

Opinnäytetyö (AMK)

Koulutusohjelma

Sulautetut järjestelmät

2010

Vadim Alexanov

Tasajännitetehton mittaus mikrokontrollerikortilla



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut järjestelmät

Valmistumisajankohta: Joulukuu 2010 | 27 sivua + 13 liitesivua

Ohjaaja: Kari Määttänen

Vadim Alexanov

Tasajännitetehten mittaaminen mikrokontrollerikortilla

Tässä opinnäytetyössä käsitellään alle 300 W:n tasajännitetehten mittaamista 12 V:n ja 24 V:n sähköjärjestelmissä.

Tehon mittaaminen toteutetaan suunnitellun mikrokontrollerikortin avulla. Kortin ytimenä toimii AVR ATmega8 -mikrokontrolleri, joka mittaa tehoa perinteisellä menetelmällä kertomalla virran ja jännitteen arvot keskenään. Mikrokontrolleri käyttää sisäänrakennettua 10-bittistä AD-muunninta. Virta mitataan Hall-ilmioon perustuvalla anturilla.

Kortti voi siirtää mitatut tehot USB-väylän kautta tietokoneeseen käsiteltäväksi. Tämän lisäksi kortille voi kytkeä LCD-näyttöä.

Ohjelmisto toteutetaan C-kielillä käyttäen GNU-pohjaista WinAVR-C-käännintä AVR Studion kehitysympäristössä.

Suunniteltu prototyyppikortti pystyy mittaamaan tehoa noin 1,5 %:n tarkkuudella. Työssä käsitellään mahdollista laitteisto- ja ohjelmistojatkokehitystä laitteen tarkkuuden ja käytettävyyden parantamiseksi.

ASIASANAT:

AVR-mikrokontrolleri, AD-muunnin, tasajännitetehten mittaaminen

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Systems

December 2010 | 27 pages + 13 appendices

Instructor: Kari Määttänen

Vadim Alexanov

DC Power Measuring with Microcontroller Card

This thesis discusses up to 300 watts DC power measurement in 12 V and 24 V electrical systems.

Power Measurement is implemented by using a designed microcontroller card. The core of this card is the AVR microcontroller ATMEGA8, which uses generic method by multiplying the current and voltage with each other for DC power measuring. Microcontroller uses built-in 10-bit AD converter. DC current is measured with Hall-effect-based sensor.

The measured values of DC power can be transferred from the card to computer via the USB port to process the data. In addition, LCD display can be connected to the card to show measured values.

Software is implemented in C language using GNU-based WinAVR C-compiler in AVR Studio environment.

The designed microcontroller card is able to measure DC power with about 1,5 % accuracy. Possible further software and hardware development to improve device accuracy and usability are also considered in this thesis.

KEYWORDS: AVR microcontroller, AD converter, DC power measuring

SISÄLTÖ

| | |
|---|-----------|
| 1 JOHDANTO | 7 |
| 1.1 Käyttökohteet | 7 |
| 1.2 Tarvemäärittely | 7 |
| 2 PERUSOSIEN VALINTA | 8 |
| 2.1 Mikrokontrolleri | 8 |
| 2.2 Anturit | 8 |
| 2.2.1 Jännite-anturi | 9 |
| 2.2.2 Virta-anturi | 9 |
| 2.3 Muut osat | 10 |
| 3 KORTIN SUUNNITTELU | 10 |
| 3.1 Laitteiston suunnittelu | 10 |
| 3.1.1 Virta-anturin esivahvistin | 11 |
| 3.1.2 Mitattavan jännitteen jakaja | 12 |
| 3.1.3 Mikrokontrolleri | 13 |
| 3.1.4 USB-muunnin | 13 |
| 3.1.5 LCD-näyttö | 13 |
| 3.1.6 EAGLE-ohjelmisto | 14 |
| 3.2 Koodin kehitys | 15 |
| 3.2.1 Kehitysympäristö | 15 |
| 3.2.2 Muut apuohjelmat | 16 |
| 3.2.3 Mikro-ohjaimen alustus | 18 |
| 3.2.4 Pääkoodisilmukka | 19 |
| 3.2.5 LCD-koodi | 21 |
| 4 SAAVUTETUT TULOKSET | 21 |
| 5 MAHDOLLINEN KORTIN JATKOKEHITYS | 24 |
| 5.1 Laitteiston jatkokehitys | 24 |
| 5.2 Koodin jatkokehitys | 24 |
| 6 YHTEENVETO | 25 |
| LÄHTEET | 27 |
| | |
| KUVAT | |
| Kuva 1. LEM LTS-25 -virta-anturi. | 10 |
| Kuva 2. Kortin lohkokaavio. | 11 |
| Kuva 3. Jännitejakajan kytkentäkaavio. | 12 |
| Kuva 4. Hitachi 44780 näytön lohkokaavio. | 14 |
| Kuva 5. Prototyypin piirilevy. | 15 |

| | |
|---|----|
| Kuva 6. AVR Studion simulaattori. | 16 |
| Kuva 7. Hapsim-simulaattorin ikkuna. | 17 |
| Kuva 8. Terminal v1.9 -ohjelman ikkuna. | 18 |
| Kuva 9. Pääsilmmukan vuokaavio. | 19 |
| Kuva 10. LTS-25-lähtöjännite. | 22 |
| Kuva 11. Virta-anturin emuloiva vastusjakaja. | 22 |

TAULUKOT

| | |
|--|----|
| Taulukko 1. Prototyypin testaustulokset. | 23 |
|--|----|

LIITTEET

| | |
|---|----|
| Liite 1a. Kortin kytkentäkaavio. | 28 |
| Liite 1b. LCD:n kytkentäkaavio. | 29 |
| Liite 2a. Lähdekoodi <i>tehomittari.c</i> . | 30 |
| Liite 2b. Lähdekoodi <i>lcd.c</i> . | 37 |
| Liite 3a. Prototyypikortin kuva 1. | 41 |
| Liite 3b. Prototyypikortin kuva 2. | 42 |

LYHENTEET

| | |
|------------|--|
| AD | Analog-Digital-muunnos |
| ADC | AD-muunnin eli analogia-digitaalimuunnin, Analog-to-Digital Converter |
| ASCII | 128 merkkipaikan laajuinen tietokoneiden merkistö, American Standard Code for Information Interchange |
| COM-portti | tietokoneen sarjaportti, Communication port RS-232 |
| DC | tasavirta, Direct Current |
| IDE | integroitu ohjelmiston kehitysympäristö, Integrated Development Environment |
| LCD | nestekidenäyttö, Liquid Crystal Display |
| USART | mikrokontrollerin sarjaporttiyksikkö, Universal Synchronous and Asynchronous serial Receiver and Transmitter |
| USB | tietokoneen laajennusväylä, Universal Serial Bus |

1 Johdanto

1.1 Käyttökohteet

Nykyään sähkön hinta nousee koko ajan. Tämän takia kehitetään nopeasti uusiutuvia energialähteitä kuten esim. tuulivoimaa ja aurinkokennojärjestelmiä. Pientaloissa ja kesämökillä sähkön kulutus on suhteellisen pieni, joten myös niissä voi käyttää tuuli- ja aurinkoenergiaa. Suurin osa näistä järjestelmistä käyttää akkuja sähköenergian varastointia varten. Akkujen kapasiteetti on rajoitettu, minkä takia on tärkeää optimoida akkujen sähkön varastointikyky ja kulutus.

Tässä työssä kuvaillaan tasajännitteen mittari, jonka avulla voisi analysoida kuluttavan tai syötettävän sähkön pientehon, muutama sata wattia, järjestelmissä, esimerkiksi aurinkopaneelin ja akuston väliin kytkettynä. Mittaustulokset näkyvät mittarin LCD-näytössä ja ne voi siirtää mittarista tietokoneeseen tutkittavaksi.

Työn rakenne on seuraava: luvussa 1 käsitellään tämän työn käyttökohteita ja määritellään mittarin vaatimukset. Seuraavassa luvussa valitaan tähän sovellukseen sopivat komponentit ja mittaamenetelmät, tämän jälkeen käsitellään laiteiston suunnitteluun ja koodin kehitykseen liittyviä asioita.

Luvuissa 4 ja 5 esitellään saavutetut tulokset ja mahdolliset laiteisto- ja ohjelmistojatkokehitykset ja lopussa käydään läpi yhteenveto tästä työstä.

1.2 Tarvemäärittely

Mittarilaitteen pitäisi kattaa 12 V:n ja 24 V:n DC:n tasavirta-järjestelmät, joissa minimijännite on 9 V ja maksimijännite on 27 V. Maksimiteho on 300 W, maksimivirta 30 A ja tehon mittaustarkkuus on 1–1,5 %. Mittaus tehdään noin

kerran sekunnissa. Mitattavan tehon tarkkuuden mukaan riittäisi LCD-näytössä 6 merkkiä.

Yhteys tietokoneeseen voidaan luoda joko COM- tai USB-väylän kautta.

Syöttöjännite mittalaitteeseen otetaan mitattavasta lähteestä tai USB-väylästä, jos se on käytössä.

2 Perusosien valinta

2.1 Mikrokontrolleri

Koska mittalaitteen täytyy sekä mitata tehoa että välittää tuloksia tietokoneeseen, niin luonnollinen valinta olisi mikrokontrolleri, jossa on sisäänrakennettu AD-muunnin. Valitaan 8-bittinen Atmelin [2] AVR-piiri, joka sisältää kaikki tarvittavat ominaisuudet. Sen lisäksi ohjelmointia varten tarvittava ohjelmisto, WinAVR C-kielen kääntäjä ja AVR Studion IDE-kehitysympäristö ovat vapaasti saatavilla netissä.

Mittari on mikrokontrollerikortti, jonka ytimenä toimii Atmel AVR -perheen Atmega8-mikrokontrolleri. Tässä mikrokontrollerissa on olemassa sisäänrakennettu 10-bittinen AD-muunnin, jonka avulla voi mitata yhtä kuudesta sisään tulevasta analogiasignaalista. Muunnosaika on alle 0,2 ms ja näin hitaasti muuttuvan tasajännitteen mittaaminen onnistuu helposti mittaamalla ensin kulkevaa virtaa I ja heti sen jälkeen jännitettä U (tai päinvastoin). Tämän jälkeen mikrokontrolleri laskee tehon $P = U \cdot I$ ja tulostaa sen sekä LCD-näytölle että tietokoneeseen.

2.2 Anturit

Tehomittausta varten järjestelmissä (tai kortilla) on oltava jännite- ja virta-anturi.

2.2.1 Jännite-anturi

Jännite-anturina toimii yksinkertainen vastusjakaja. Vastusten tarkkuuden ollessa 0,1 %, kahden vastuksen jakajan tarkkuus on 0,2 %:n luokkaa.

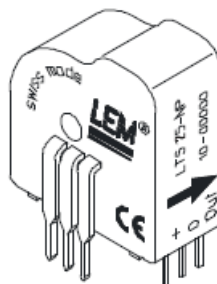
2.2.2 Virta-anturi

Virranmittaus onnistuu kahdella eri tavalla: shunttivastuksella resistiivisenä menetelmänä tai Hall-anturin avulla.

Resistiivisellä virranmittauksella mitataan shunttivastuksen yli syntyvää jännitettä, joka on suoraan verrannollinen virtaan. Tämän menetelmän pääongelmina ovat tehohäviö vastuksessa, joka kasvaa suhteessa virran toiseen potenssiin, sekä pienehkö, muutaman kymmenen millivoltin anturin ulostulojännite. Tässä tapauksessa tarvitaan suuri esivahvistus, koska AD-muuntimen muunnosalue on muutama voltti. Vahvistus onnistuu esimerkiksi operaatiovahvistimella, mutta tämä tarkoittaa kortille lisää komponentteja ja mittausrvirheitä.

Toinen virta-anturin vaihtoehto on Hall-anturi, joka perustuu kulkevan virran magneettikentän mittaamiseen. Tämä anturi sopii hyvin suurempien, yli 10 ampeerin virtojen mittauksiin, koska mitattavassa piirissä ei synny mitään tehohäviötä. Nykyään markkinoilla on useita erilaisia valmiita Hall-anturimoduleita. Näitä löytyy Honeywelliltä, LEM:ltä, Siemensiltä sekä monilta muilta valmistajilta. Näissä moduuleissa on kaikki virranmittausta varten tarvittavat osat, ja ulostulona on säädettävä muutaman voltin jännite, joka sopii hyvin mikrokontrollerin AD-muuntimeen. Moduulit on pakattu kompaktiin, piirilevylle sopivaan koteloon.

Eräs esimerkki on LEM:n LTS 25. Tämän moduulin nimellisvirta on 25 A, käyttöjännite 5 V ja tarkkuus 0,7 %. Se on esitetty kuvassa 1.



Kuva 1. LEM LTS-25 -virta-anturi.

2.3 Muut osat

Kortin käyttöjännite on 5 V, joka saadaan suoraan USB-väylästä tai mitattavasta piiristä regulaattorin avulla.

Koska Atmega8-mikrokontrollerissa on sisäänrakennettu USART-sarjaporttisyysikkö, yhteys tietokoneeseen järjestetään helposti COM-portin kautta tai COM-USB-muuntimen avulla, esimerkiksi FT232RL-piirillä.

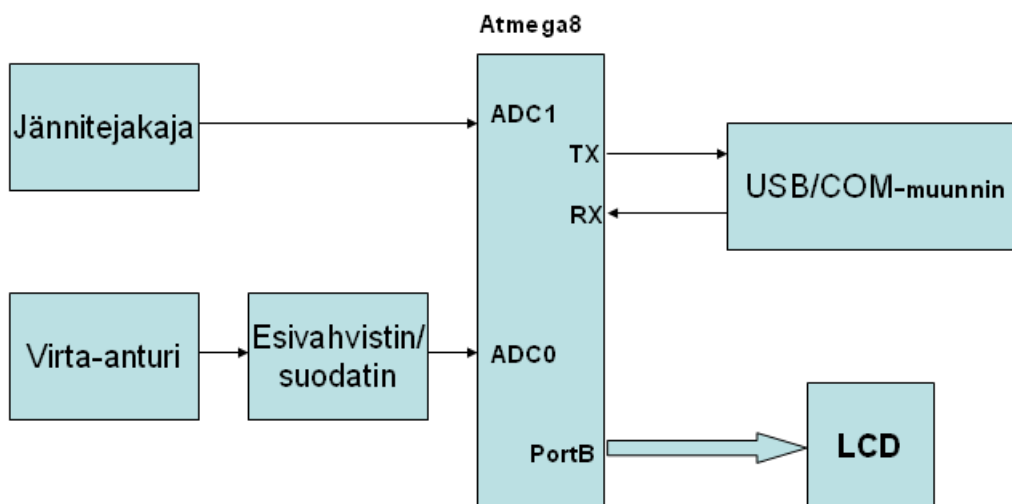
Kortin LCD-näyttö on yhteensopiva Hitachin HT44780 kanssa, näytössä on 1 tai 2 riviä, 16 merkkiä rivissä.

3 Kortin suunnittelu

3.1 Laitteiston suunnittelu

Korttiin kuuluvat ATmega8 mikrokontrolleri IC3, LTS-25 virta-anturi IC1, mitattavan jännitteen jakaja R1-R3, C1, IC2-esivahvistin LMV321-operaatiovahvistimella, FT232RL USB-RS232 -muunnin IC4 sekä LCD-näyttö DIS1.

Kortin kytkentäkaavio on esitetty liitteessä 1 ja lohko-kaavio kuvassa 2.



Kuva 2. Kortin lohkokkaavio.

Kortin analogiseen osaan sisältyy virta-anturi ja sen esivahvistin, jännitteen jakaja ja mikro-ohjaimen AD-muunnin. Näissä kaikissa on oma AGND-maataso, joka liittyy digitaalisen osan maihin vain yhdessä pisteessä häiriöiden vähentämiseksi [3]. Koko analogisen osan 4,7 V:n käyttöjännite tulee regulaattorista IC5. Seuraavaksi eritellään laitteiston osat.

3.1.1 Virta-anturin esivahvistin

Virta-anturina käytetään LEM:n LTS-25-laitetta. LTS-25:n datalehden [4] mukaan jos virta ei kulje anturin läpi, sen lähtöjännite on puolet käyttöjännitteestä, ja poikkeaa -25 mV jokaista mitattua ampeeria kohti, kun virta kulkee anturin läpi. Kortin analogisen osan käyttöjännite on 4,7 V. Suurin mitattava virta on 30 A, jolloin virta-anturin ulostulojännitteen pienin arvo on $\frac{4,7}{2}V - 25mV \cdot 30$ V. Tästä seuraa, että lähtöjännite pysyy 2,35–1,6 V:n alueella.

Koska mikro-ohjaimen AD-muunnin käyttää ulkoista 4,7 V:n referenssijännitettä, AD-muunnosalue on 0–4,7 V. Esivahvistimen tehtävänä on siirtää anturin lähtöjännitettä AD-muunnosalueelle. Tätä varten on invertoiva operaatiovahvistin IC2, jonka lähtöjännite voi olla vain 0,1–4,6 V:n välissä

operaatiovahvistimen saturaation takia [10]. Näin ollen esivahvistimen

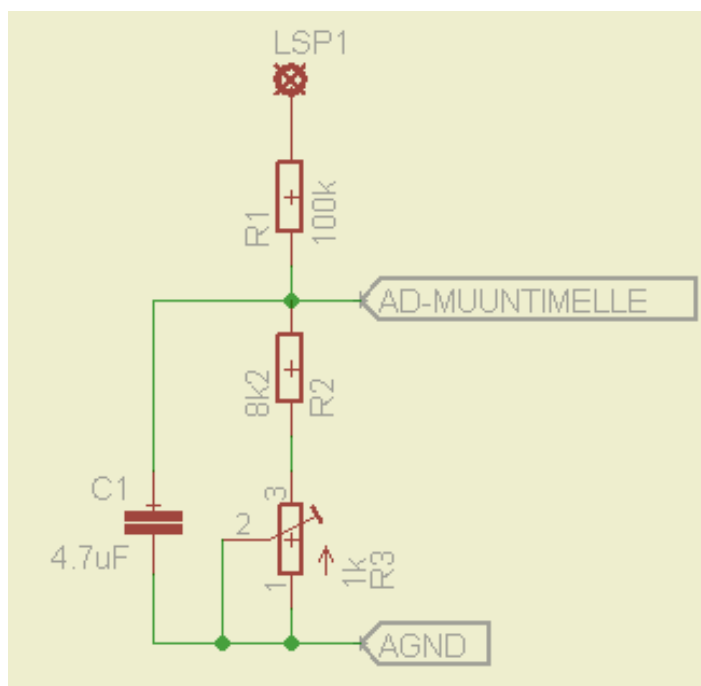
vahvistuskertoimen k on maksimissaan $k = \frac{4,5V}{1,6V - 2,35V} = -6 = -\frac{R9}{R4 + R7}$.

Toinen IC2:n tehtävä on mitattavan signaalin suodatus. Operaatiovahvistin IC2 sekä komponentit R4–R9, C2 ja C3 muodostavat alipäästösuodattimen, joka suodattaa pois yli 10 Hz:n häiriöt ja sillä tavalla parantaa AD-muuntimen tarkkuutta.

Datalehden [8] mukaan TK112-regulaattorin jännite voi saada arvoja $4,7V \pm 50mV$ ja Atmega8:n AD-muunnin käyttää tätä jännitettä referenssijännitteenä. Säätovastukset R7 ja R8 kompensoivat mahdollisen TK112:n jännitteen poikkeaman ja virta-anturin toleranssin.

3.1.2 Mitattavan jännitteen jakaja

Korttiin tuleva mitattava 0–30 V:n jännite muutetaan 0–4,7 V:ksi AD-muunnosalueelle jakajan R1–R3 avulla. Tämän jälkeen jännite tulee IC3-mikrokontrollerin AD-muuntimelle. Jakajan kytkentäkaavio näkyy kuvassa 3.



Kuva 3. Jännitejakajan kytkentäkaavio.

Samalla vastukset R1–R3 ja kondensaattori C1 muodostavat yhdessä RC-alipäästösuodattimen vaimentaen epämieluisat signaalit yli 10 Hz:n taajuudella. Säädetty vastus R3 kompensoi AD-muuntimen referenssi-jännitteen poikkeaman.

3.1.3 Mikrokontrolleri

Mikrokontrolleri IC3 saa käyttöjännitteen USB-väylästä L2, C6, C8 -suodattimen kautta. Mikrokontrollerin 6 MHz:n kello-signaali tulee kideoskillaattorista Q1.

Koodin lataaminen mikro-ohjaimen flash-muistiin tapahtuu SV1-liittimen kautta.

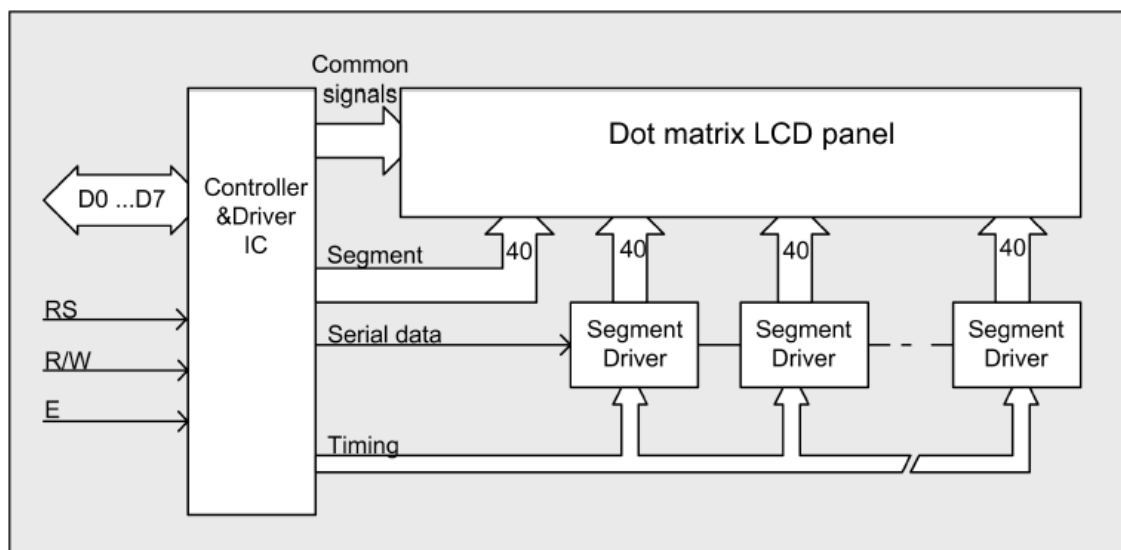
3.1.4 USB-muunnin

USB-muuntimena toimii IC4 FT232RL -piiri [5]. Tämä piiri sisältää lähes kaikki tarvittavat osat ja muodostaa valmiin muuntimen, joka kääntää USB-väylän signaalit RS-232-signaaleiksi ja päinvastoin. Tämän piirin kautta mikrokontrolleri IC3 pystyy lähettämään mitatut tehon arvot tietokoneeseen. Sekä Windows että Linux käyttöjärjestelmien tarvittavat ajurit ovat vapaasti saatavilla FTDI:n nettisivulta [5].

3.1.5 LCD-näyttö

Tällä kortilla käytetään LCD-näyttöä mitattujen tehon arvojen näyttämistä varten. Näin korttia voi käyttää ilman tietokonetta syöttämällä ulkoisen 5 V:n jännitteen USB-liittimen kautta. Näyttö liitetään kortin SV2-liittimeen, josta tulee myös 5 V:n käyttöjännite. Näytön kontrasti säädetään trimmeripotentimetrillä R14.

LCD-näyttö sisältää sisäänrakennetun, Hitachin 44780:n yhteensopivan mikro-ohjaimen, jolla on oma muisti. Näytön lohkokaavio on esitetty kuvassa 4.



Kuva 4. Hitachi 44780 näytön lohkokkaavio.

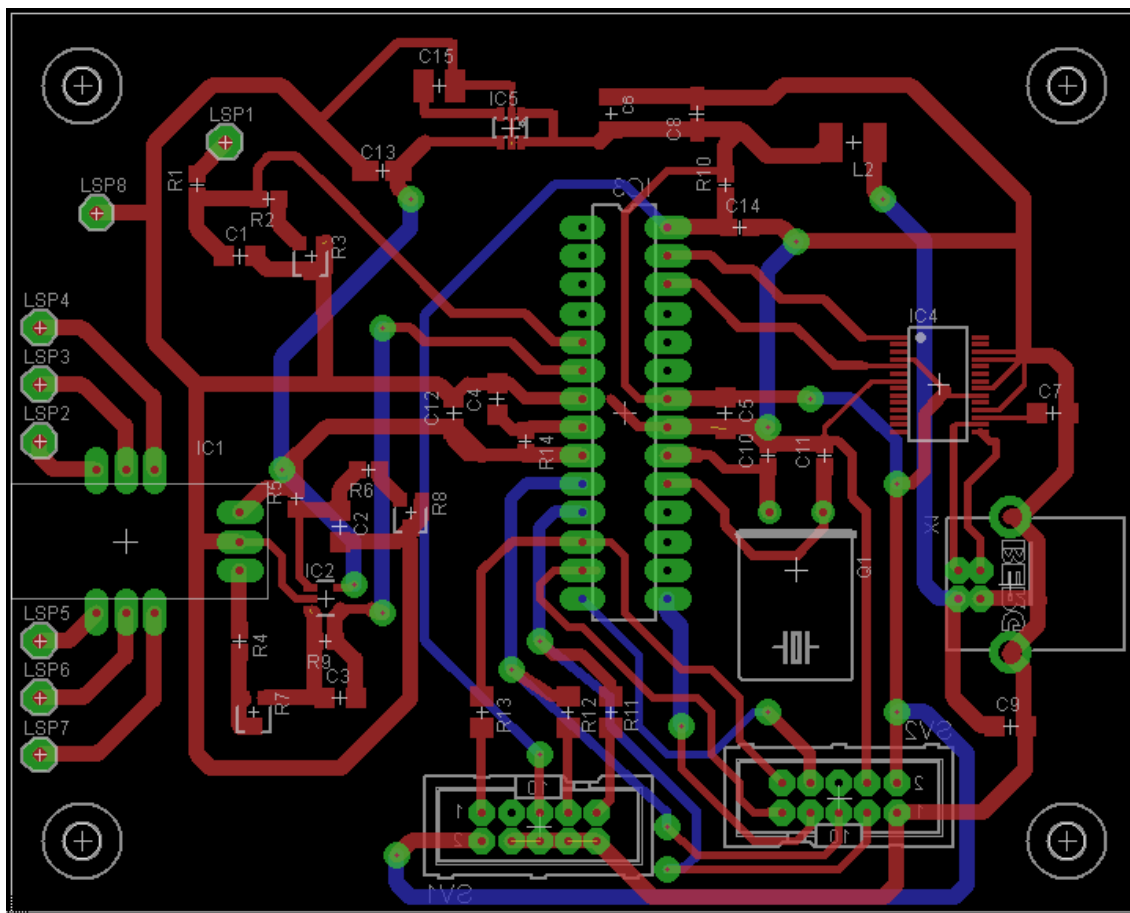
Näytettävät merkit ja käskyt siirretään näytön mikro-ohjaimelle 8-bittisellä dataväylällä. Sen lisäksi on kolme ohjaussignaalia: R/W (Read/Write), RS (Register Select) ja E (Enable). Tavallisesti näyttö kuitenkin kytketään 4-bittisellä toimintomoodilla, jolloin käytössä on ainoastaan neljä datasiinaalia ja kolme ohjaussignaalia [1]. Näin säästetään mikrokontrollerin nastoja.

Koska tämäntyyppisiä näyttöjä käytetään hyvin usein, valmiita koodikirjastoja löytyy paljon, esimerkiksi internetistä. Tässä työssä kirjoitetaan kuitenkin oma pelkistetty koodi mikrokontrollerimuistin tilan säästämiseksi. Näyttö kytketään edelleen 4-bittisellä toimintomoodilla, mutta R/W-signaalia ei käytetä vaan se on kytketty maihin. Tämä on mahdollista, koska kortin mikro-ohjaimen ei tarvitse lukea mitään näytön muistista, vaan mikro-ohjain lähettää merkkejä ja käskyjä oman B-portin kautta käyttäen vain kuutta signaalia.

3.1.6 EAGLE-ohjelmisto

Prototyypikortin kytkentäkaavion ja piirilevyn suunnitteluun käytettiin Cadsoffin EAGLE 5.10.0 Light Edition piirilevyn suunnitteluohjelmaa [9]. Tällä EAGLE:n ilmaisversiolla saa luoda yhden sivun kytkentäkaavion ja 100x80 mm piirilevyn. Tämä riittää hyvin prototyypin suunnitteluun. Kytkentäkaavio on esitetty

liitteessä 1. Vastaava piirilevy näkyy kuvassa 5 ja valmistetun prototyypikortin kuvia on liitteessä 3.



Kuva 5. Prototyypin piirilevy.

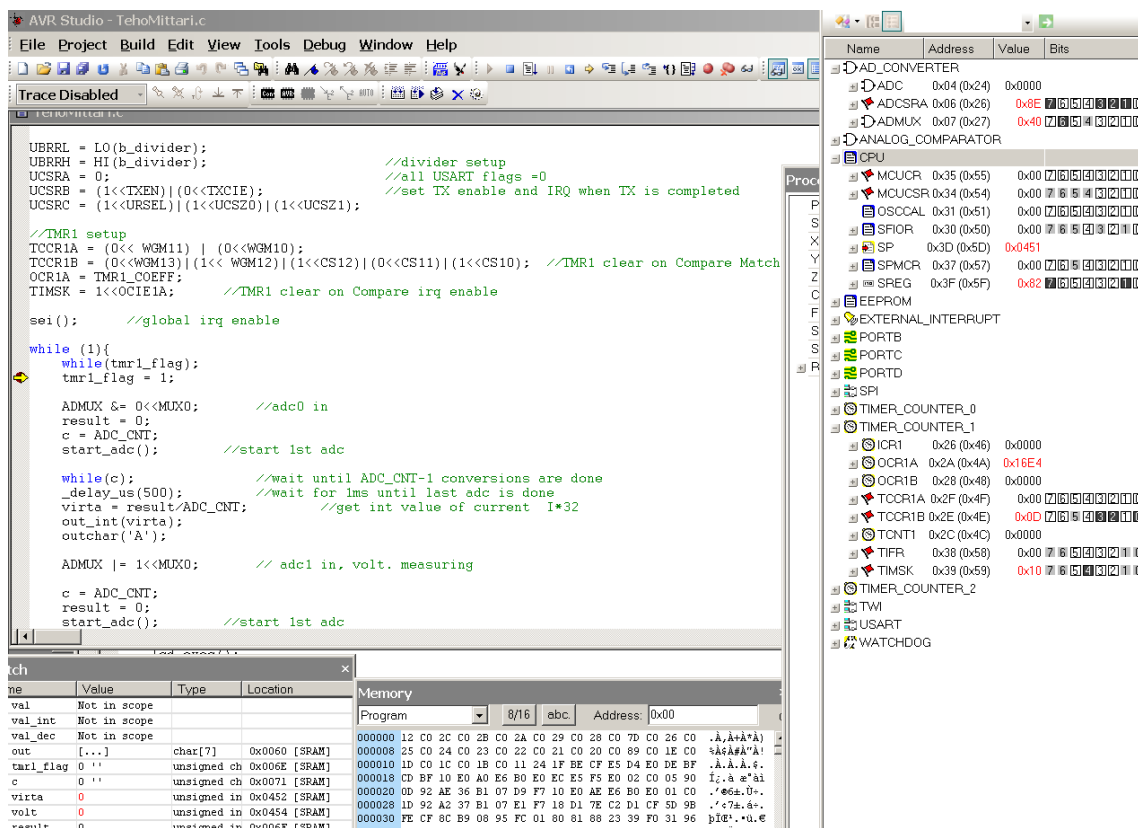
3.2 Koodin kehitys

3.2.1 Kehitysympäristö

Lähdekoodi kirjoitettiin AVR Studioon, joka on vapaasti jaettava valmis IDE eli Integrated Development Environment kehitysympäristö. AVR Studion ohjelmistopaketti sisältää assembler-kääntäjän ja tuen useille ohjelmointi- ja testauslaitteille sekä yksinkertaisen simulaattorin, jonka avulla voisi seurata koodin toimintaa tietokoneella.

Vaikka AVR Studio on alunperin tarkoitettu assembler-koodin kääntämiseen, mikro-ohjaimen koodi kirjoitetaan silti C-kielellä käyttämällä WinAVR GNU-gcc-

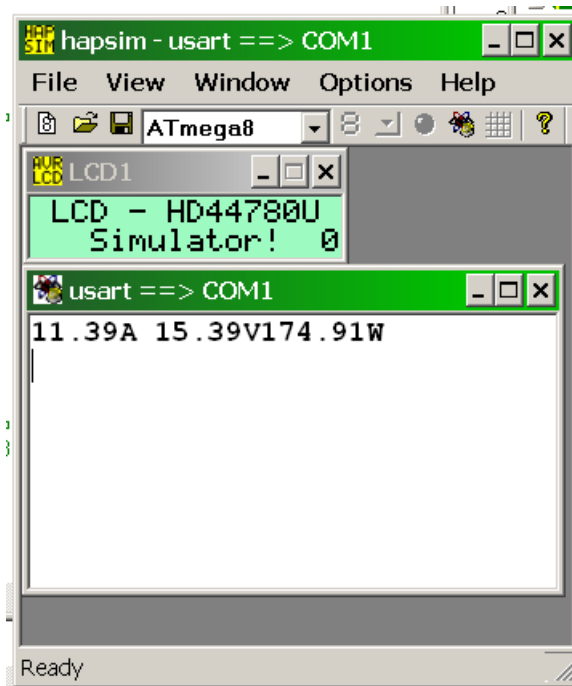
pohjaista käännintä [6]. WinAVR-ohjelmistopaketti koostuu useasta vapaan lähdekoodin projektista ja levitetään vapaasti GNU-lisenssillä. Avr Studio tukee WinAVR:n gcc-käännintä, jolloin C-kieltä saa käyttää AVR Studiossa samalla tavalla kuin assembler-koodia. Kuvassa 6 näkyy AVR Studion simulaattori.



Kuva 6. AVR Studion simulaattori.

3.2.2 Muut apuohjelmat

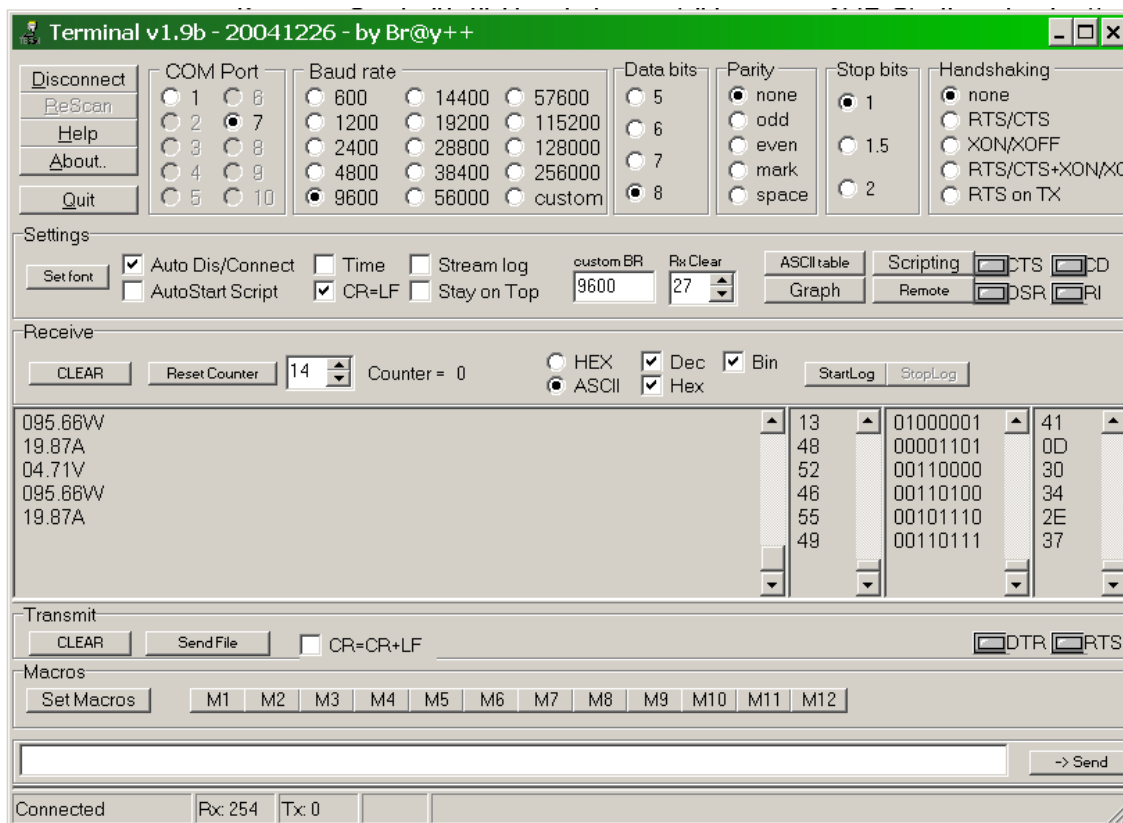
AVR-perheen mikrokontrollerit ovat hyvin suosittuja. Internetissä löytyy useita valmiita vapaita simulaattoreita, joilla pystyy simuloimaan muun muassa 44780 Hitachin LCD:n ja Terminaalin toiminnan. Esimerkiksi Hapsim-simulaattorilla [7] voi nähdä AVR-piiristä LCD-näytölle ja PC:n COM-porttiin lähetettävät merkit. Kuvassa 7 näkyy Hapsimin usart-ikkunaan AVR Studion simulaattorista tulevia merkkejä.



Kuva 7. Hapsim-simulaattorin ikkuna.

Toinen käyttämäni apuohjelma on Terminal v1.9b. Tämä ilmainen ohjelma on hyvä apuväline tässä työssä. Sillä voi seurata COM-portin kommunikointia Windowsin Hyperterminalin sijaan. Ohjelmassa on paljon käteviä asetuksia yhdessä ikkunassa.

Kuvassa 8 näkyy Terminal v1.9b -ikkunaan tulevia arvoja.



Kuva 8. Terminal v1.9 -ohjelman ikkuna.

3.2.3 Mikro-ohjaimen alustus

Alustus tehdään heti sen jälkeen, kun laite on käynnistynyt. Lähdekoodissa alustusosa sijaitsee *tehomittari.c*-tiedostossa main-funktion alussa.

LCD:n alustus tehdään kutsumalla `lcd_init()`-funktion.

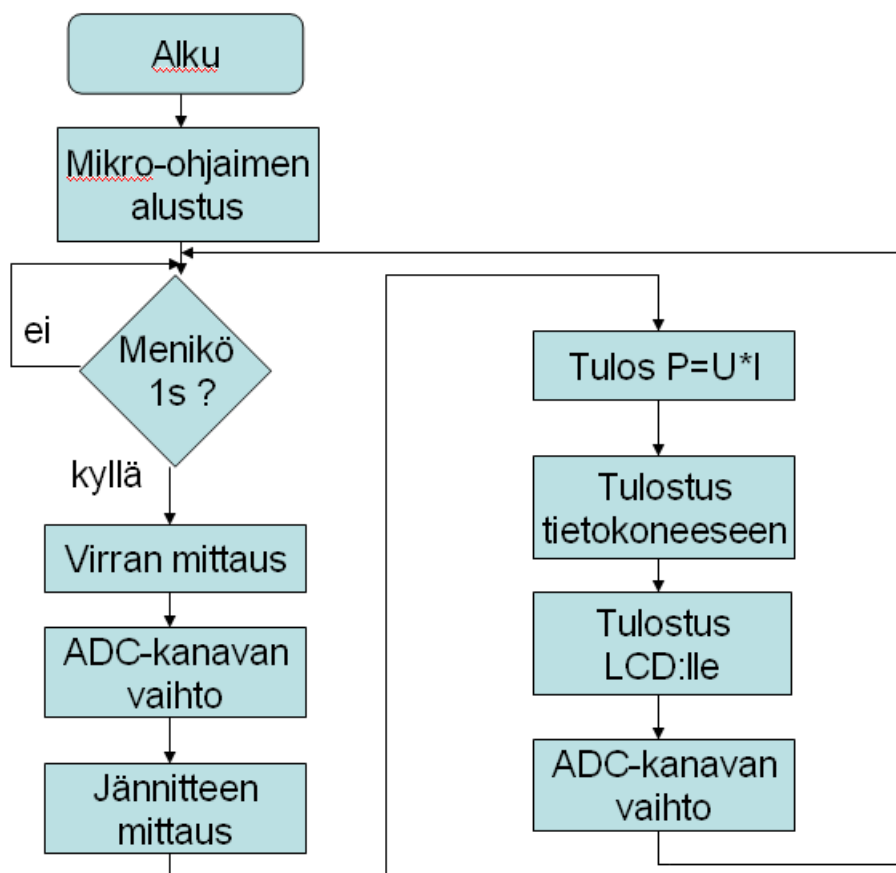
AD-muunnin asetetaan `single conversion` -tilaan eli muunnos alkaa käskystä. AD-muunnoksen valmistuessa syntyy AD-keskeytys. AD-kellotaajuus asetetaan noin 93 kHz:iin datalehden [3] mukaisesti, jolloin yhden muunnoksen aika on noin 0,15 ms.

Mikroprosessorin kellotaajuus on 6 MHz, jolla onnistuu USART-kommunikointi 9600 kbit/s nopeudella. Kääntäjä laskee itse oikeaa USART-alustusta varten tarvittavat arvot datalehdestä otettujen kaavojen mukaan ja kirjoittaa nämä arvot mikro-ohjaimen rekistereihin. Sallitaan myös USART-lähetys ja lähetyksen jälkeen syntyvä keskeytys.

Timer1-ajastin asetetaan Clear Timer On Compare Match CTC -tilaan, jossa laskuri nollautuu tietyllä laskurin arvolla. TMR1_COEFF-vakio määrittelee tämän arvon. Näin laskuri mittaa yhden sekunnin aikavälin. CTC-keskeytys sallitaan.

3.2.4 Pääkoodisilmukka

Pääkoodi sijaitsee *tehomittari.c*-tiedostossa. Ohjelman vuokaavio on esitetty kuvassa 9.



Kuva 9. Pääsilmukan vuokaavio.

Alustuksen jälkeen ohjelma pyörii ikuisessa pääkoodisilmukassa. Yhden sekunnin välein mikrokontrolleri mittaa ensin virran, vaihtaa ADC-kanavan ja mittaa jännitteen. Sitten mikrokontrolleri laskee tehon arvot ja tulostaa ne

kahden desimaalin tarkkuudella tietokoneeseen USB-COM-muuntimen kautta ja LCD-näytölle.

Tarkkuuden parantamiseksi jokainen virta- ja jännitemittaus tapahtuu tekemällä AD-muunnoksen kahdeksan kertaa peräkkäin ja laskemalla sitten keskiarvon. Tämä on mahdollista, koska yhden AD-muunnoksen aika on alle 0,2 ms. Muunnosten määrä vaihdetaan muuntamalla ADC_CNT-vakio.

Koska liukulukulaskenta vie paljon muistitilaa ja prosessori-aikaa, ohjelmassa käytetään ainoastaan kokonaislukulaskentaa.

ATMega8-mikrokontrollerissa oleva AD-muunnin on 10-bittinen, jolloin sen suurin antama lukema on 1023.

Virtamittauksessa AD-muuntimesta tuleva lukema nousee 32:lla jokaista mitattua ampeeria kohti. Lukema ei voi olla kuitenkaan pienempi kuin 22 esivahvistimen saturaation takia. Tämä korjataan vähentämällä AD-muuntimen tuloksesta CUR_SHIFT_LOW-vakion. Suurin mitattava virta on 30 A, jota vastaa AD-muuntimen lukema 960. Jos tämä lukema ylitetään, tehdään virheilmoitus.

Jännitteen mittaus toteutetaan samalla tavalla mutta eri ADC-kanavalla. Jännitteen ollessa 30 V AD-muuntimen ilmoittama lukema on 960 eli jännitearvo voltteina kerrottuna 32:lla. Jos jännite menee yli 30 V, ohjelma lähettää jälleen virheilmoituksen.

Teho lasketaan kertomalla virta- ja jännitearvot keskenään. Näin saadaan long-tyyppinen kokonaisluku, joka sisältää tehon arvon kerrottuna 1024:lla. Tämän jälkeen tehon arvo muunnetaan ASCII-merkkijonoksi set_out()-funktion avulla.

Saatu merkkijono on tehon arvo kahden desimaalin tarkkuudella, ja se tulostetaan tietokoneeseen mikrokontrollerin USART-yksikön kautta. Koska C-kielessä oleva alkuperäinen printf()-funktio kuormittaa mikrokontrolleria paljon, tämän funktion tilalla käytetään paljon kevyempää putchar()-funktia.

Sama merkkijono tulostetaan LCD-näytölle käyttäen lcd_out_string()-funktia.

Virhetilanteissa tulostetaan `er_mess[]`-merkkijono tehon arvon sijaan.

Ohjelman testaamista varten lähetetään tietokoneeseen myös virta- ja jännitearvot `out_int()`-funktiolla.

3.2.5 LCD-koodi

LCD-koodi on koottu `lcd.c`-tiedostoon, joka sisältää kaikki tässä sovelluksessa tarvittavat LCD-funktiot.

Näytön alustus tehdään `lcd_init()`-funktiolla.

`lcd_com()`-funktio antaa yhden tavun kokoisen käskyn näytön mikrokontrollerille.

`lcd_dat()`-funktio tulostaa yhden merkin LCD-näytölle.

`lcd_clear()`-funktio poistaa kaikki merkit näytöstä.

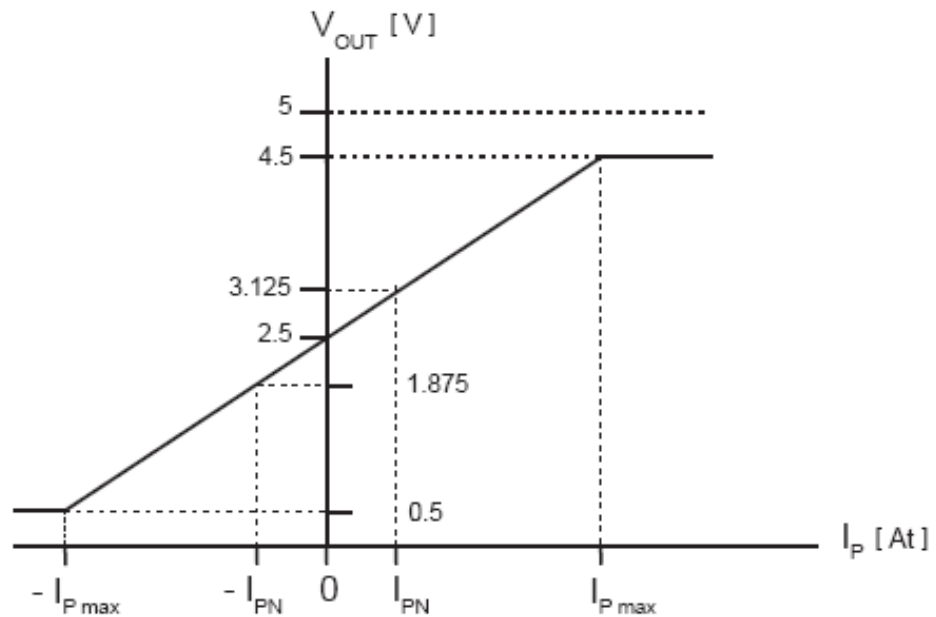
`lcd_home()`-funktio palauttaa näytön osoittimen alkuasentoon.

`lcd_out_string()`-funktio tulostaa näytölle merkkijonon. Merkkijonon lopussa on oltava `'\0'` -merkki.

4 Saavutetut tulokset

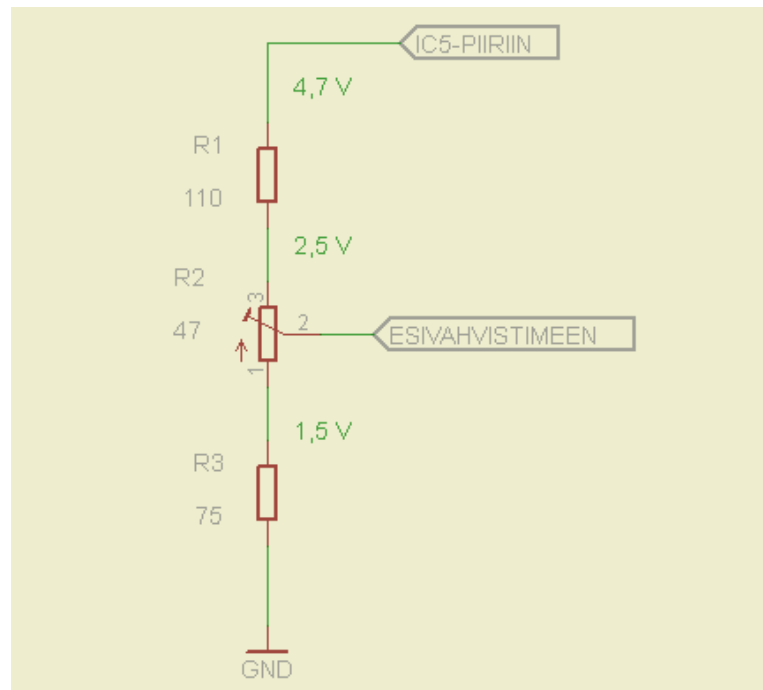
Valmis prototyyppikortti testattiin 12 V:n lyijyakun kanssa. Akun mitattu jännite on 13,16 V. Datalehdessä [4] otettu LTS-25-lähtöjännitteen käyrä on esitetty kuvassa 10. *I_{pn}*-merkki tarkoittaa nimellisvirtaa. Kuvassa käyttöjännite on 5 V, mutta mittarissa anturin käyttöjännite on 4,7 V. Tällöin virran ollessa 0 A lähtöjännite on puolet käyttöjännitteestä eli 2,35 V ja koko käyrä siirtyy 0,15 V:lla alaspäin. Virran muuttuessa 0:sta 30 ampeeriin anturin lähtöjännite muuttuu 2,35:stä 1,6 volttiin.

Output Voltage - Primary Current



Kuva 10. LTS-25-lähtöjännite.

Koska käytössä ei ole LTS-25-tyyppistä virta-anturia, se emuloidaan vastusjakajalla, joka näkyy kuvassa 11.



Kuva 11. Virta-anturin emuloiva vastusjakaja.

Vastusjakajan lähtöjännite on 1,5–2,5 V:n alueella. Tämä jännite syötetään esivahvistimeen virta-anturin lähtöjännitteen tilalla.

Mitatut tulokset ovat taulukossa 1.

Taulukko 1. Prototyypin testaustulokset.

| Virta-anturin jännite, V | Vastaava virta, A | Mitattu arvo, W | tehon Laskettu tehon arvo, W | Suurin poikkeama, W | Suurin poikkeama, % |
|--------------------------|-------------------|-----------------|------------------------------|---------------------|---------------------|
| 2,35 | 0 | 0–0,42 | 0 | 0,42 | – |
| 2,30 | 2 | 25,14–25,56 | 26,32 | 1,18 | 4,5 |
| 2,25 | 4 | 50,7–51,52 | 52,64 | 1,94 | 3,7 |
| 2,20 | 6 | 77,41–78,3 | 78,96 | 1,55 | 2,0 |
| 2,15 | 8 | 104,27–105,93 | 105,28 | 1,01 | 1,0 |
| 2,10 | 10 | 130,23–131,89 | 131,6 | 1,37 | 1,1 |
| 2,05 | 12 | 157,01–158,21 | 157,92 | 0,91 | 0,6 |
| 2,00 | 14 | 183,22–184,64 | 184,24 | 1,02 | 0,6 |
| 1,95 | 16 | 208,65–210,35 | 210,56 | 1,91 | 0,9 |
| 1,90 | 18 | 235,04–236,44 | 236,88 | 1,84 | 0,8 |
| 1,85 | 20 | 261,90–262,95 | 263,2 | 1,3 | 0,5 |
| 1,80 | 22 | 288,76–290,26 | 289,52 | 0,76 | 0,3 |
| 1,75 | 24 | 314,7–316,77 | 315,84 | 1,14 | 0,4 |
| 1,70 | 26 | 341,63–342,45 | 342,16 | 0,53 | 0,2 |
| 1,65 | 28 | 366,7–368,23 | 368,48 | 1,78 | 0,5 |
| 1,60 | 30 | 393,5–395,75 | 394,8 | 1,3 | 0,3 |

Tuloksista näkyy, että mitatut arvot vaihtelevat paljon, näkymät hyppäävät koko ajan. Yli puolisadan watin tehoa mittaessa suurin poikkeama voi olla 2 %. Tämä iso virhe ja näkymän hyppääminen johtuu varmasti sisäänrakennetun AD-muuntimen huonosta laadusta ja häiriöiden vaikutuksesta. Seuraavassa luvussa käsitellään mahdollisia ongelmien ratkaisuja.

5 Mahdollinen kortin jatkokehitys

5.1 Laitteiston jatkokehitys

Saavutettujen tulosten perusteella kortti pystyy mittaamaan 70–300 W:n tehon noin 1–1,5 %:n tarkkuudella. Jatkokehityksen tärkein kohde on kuitenkin laitteen tarkkuuden parantaminen.

AVR-mikrokontrollerin sisäisen 10-bittisen AD-muuntimen resoluutio on yleensä 8 bitin luokkaa [1]. Huolellinen piirilevyn suunnittelu ja parempi referenssijännitelähde voisi hieman parantaa tarkkuutta. Radikaali ratkaisu voisi olla kuitenkin ulkoinen 12-bittinen, 2-kanavainen ADC-mikropiiri, jossa on sisäänrakennettu SPI-väylä. Samantyyppinen SPI-väylä on AVR-mikrokontrollerissa, jolloin AD-muunnoksen tuloksen siirto tapahtuu vaivattomasti ADC-piiristä mikrokontrolleriin digitaalisessa muodossa. Tällaisissa erityisissä ADC-piireissä on oma referenssilähde, vahvistin sekä muita tarvittavia osia. Käyttäen tätä piiriä kortin rakenne yksinkertaistuu, koska esivahvistinta ja säädettäviä vastuksia ei enää tarvita. Tietysti tässä tapauksessa koodin muutos on välttämätöntä.

Toinen parannuskohde on pienempien, alle 70 W:n tehojen mittaaminen. LTS-25:n datalehden [4] mukaan kytkemällä anturin jalat eri tavoin saadaan mitattava nimellisvirta 12 tai 8 A. Näin ollen voidaan parantaa virta-anturin ja mittarin tarkkuutta lisäämällä kortille kytkin, jolla valitaan sopiva virran nimellisarvo.

5.2 Koodin jatkokehitys

AD-muuntimen tarkkuuden parantamiseksi voidaan käyttää myös koodinkehitystä. Esimerkiksi on mahdollista pysäyttää mikroprosessori AD-muunnoksen ajaksi [3]. Koska mikrokontrollerin prosessoriyksikkö ei toimi, häiriöitä ei synny ja sillä tavalla tarkkuus paranee.

Nyt ohjelmassa lasketaan keskiarvo kahdeksasta AD-muunnoksesta. Koska yhden muunnoksen aika on hyvin lyhyt, muunnosten lukumäärää voidaan lisätä esimerkiksi sataan asti vähentäen näin lukeman hyppäämistä, jolloin saadaan enemmän tarkkuutta.

Jos käytetään ulkoista ADC-piiriä, pääkoodin rakenne pysyy suurin piirtein samana, mutta mikrokontrollerin alustukseen ja mittamiseen tulee oleellisia muutoksia. Sisäinen AD-muunnin kytketään pois päältä ja SPI-väylä otetaan käyttöön.

Nykyinen ohjelma käyttää ainoastaan noin 20 % mikroprosessoriresursseja. Tämän takia ohjelmaan voidaan lisätä joitakin hyödyllisiä ominaisuuksia. Voidaan esimerkiksi laskea vuorokauden tehon keskiarvo, suurin ja pienin arvo jne. Nämä arvot voidaan sitten tallentaa mikrokontrollerissa olevaan muistiin tai tulostaa tietokoneeseen.

Tarvittaessa myös virta- ja jännitearvot voidaan tulostaa LCD-näytölle. Hälytys- ja ohjaustoiminnot ovat myös mahdollisia tämän kortin kanssa. Esimerkiksi jos kulkeva virta ylittää tietyn rajan, mikrokontrolleri antaa ohjaussignaalin, jolla kuorma kytketään pois päältä. Kortti voi antaa hälytysmerkin, jos jännite on rajojen ulkopuolella ym.

6 YHTEENVETO

Tässä työssä tarvemäärittelyn mukaan on suunniteltu ja valmistettu toimiva prototyyppikortti eli tasajännitetehton mittari, jolla pystyy mittaamaan akkusysteemissä kulkevan virran tehoa noin 1,5–2 %:n tarkkuudella. Hallimiöön perustuvaa virta-anturia ei kokeiltu, mutta tämä anturi on ainoa hyvä ratkaisu yli 10 ampeerin virtaa mitattaessa, koska anturissa ei synny tehohäviötä. Virta-anturin emulointi antaa kuvan sen toiminnasta tässä mittarissa.

Mittarilla pystytään myös siirtämään saadut tulokset tietokoneeseen USB-väylän kautta. Kokeiltu USB-COM-muunnin toimii virtuaalisen COM-portin kanssa ilman katkoksia ainakin Windows XP -käyttöjärjestelmässä.

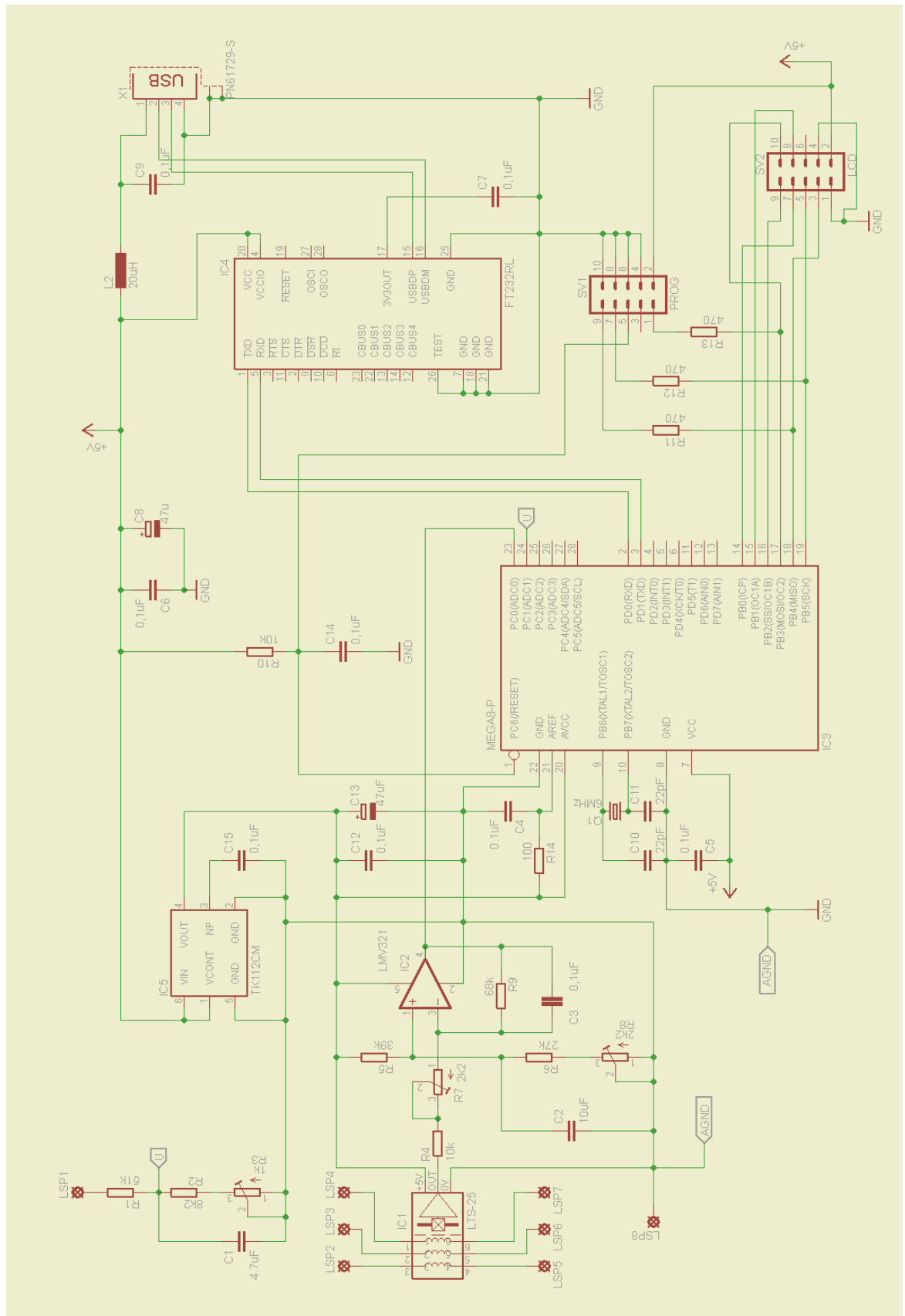
Käytettyjen ilmaisten ohjelmien avulla saatiin lähes valmis, toimiva prototyypin piirilevy. Näitä ohjelmia voi käyttää kortin jatkokehityksessäkin. Näin saadaan käyttökelpoinen malli pienemmillä kustannuksilla.

Saavutettu mittaustarkkuus ei kuitenkaan ole hyvällä tasolla käytetyn sisäänrakennetun AD-muuntimen takia. Luvussa 5 käsiteltiin mahdollisia parannusehdotuksia mittaustarkkuuden parantamiseksi. Nämä parannuskohteet vaativat silti suhteellisen pitkää suunnittelu-aikaa eikä niitä ole kokeiltu tässä työssä. Jatkokehityksen jälkeen kortin pitäisi täyttää kaikki tehomittarin vaatimukset.

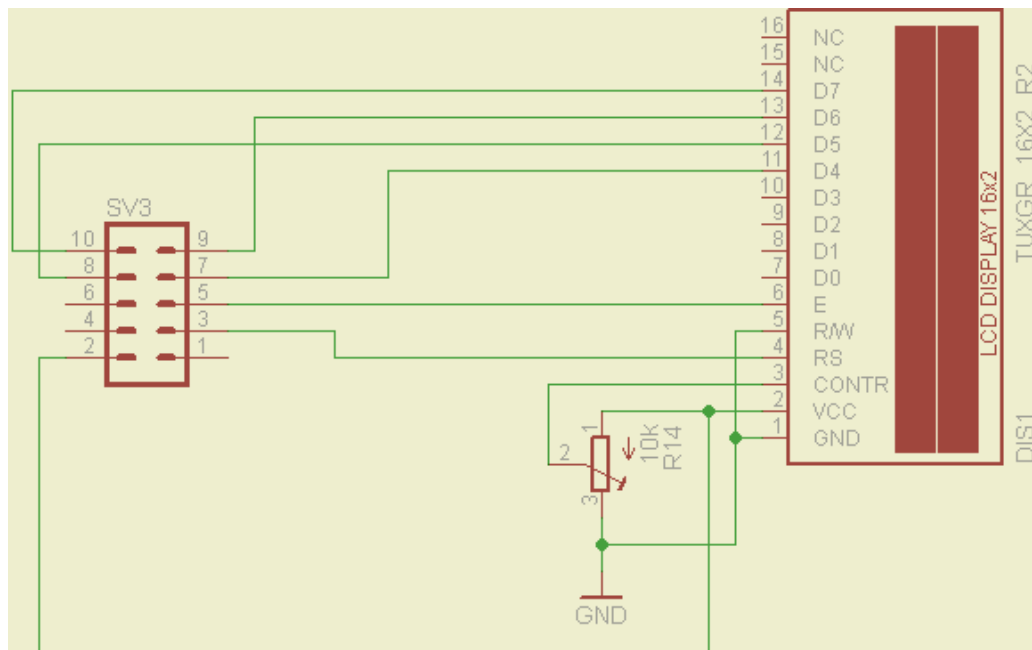
LÄHTEET

- [1] Vahtera, Pentti Mikro-ohjaimen ohjelmointi C-kielellä 2. Microsalo Oy, viitattu 03.12.2010, <http://www.microsalo.com>, 2006.
- [2] Atmel Corporation. <http://www.atmel.com/>, viitattu 03.12.2010
- [3] Atmega8 datasheet
http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf, viitattu 03.12.2010
- [4] LTS 25 datasheet http://www.lem.com/docs/products/lts_25-np_e.pdf, viitattu 03.12.2010
- [5] Future Technology Devices International Ltd <http://www.ftdichip.com/>, viitattu 03.12.2010
- [6] WinAVR GNU-gcc kääntäjä <http://winavr.sourceforge.net/>, viitattu 03.12.2010
- [7] HAPSIM kotisivu <http://www.helmix.at/hapsim/>, viitattu 03.12.2010
- [8] TK112 datasheet
<http://www.digchip.com/datasheets/parts/datasheet/484/TK112-pdf.php>, viitattu 03.12.2010
- [9] EAGLE-ohjelmiston kotisivu <http://www.cadsoft.de/>
- [10] LMV321 datasheet <http://www.national.com/ds/LM/LMV321.pdf> , viitattu 03.12.2010

Liite 1a: Kortin kytkentäkaavio.



Liite 1b: LCD:n kytkentäkaavio.



Liite 2a: Lähdekoodi *tehomittari.c*.

```

#define F_CPU 6000000L           //CPU frequency
#define V_REF 2560L             //internal Vref in mV
#define b_rate 9600L           //USART boudrate
#define b_divider (F_CPU/(16*b_rate)-1) //USART divider
#define HI(x) ((x)>>8)         //high byte of word x
#define LO(x) ((x)& 0xFF)      //low byte of word x
#define ClearBit(reg, bit)  reg &= ~(1<<(bit)) //macro to clear one bit of register
#define SetBit(reg, bit)   reg |= (1<<(bit))   //macro to set one bit

#define TMR1_COEFF 5860        //number to load into TMR1 to get 1s
#define CUR_SHIFT_LOW 22

#define TX PD1                  //USART tx output
#define ADC_CNT 8               //how much conversions
#define start_adc()             ADCSR |= (1<<ADSC) //start 1st AD-conversion
#define CURR_LIM 1000
#define VOLT_LIM 1000

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

//Global variables
volatile unsigned char tmr1_flag = 0; //flag of TMR1
char out[] = "000.00"; //string to UART

static char er_mess[] = "overload";
volatile unsigned int result = 0; //contains adc*8 (max = 1023*8)
volatile unsigned char c = 0; //glob adc counter

```

```

void outchar(unsigned char ch){                                //outputs one byte to UART
    while(!(UCSRA&(1<<UDRE)));
    UDR = ch;
}

void outstr(char *str){                                       //outputs string
while(*str != '\0')    outchar(*str++);
}

void out_int(int v){
    unsigned char vint=v/32;
    unsigned char vdec = v - vint*32;
    outchar(vint/10+0x30);
    outchar(vint%10+0x30);
    outchar('.');
    outchar((vdec*3)/10 +0x30);
    outchar((vdec*3 + 3)%10 +0x30);
}

ISR(TIMER1_COMPA_vect){                                     //timer1 CTC-interrupt routine
    tmr1_flag = 0;                                         // clear flag every second
}

ISR(ADC_vect){                                             //ADC interrupt routine
    while(c){                                              //counters ad conversions
        c--;
        result += ADCL + (((int)ADCH)<<8);                //get 16-bit result when adc completed
        start_adc();                                     //start new adc
    }
}

```

```

unsigned char set_out(unsigned long val){ //converts 0< val/1024 <999 to string
int val_int= val/1024;
int val_dec = (int)(val-val_int*1024);
if(val_dec > 999) val_dec = 999; //decimal fraction cannot exceed 999
unsigned char i=2;
    while(i) {out[i--] = (val_int %10) +48; val_int /= 10;}
    if (val_int >9) return 1; //return 1, if val>999
    else out[0] = val_int +48;

    out[4] = val_dec /100;
    out[5] = (val_dec - out[4] *100) /10 +48;
    out[4] +=48; //digit to ascii
    return 0; //return 0, if conversion succeeded
}

```

```

void main (void){

```

```

volatile unsigned long teho = 0; //contains calculated power value
volatile int virta = 0; //contains measured current value
volatile int volt = 0; //contains measured voltage value

```

```

//PortD setup

```

```

DDRD = (1<<TX) | (1<<PD2); //PD.1 = USART TX is output
DDRC = 0x00; //all PortC pins are inputs
PORTC &= 0xFC;

```

```

lcd_init();

```

```

//ADC enable, single conver., ADC irq enable, ADCfreq = 6000000/64 = 93,75 kHz

```

```

ADCSR = (1<<ADEN) | (0<<ADFR )| (1<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (0<<ADPS0);

```



```

//ADC setup External Uref=4,7V, ADCchan =0

ADMUX=(0<<REFS1)|(0<<REFS0)|(0<<ADLAR)|(0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0);

//USART setup

UBRR_L = LO(b_divider);
UBRR_H = HI(b_divider);           //divider setup
UCSRA = 0;                       //all USART flags =0
UCSRB = (1<<TXEN)|(0<<TXCIE);     //set TX enable and IRQ when TX is completed
UCSRC = (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);

//TMR1 setup

TCCR1A = (0<<WGM11) | (0<<WGM10);
TCCR1B = (0<<WGM13)|(1<<WGM12)|(1<<CS12)|(0<<CS11)|(1<<CS10);           //TMR1
//clear on Compare Match, divider = 1024

OCR1A = TMR1_COEFF;
TIMSK = 1<<OCIE1A;               //TMR1 clear on Compare irq enable

sei();                            //global irq enable

while (1){                        //loop forever
    while(tmr1_flag);             //wait for TMR1 interrupt (until tmr1_flag is cleared)
    tmr1_flag = 1;
    ADMUX &= 0<<MUX0;             //adc0 in - current measurement
    result = 0;
    c = ADC_CNT;                  //sets number of AD-conversions
    start_adc();                  //start 1st adc

    while(c);                     //wait until ADC_CNT-1 conversions are done
    _delay_us(500);               //wait for 1ms until last adc is done
    virta = result/ADC_CNT - CUR_SHIFT_LOW; //get int value of current I*32
    if (virta < 0) virta =0;
}

```

```

out_int(virta);                //output of current value
outchar('A');
outchar(0x0D);

ClearBit(ADCSR,ADEN);        //ADC off
ADMUX |= 1<<MUX0;            // adc1 in, volt. measuring
SetBit(ADCSR,ADEN);         //ADC on
_delay_us(500);              //wait for 500 us until adc channel is switched
c = ADC_CNT;                 //voltage measurement
result = 0;
start_adc();                 //start 1st adc
while(c);                    //wait until ADC_CNT-1 conversions are done
_delay_us(500);              //wait for 500 us until last adc is done
volt = result/ADC_CNT;       //get voltage int value U*32
out_int(volt);               //voltage output
outchar('V');
outchar(0x0D);

teho = ((long)virta)*((long)volt); //teho P*1024*/
//if measured value doesn't exceed limits
if(!set_out(teho) && (virta < CURR_LIM) && (volt < VOLT_LIM)){
    outstr(out);              //output to USART
    outchar('W');
    outchar(0x0D);           //new line

    lcd_out_string(out);      //output to LCD
    lcd_dat('W');
    lcd_home();
}
else{                         //if measured value is too high

```

```
    outstr(er_mess);          //output error message to USART
    outchar(0x0D);           //new line
    lcd_out_string(er_mess); //output error message to LCD
    lcd_home();
}
}
}
```

Liite 2b: Lähdekoodi *lcd.c*.

```

/* LCD driver

AtMega8 LCD port =PORTB ; R/W = 0 ; RS = PB4 ; E = PB5; data = PB0-PB3*/

#include <avr/io.h>
#include <util/delay.h>

//Define LCD-port pins
#define PORT_LCD      PORTB
#define PORT_DDR_LCD  DDRB
#define RS            4
#define E             5

#define ClearBit(reg, bit)  reg &= ~(1<<(bit))      //macro to clear one bit of register
#define SetBit(reg, bit)   reg |= (1<<(bit))        //macro to set one bit

//this define gives one 2us long pulse on pin E
#define lcd_exec()        SetBit(PORT_LCD, E); _delay_us(2); ClearBit(PORT_LCD, E)

void lcd_com(unsigned char dat){                    //writes 1 byte of command
    ClearBit(PORT_LCD, RS);
    PORT_LCD &= 0xF0;
    PORT_LCD |= 0x0F & ( dat>>4);                //send 4-7 bits of dat
    lcd_exec();
    PORT_LCD &= 0xF0;
    PORT_LCD |= 0x0F & dat;                        //send 0-3 bits of dat
    lcd_exec();
    _delay_us(600);
}

void lcd_dat(unsigned char dat){                    //writes 1 byte of data

```

```

        SetBit(PORT_LCD, RS);
        PORT_LCD &= 0xF0;
        PORT_LCD |= 0x0F & ( dat>>4);           //send 4-7 bits of dat
        lcd_exec();
        PORT_LCD &= 0xF0;
        PORT_LCD |= 0x0F & dat;                 //send 0-3 bits of dat
        lcd_exec();
        _delay_us(100);
    }

void lcd_clear(void){           //clears display
    lcd_com(0x01);
    _delay_ms(2);              //min delay 1,6 ms
}

void lcd_home(void){
    lcd_com(0x02);
    _delay_ms(2);
}

void lcd_init(void){           //initialize LCD

    PORT_DDR_LCD = 0x3F;      //bits 0-5 are used for LCD display
    PORT_LCD = 0;
    _delay_ms(60);

    //Magic init sequence
    PORT_LCD = 0x03;
    lcd_exec();
    _delay_ms(5);
}

```

```

PORT_LCD = 0x03;
lcd_exec();
_delay_us(200);

PORT_LCD = 0x03;
lcd_exec();
_delay_us(200);

PORT_LCD = 0x02;                                // 4 bit mode
lcd_exec();
_delay_us(40);

PORT_LCD = 0x02;                                // 4 bit mode
lcd_exec();
_delay_us(40);

lcd_com(0x28);  //, 2 lines, 5*7 font
lcd_com(0x0C); // cursor OFF, display ON

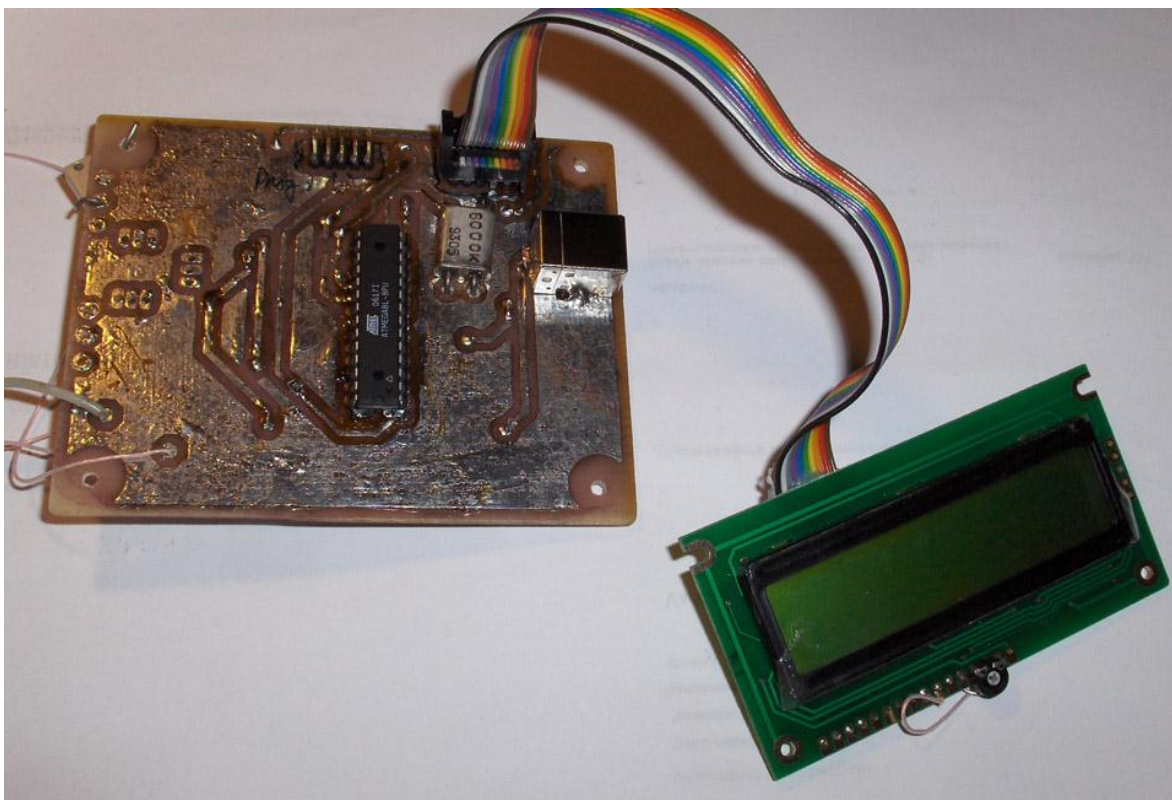
lcd_com(0x06); //autoshift cursor right

lcd_clear(); //clears display
}

void lcd_out_string(char *ptr){
    while(*ptr != '\0'){
        lcd_dat(*ptr++);
    }
}

```

Liite 3a: Prototyypikortin kuva 1.



Liite 3b: Prototyypikortin kuva 2.

