;            //nollataan laskurit
        a=0;

}


/*****************************************************************************************
SPI_MasterInit-funktio

Funktio, joka sisältää master-ohjaimen alustukset SPI-kommunikointia varten.
*****************************************************************************************/
void SPI_MasterInit(void)
{
    DDRB = (1<<5) | (1<<7) | (1<<4);  //MOSI, SCK & SS out
    DDRB &= ~(1<<6);                  //MISO in

    //SPI:n aktivointi, master, esijakaja fck/16 & LSB ensin
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0) | (1<<DORD);
}


/*****************************************************************************************
SPI_MasterTransmit-funktio

Merkkien lähetys SPI:n kautta slavelle.
*****************************************************************************************/
void SPI_MasterTransmit(unsigned char data)
{
    PORTB &= ~(1<<4);                 //SS alas
    SPDR = data;                      //lähetettävän datan muuttuja
    while (!(SPSR & (1<<SPIF)));      //odotetaan kunnes lähetys on valmis

}


/*****************************************************************************************
Syotto-funktio

Tässä funktiossa lähetetään taajuustieto slave-ohjaimelle. Slavelle lähetetään joka kerta
kolme merkkiä pitkä muuttuja riippumatta taajuuden suuruudesta, mistä johtuen 999 Hz on suurin
taajuus joka voidaan lähettää eteenpäin.
*****************************************************************************************/
void syotto(void)
{
    #asm("cli")                       //keskeytyksiä ei sallita
    for(i=0; i<3; i++)                //lähetetään kolme merkkiä
        {
            SPI_MasterTransmit(text[i]);//kutsutaan SPI_MasterTransmit-funktiota

            delay_ms(1);              //pieni viive, jotta ohjain pysyy mukana lähetyksessä
        }
     #asm("sei")                      //sallitaan keskeytykset

}


/*****************************************************************************************
Kalibrointi-funktio

Laitteen testauksessa kävi ilmi, että laskettavat taajuudet alkavat jätättämään tiettyjen
arvojen jälkeen, joten tällä funktiolla ne taajuus arvot kalibroidaan ohjelmallisesti.
Kalibrointikerta-muuttujalla varmistetaan, että kalibrointi tapahtuu vain kerran eikä kahta
kertaa, joka olisi muuten mahdollista kalibrointien ääriarvoilla.
*****************************************************************************************/

void kalibrointi(void)
{
    kalibrointikerta=0;
```

```
    if (taajuusarvo<=69)
        {
            taajuusarvo=taajuusarvo;
            kalibrointikerta=1;
        }

    if((taajuusarvo>69) && (taajuusarvo<=172) && (kalibrointikerta==0))
        {
            (taajuusarvo=taajuusarvo+1);
            kalibrointikerta=1;
        }

    if((taajuusarvo>172) && (taajuusarvo<=218) && (kalibrointikerta==0))
        {
            (taajuusarvo=taajuusarvo+2);
            kalibrointikerta=1;
        }

    if((taajuusarvo>218) && (taajuusarvo<=282) && (kalibrointikerta==0))
        {
            (taajuusarvo=taajuusarvo+3);
            kalibrointikerta=1;
        }

    if((taajuusarvo>282) && (taajuusarvo<=319) && (kalibrointikerta==0))
        {
            (taajuusarvo=taajuusarvo+4);
            kalibrointikerta=1;
        }

    if (taajuusarvo>319 && (kalibrointikerta==0))
        {
            (taajuusarvo=taajuusarvo+5);
            kalibrointikerta=1;
        }

}


/*****************************************************************************************
Pääohjelma
*****************************************************************************************/
void main (void)
{
    SPI_MasterInit();                //kutsutaan SPI-MasterInit-fuktiota

    #asm("sei")                      //sallitaan globaalit keskeytykset

    //INT0-asetukset
    GICR  |= (1 << INT0);                 //keskeytys sallittu
    MCUCR |= (1 << ISC01) | (0 << ISC00); //laskevalla reunalla keskeytys

    TCCR1B |= (1 << CS10) |(1 << CS11) | (1<<WGM12) ;
    //CS10 & CS11 -> järjestelmän kello/64, WGM12 -> CTC, clear timer on compare match


    while(1)
    {

        OCR1A = 3125;                //kahdensadan sadasosan OCR1A-arvo (1MHz/64clk)

        TCCR1A = 0x00;               //timerin normaali toiminta
        TIMSK |= 1 << OCIE1A;        //timerin vertailutila sallittu

        if(a==5)
        {
            mittaus();               //kutsutaan mittaus-funktiota
        }



    }



}
```

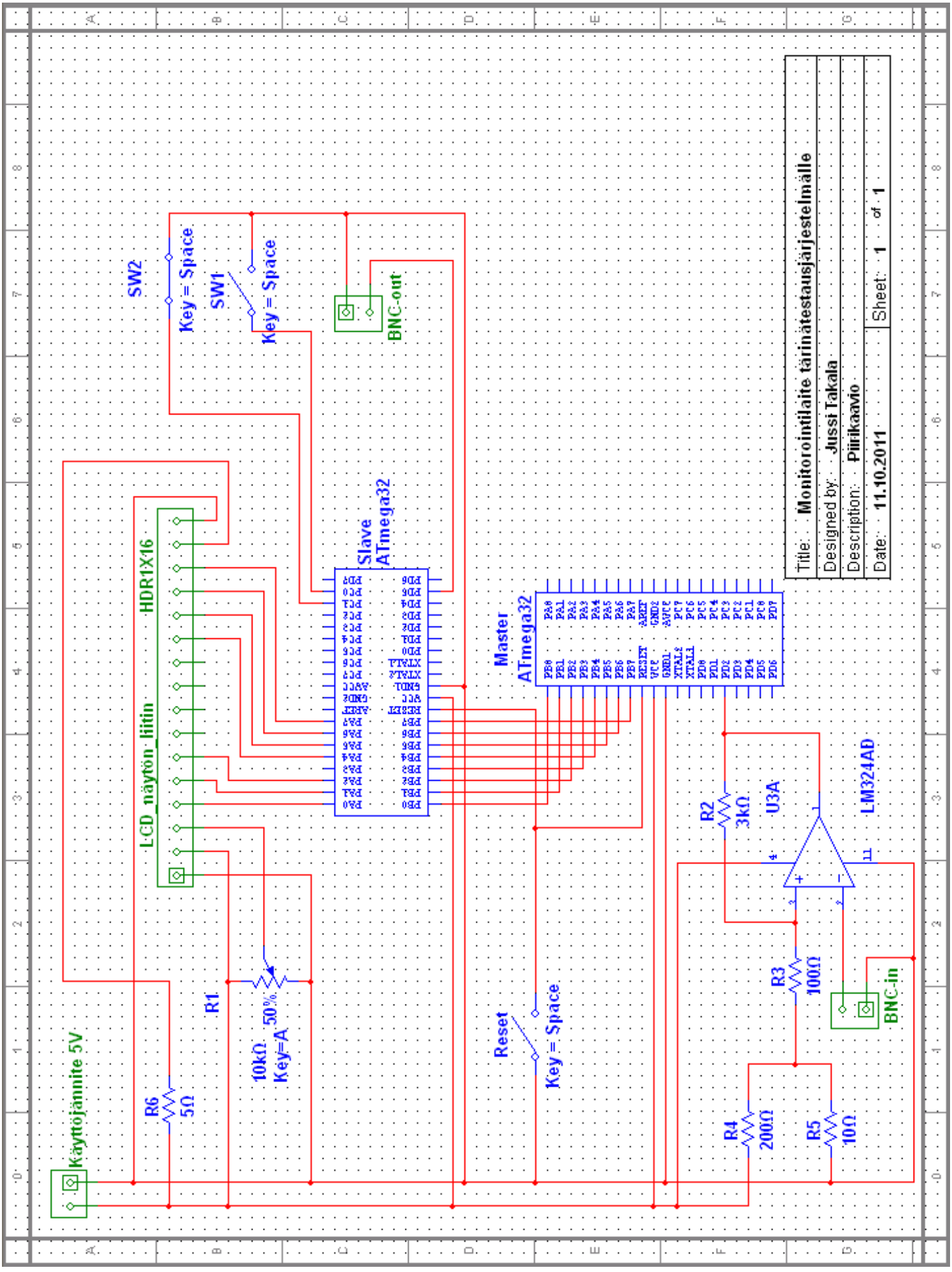
```
/*****************************************************************************************
Project:        Monitorointilaite tärinätestausjärjestelmälle
File:           Slave-ohjelma
Date:           12.7.2011
Author:         Jussi Takala
Chip type:      ATmega32
*****************************************************************************************/


/*****************************************************************************************
Käytettävät kirjastot
*****************************************************************************************/
#include <mega32a.h>
#include <delay.h>
#include <math.h>
#include <lcd.h>
#include <stdio.h>
#include <stdlib.h>

#asm
   .equ __lcd_port=0x1b;
#endasm

/*****************************************************************************************
Vakioiden määrittelyt
*****************************************************************************************/
#define RIVI 0                          //LCD-näytön riviluku
#define SARAKE 0                        //LCD-näytön sarakeluku

#define SW1         !PINC.0             //pluskytkin
#define SW2         !PINC.1             //miinuskytkin

#define ANYSW       (SW1 || SW2)        //kumpi tahansa kytkin SW1-SW2

#define KV_MAX      5000                //kytkinvärähtelyn maksimiarvo

/*****************************************************************************************
Muuttujien määrittelyt
*****************************************************************************************/
unsigned char datas = 0x00;             //SPI:llä vastaanotettava merkki
unsigned char datajono [];              //taulukko johon siirretään datas-muuttujasta tieto
unsigned char kalibrointikerta=0;       //kalibrointikertojen muuttuja
int datavalmis =0;                      //lopullinen datatieto
int datalippu=0;                        //SPI-siirron merkkilippu
int pois=0;                             //apumuuttuja datasiirtoon
int tallennuslaskuri=0;                 //vastaanotettujen merkkien laskurimuuttuja
unsigned char puskuri [21];             //LCD-näytölle puskuroitava tieto
int muutos=2;                           //poikkeutusmuuttuja

/*****************************************************************************************
Funktioiden alustukset
*****************************************************************************************/
void SPI_SlaveInit(void);
void kv_pois(void);
void datatallennus(void);
void datasiirto(void);
void kalibrointi(void);


/*****************************************************************************************
SPI_SlaveInit-funktio

Funktio, joka sisältää slave-ohjaimen alustukset SPI-kommunikointia varten.
*****************************************************************************************/
void SPI_SlaveInit(void)
{
    DDRB = (1<<6);      //PB6=MISO--->output

    //SPI:n aktivointi, SPI-keskeytyksen aktivointi, esijakaja fck/16 & LSB ensin
    SPCR |= (1<<SPE);
    SPCR |= 1<<SPIE;
    SPCR |= 1<<SPR0;
    SPCR |= 1<<DORD;
```

```
}


/*****************************************************************************************
SPI-keskeytyksen vastaanotto
*****************************************************************************************/
interrupt  [SPI_STC] void spi_isr(void)
{
    datalippu=1;

}


/*****************************************************************************************
Kytkinvärähtelyn poistofunktio

Tässä funktiossa on poistettu ohjelmallisesti kytkinvärähtely, jotta kytkintä painettaessa
ei syntyisi värähtelyä, jota laite saattaa tulkita monena panaluksena. Lisäksi tällä
varmistetaan esimerkiksi se, että yhdellä kytkimen painalluksella taajuutta poikkeutetaan
ainoastaan yhdellä hertsillä.
*****************************************************************************************/
void kv_pois(void)
{
  int kv_on =KV_MAX;                  //kytkinvärähtelyn poiston muuttuja

  while (kv_on)                       //Ohjelma on tässä silmukassa niin kauan, kun
  {                                   //kv_on on jokin muu arvo kuin nolla (0)
    if (ANYSW)
      kv_on = KV_MAX;                 //kv_on-muuttuja saa arvokseen 5000(KV_MAX) niin kauan,
    else                              //kun jokin kytkimistä on painettuna alas
      kv_on--;                         //kun mikään kytkimistä ei ole painettuna, ohjelma
                                      //askeltaa kv_on-arvoa alaspäin ja kun se saavuttaa
  }                                  //nolla-arvon, poistutaan ohjelmasta
}


/*****************************************************************************************
Datasiirto-funktio

Kun SPI-keskeytys nostaa datalipun ylös, siirrytään tähän funktioon, jossa vastaanotetaan
masterilta merkit (3kpl) ja siirretään ne omaan taulukkoon.
*****************************************************************************************/
void datasiirto(void)
{
        static unsigned char i=0;     //alustetaan i-muuttuja, joka määrää taulukkopaikan
                                      //kun vastaanotettu merkki siirretään taulukkoon.
         if(tallennuslaskuri==3)      //jos kolme merkkiä on vastaanotettu
        {
            datatallennus();          //kutsutaan datatallennusmuuttujaa
            delay_ms(50);             //viive, jotta ohjain pysyy mukana

            datajono[0] = '\0';       //alustetaan datajonotaulukko
            datajono[1] = '\0';
            datajono[2] = '\0';
            datajono[3] = '\0';

            i=0;                      //nollataan i-muuttuja seuraavaa siirtoa varten
            tallennuslaskuri=0;       // nollataan tallennuslaskuri seuraavaa siirtoa varten
            pois=1;                   //pois muuttujalle arvo 1, eli poistutaan ohjelmasta
        }

        if (pois==0)                  //jos pois-muuttuja on nolla, vastaanotetaan dataa
        {
            datas = SPDR;             //vastaanotetaan merkki
            datajono[i] = datas;      //siirretään merkki datajono-taulukkoon
            i++;                      //kasvatetaan taulukkopaikan numeroa yhdellä
            tallennuslaskuri++;       //kasvatetaan tallennuslaskuria yhdellä

        }
        datalippu=0;                  //nollataan datalippu seuraavaa SPI-keskeytystä varten


}
```

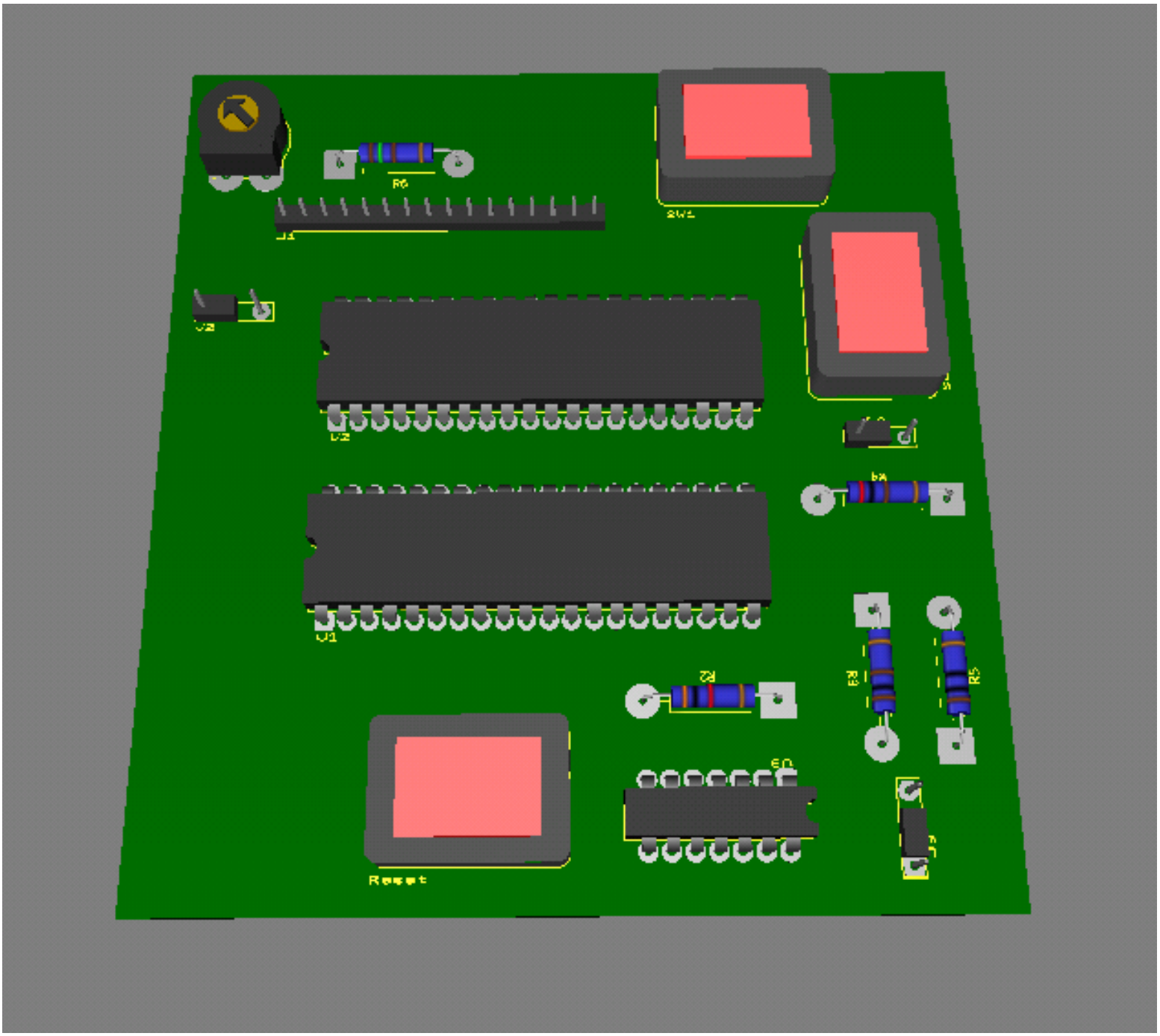
```
/*********************************************************************************************
Datatallennus-funktio

Tässä funktiossa tulostetaan näytölle laitteeseen tuleva taajuus ja taajuuden poikkeutusarvo
*********************************************************************************************/
void datatallennus(void)
{
        datavalmis = atoi(datajono);  //siirretään taulukkomuotoinen taajuustieto
                                      //int-muotoiseen datavalmis-muuttujaan
        kalibrointi();                //kutsutaan kalibrointi-funktiota

        lcd_gotoxy(SARAKE, RIVI);     //määritellään mihin kohtaan näyttöä tulostetaan
        sprintf(puskuri,"Taajuus: %d Hz   ",datavalmis); //siirretään puskurille tieto
        lcd_puts(puskuri);            //tulostetaan tieto näytölle

        if(muutos<0)
        {
            lcd_gotoxy(SARAKE, RIVI+2);      //määritellään mihin kohtaan näyttöä tulostetaan
            sprintf(puskuri,"Poikkeutus: %d  ",muutos); //siirretään puskurille tieto
            lcd_puts(puskuri);        //tulostetaan tieto näytölle
        }
        else
        {
            lcd_gotoxy(SARAKE, RIVI+2);      //määritellään mihin kohtaan näyttöä tulostetaan
            sprintf(puskuri,"Poikkeutus: +%d ",muutos); //siirretään puskurille tieto
            lcd_puts(puskuri);        //tulostetaan tieto näytölle
        }


}


/*********************************************************************************************
Kalibrointi-funktio

Laitteen testauksessa kävi ilmi, että laskettavat taajuudet alkavat jätättämään tiettyjen
arvojen jälkeen, joten tällä funktiolla ne taajuus arvot kalibroidaan ohjelmallisesti.
Kalibrointikerta-muuttujalla varmistetaan, että kalibrointi tapahtuu vain kerran eikä kahta
kertaa, joka olisi muuten mahdollista kalibrointien ääriarvoilla.
*********************************************************************************************/
void kalibrointi(void)
{
    kalibrointikerta=0;

    if(datavalmis<=39)
    {
        OCR1A = (1000000/(2*8*(datavalmis+muutos))-1);
        kalibrointikerta=1;
    }

    if((datavalmis>39) && (datavalmis<=120) && (kalibrointikerta==0))
    {
        OCR1A = (1000000/(2*8*(datavalmis+1+muutos))-1);
        kalibrointikerta=1;
    }

    if((datavalmis>120) && (datavalmis<=219) && (kalibrointikerta==0))
    {
        OCR1A = (1000000/(2*8*(datavalmis+2+muutos))-1);
        kalibrointikerta=1;
    }

    if ((datavalmis>219) && (kalibrointikerta==0))
    {
        OCR1A = (1000000/(2*8*(datavalmis+3+muutos))-1);
        kalibrointikerta=1;
    }

}
```

```
/*****************************************************************************************
Pääohjelma
*****************************************************************************************/
void main (void)
{
    DDRD |= (1<<5);                  //PD5 = OCR1A output

    TCCR1A |= (1 << COM1A0);         // OC1A-pin toggles on compare match
    TCCR1B |= (1 << WGM12);          // CTC, Clear Timer on Compare match
    TCCR1B |= (1 << CS11);           //jakoluku 8

    DDRC = 0x00;
    PORTC = 0xFF;

    lcd_init(20);                    //näytön alustus
    SPI_SlaveInit();                 //kutsutaan SPI_SlaveInit-funktiota
    #asm("sei")                      //sallitaan keskeytykset



    while(1)
    {

        if(datalippu==1)             //jos datalippu on 1
        {
            datasiirto();            //kutsutaan datasiirto-funktiota
            pois=0;                  //nollataan pois-muuttuja
        }

       if(SW1)                       //Jos SW1 on painettu
        {
            muutos++;                //kasvatetaan muutos-muuttujaa yhdellä

            kv_pois();               //kutsutaan kv_pois-muuttujaa
        }
        if(SW2)                      //Jos SW1 on painettu
        {
            muutos--;                //vähennetään muutos-muuttujaa yhdellä

            kv_pois();               //kutsutaan kv_pois-muuttujaa
        }

    }


}
```

LIITE 5


Käyttöjännite 5V
R6
5Ω
R1
10kΩ
Key=A
50%
LCD_näytön_liitin
HDR1X16
SW2
Key = Space
SW1
Key = Space
BNC-out
Slave
ATmega32
Master
ATmega32
Reset
Key = Space
R2
3kΩ
U3A
LM324AD
R3
100Ω
BNC-in
R4
200Ω
R5
10Ω
Title: Monitorointilaite tärinätestausjärjestelmälle
Designed by: Jussi Takala
Description: Piirikaavio
Date: 11.10.2011
Sheet: 1 of 1

LIITE 6



LIITE 7