Tampereen Ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät

Opinnäytetyö

Jukka Viinamäki

AMPEERITUNTIMITTARIN SUUNNITTELU JA TOTEUTUS

Työn ohjaaja Kai Poutanen

Tampere 4/2009

Tampereen Ammattikorkeakoulu
Tietotekniikan koulutusohjelma, sulautetut järjestelmät

Tekijä: Jukka Viinamäki
Työn nimi: Ampeerituntimittarin suunnittelu ja toteutus
Sivumäärä: 52 sivua ja 20 liitesivua
Valmistumisaika: 29.05.2009
Työn ohjaaja: Kai Poutanen

TIIVISTELMÄ

Tässä työssä suunniteltiin ja toteutettiin ampeerituntimittari, jolla voidaan seurata akun varaustilaa ja estää sen syväpurkautuminen. Mittarilla mitataan akulta kuormaan kulkevan virran arvoa ja kerrotaan se mittauksien välisellä ajalla. Kertyneiden ampeerituntien määrä on luettavissa LCD-näytöltä.

Virran mittaus perustuu akulta kuormaan menevän syöttökaapelin yli vaikuttavan jännitehäviön mittaamiseen. Mittarissa on mikrokontrolleri, jonka sisäisellä A/D-muuntimella vahvistettua jännitetietoa luetaan. Lämpötilamuutoksen aiheuttama mittausvirhe kompensoidaan mikrokontrollerin ohjelmassa.

Ampeerituntimittari suunniteltiin mökille, jossa on 12 voltin valaistusjärjestelmä ja virtalähteenä toimii suurikapasiteettinen lyijyakku. Akku ladataan muualla ja tuodaan mukana mökkireissun ajaksi. Mökin valaistusta ohjataan releillä ja niiden ohjausjännite voidaan katkaista mittarilla, jolloin valaistus ei enää kuormita akkua.

Käyttäjä voi nollata kertyneet ampeeritunnit, sammuttaa mittarin tai asettaa raja-arvon, jolla mökin valaistus sammutetaan. Tiedot säilyvät muistissa myös silloin, kun mittari ei ole päällä.

Mittarin lopullinen versio on toimiva ja sitä on helppo käyttää. Mittaustarkkuutta voisi parantaa, vaikkakin se sinällään jo käyttökohteeseen riittää.

Suurin osa mittausvirheestä tulee A/D-muunnoksessa, joten seuraavassa versiossa voitaisiin käyttää ulkoista muunninta. Muita merkittäviä virheen aiheuttajia ovat lukujen käsittely ohjelmassa, sekä referenssijännitteen vaihtelu. Mittausvirheeseen vaikuttaa myös lukuisa määrä muita tekijöitä, mutta niiden vaikutus on pienempi.

Mikäli mittaustarkkuutta halutaan parantaa, pitää myös kalibrointiin käytetyn mittarin olla tarkempi.

Avainsanat opinnäytetyö, AVR-GCC, virtamittaus, ampeerituntimittari

TAMK University of Applied Sciences
Department of Computer Engineering, Embedded Systems

Writer: Jukka Viinamäki
Thesis: Design and Development of Ampere-hour Meter
Pages: 52 pages and 20 appendix pages
Graduation time: 29.05.2009
Thesis Supervisor: Kai Poutanen

ABSTRACT

The aim of this project was to design and build an ampere-hour meter which can be used to measure the charge of a battery and to prevent deep discharge. The operating principle of the meter is to measure current and multiply it with the measuring cycle. The accumulated ampere-hours can be read on the LCD-display.

Current is measured by measuring the voltage drop across supply cable. The meter includes microcontroller, which features A/D converter. ADC is used to measure the voltage drop. Temperature drift is compensated in the program of microcontroller.

The ampere-hour meter was designed to operate in a certain summer cabin. There is a 12 volt lighting and lead-acid battery as a power supply in the cabin. The battery is only used in the cabin, but charged elsewhere. Ampere-hour meter controls the lightning. The lights could only be switched on when the meter is turned on

The user could zero the accumulated ampere-hours, turn of the meter or set the limiting value which turns the lightning off. Information will not be lost if the meter is turned off.

The result of this project is a fully functioning ampere-hour meter which is easy to use. The measuring accuracy is not too high, but adequate to the application.

Main part of the measuring error is caused by error in A/D conversion. This error could be reduced by using an external and more precise A/D-converter. Another important factor that affects the measuring error is the use of floating point variables in the program. There are also many other different factors that slightly reduce the accuracy.

If the accuracy of the ampere-hour meter was to be further improved, a more precise meter should be used for calibration.

Keywords Bachelor’s thesis, AVR-GCC, current measurement, ampere-hour meter

SISÄLLYSLUETTELO

1 JOHDANTO

Työn tavoitteena oli suunnitella ja rakentaa ampeerituntimittari, jolla pystytään seuraamaan akun varaustilaa ja estämään sen syväpurkautuminen. Suunnittelun lähtökohtia käsitellään seuraavassa luvussa. Markkinoilla ei ole tiettävästi vastaavaa laitetta, sillä se suunniteltiin nimenomaan tämän käyttökohteen mukaan.

Suunnittelussa käytettiin ohjelmien ilmaisversioita ja komponentit ostettiin paikallisista elektroniikkaliikkeistä. Mittaukset tehtiin Velleman HPS10 -oskilloskoopilla ja Biltema Art.15-124 -yleismittarilla.

Ampeerituntimittarissa on mikrokontrolleri, jonka ohjelman kehittäminen aloitettiin jo alkuvaiheessa kytkennän suunnittelun kanssa. Tällä tavoin projektia saatiin kokoajan eteenpäin, vaikka toisella osa-alueella ilmeni ongelmia. Kytkentää päästiin myös kokeilemaan testikytkennällä ohjelman kanssa ennen piirilevyn suunnittelua.

Suunnittelun osa-alueista kytkentään kului eniten aikaa. Mittarin kytkennästä ja työssä käytetyistä komponenteista kerrotaan laajasti viidennessä luvussa. Jälkimmäisissä luvuissa on kerrottu suppeammin piirilevyn suunnittelusta ja mikrokontrollerin ohjelmasta. Ohjelmakoodi on liitteessä 3.

2 SUUNNITTELUN LÄHTÖKOHDAT

Suunnittelu toteutettiin siten, että ampeerituntimittarissa on käyttökohteessa tarvittavat toiminnot. Lisäksi mittarin käyttöjännite ja mittausalue määräytyivät käyttökohteen mukaan. Komponentteja valittaessa huomiota kiinnitettiin hintaan, saatavuuteen ja suoritusarvoihin.

2.1 Käyttökohde ja tarvittavat ominaisuudet

Ampeerituntimittaria tullaan käyttämään mökillä, jossa on 12 voltin valaistusjärjestelmä ja virtalähteenä toimii suurikapasiteettinen traktorin akku. Mökillä käydään satunnaisesti ja siellä viivytään yleensä päivästä kahteen kerrallaan. Akku ladataan kotona akkulaturilla ja otetaan mukaan mökkireissun ajaksi. Aurinkopaneelit olisivat hyvä vaihtoehto, mutta koska tarve on niin satunnaista, ei niin suurta sijoitusta ole toistaiseksi haluttu tehdä.

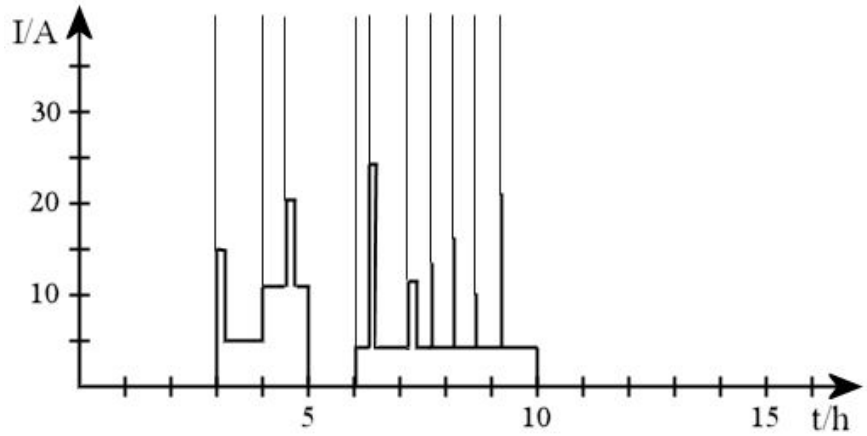
Valaistus on jaettu 10 ampeerin ryhmiin ja valoja ohjataan releillä, joiden ohjausjännitteet tulevat kytkimiltä. Sulakkeita varten on oma keskuksensa. Kaikki liitokset on tehty joko kullatuilla liittimillä tai tinajuotoksilla. Käyttöjännite viedään akulta sulakepesille paksulla kaapelilla.

Mökin lämpötila saattaa muuttua käytön aikana melko paljon, varsinkin syksyllä, jolloin valoja enemmän käytetäänkin. Esimerkiksi mökille tultaessa lämpötila saattaa olla pakkasen puolella. Kun lämmitin ehtii olla muutaman tunnin päällä, lämpötila nousee yli kahdenkymmenen celsiusasteen.

Sähkön kulutusta on vaikeata arvioida. Tästä syystä on myös vaikeata arvioida, paljonko akun kapasiteetista on jäljellä. Mikäli tiedossa ei ole yhtään, paljonko kapasiteetista on käytetty, valot saattavat sammua kesken käytön. Tätä suurempi haitta on akun syväpurkautuminen.

Mökin valaistuksen kokonaisteho on n. 300 W. Yleensä kaikki valot eivät kuitenkaan ole päällä yhtä aikaa, ainakaan pitkiä aikoja. Näin akun kapasiteetti saadaan riittämään muutaman päivän ajan. Usein valoja käytetään myös vain muutama tunti illassa, joten

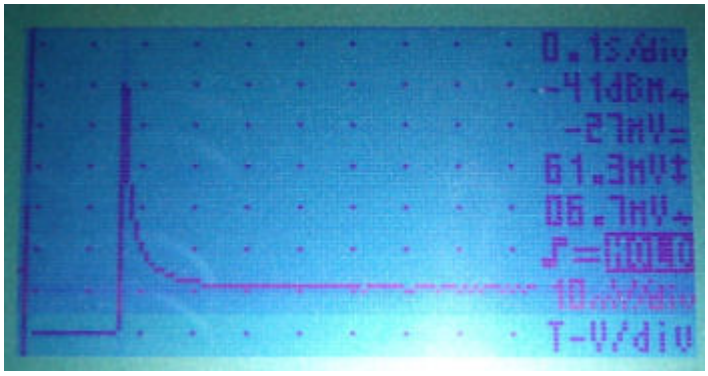
kulutus jää melko vähäiseksi. Kuvassa 1 on arvio, kuinka paljon virran kulutus saattaisi yhden illan aikana olla.



Kuva 1 - Kuvaaja virran kulutuksesta (arvio)

Kuvan 1 kuvaajassa näkyvät kapeat ja korkeat piikit kuvaavat polttimoiden aiheuttamia kytkentäpiikkejä. Suurin osa mökillä olevista valaisimista toimii halogeenipolttimolla. Kun halogeenipolttimossa ei kulje virtaa ja se on kylmä, sen hehkulangan resistanssi on todella pieni. Siinä vaiheessa kun polttimoon kytketään jännite, hehkulangan läpi kulkee hetkellisesti suuri virta. Hehkulanka lämpenee nopeasti suuren virran johdosta ja virran suuruus pienenee nimellisvirran suuruiseksi.

Kytkentäpiikin suuruutta tutkittiin mittauskokeilulla. Akkuun kytkettiin 55 watin tehoinen H1-polttimo, joita käytetään mm. henkilöauton ajovalona. Oskilloskoopilla mitattiin polttimon ja akun positiivisen navan välillä kulkevan johdon yli vaikuttavaa jännitettä. Ohmin lain mukaisesti tämän jännitteen suuruus kuvaa johtimessa kulkevan virran suuruutta. Mittaustulos on kuvassa 2.



Kuva 2 - Välijohdon jännitehäviön suuruus kytkettäessä H1-polttimo

Virran suuruus on kytkentähetkellä n. kuusinkertainen nimellisvirtaan verrattuna (kuva 2). Huippuvirran suuruus on todennäköisesti vielä suurempi. Oskilloskoopin pisteväliksi valittiin 100 ms, joten äärimmäisen kapea piikki ei näy näytöllä. Lisäksi oskilloskoopin näytteenottotaajuus vaikuttaa erottelutarkkuuteen.

2.2 Akun kapasiteetti

Valaistusjärjestelmässä käytetty akku on tyypiltään huoltovapaa lyijyakku. Tämän tyypin akkuja käytetään mm. käynnistysakkuina erilaisissa kulkuneuvoissa, kuten työkoneissa, autoissa ja moottoripyörissä. Yleisesti ottaen akku on sitä suurempi, mitä suurempi kulkuneuvo on kyseessä.

Yleensä akkuvalmistajat ilmoittavat akun kapasiteetin Ah-kapasiteettina, joka määritellään kaavan 1 mukaisesti. (Berndt 2003, 72)

$$C_{Ah} = \int_0^t I(t) \cdot dt \qquad (1)$$

Kaavan 1 mukaisesti kapasiteetti on virran integraali ajan suhteen. Jos virta pysyy vakiona, kapasiteetti on virta kerrottuna ajalla. Ampeerituntimittarin toiminta perustuu tähän ajatukseen. Mittarilla mitataan virta 100 ms välein. Virran oletetaan pysyvän vakiona mittausten välillä. Jokaisen mittauksen jälkeen mitattu virta kerrotaan 100 millisekunnilla ja lisätään kertyneeseen ampeerituntimäärään.

Akun todellinen kapasiteetti ei kuitenkaan ole näin yksiselitteinen. Tämän kaltaisten akkujen valmistajat ilmoittavat akun kapasiteetin, kun akku puretaan vakiovirralla tyhjäksi +25 °C lämpötilassa 20 tunnin aikana. Akun kapasiteetti riippuu kuitenkin purkuvirrasta (tai vastaavasti purkuajasta), lämpötilasta, sekä napajännitteestä jolloin akun katsotaan olevan tyhjä.
(Berndt 2003, 73 - 75)

Huoltovapaa lyijyakku on herkempi syväpurkaukselle, kuin monet muut akkutyypit. Siksi on tärkeää, että akkua ei purettaisi liikaa. (Berndt 2003, 89)

Ampeerituntimittari päätettiin suunnitella siten, että sillä ei koskaan pyrittäisi ottamaan irti täyttä kapasiteettia akusta. Käyttäjä voi asettaa ampeerituntirajan, jolloin akun kuormittaminen lopetetaan. Tämä raja olisi turvallista asettaa esimerkiksi 75 prosenttiin valmistajan ilmoittamasta kapasiteetista.

3 AMPEERITUNTIMITTARIN TOIMINTAPERIAATE

Ampeerituntimittarissa on mikrokontrolleri, jonka sisäisellä A/D-muuntimella mitataan kuormaan menevää virtaa kuvaavaa jännitetietoa 100 ms välein. Mitatun virran arvo kerrotaan 100 millisekunnilla ja lisätään kertyneeseen ampeerituntimäärään.

Virran mittaus perustuu akulta kuormaan menevän kaapelin jännitehäviön mittaamiseen. Jos lämpötila pysyy vakiona, niin myös kaapelin resistanssi pysyy vakiona.

Paikassa, jossa ampeerituntimittaria käytetään, lämpötila ei kuitenkaan pysy vakiona. Tästä syystä virtaa mitataan edellä kuvatulla tavalla, mutta lämpötilan muuttumisesta aiheutuva mittausvirhe korjataan mikrokontrollerin ohjelmassa lämpötilatiedon mukaan. Mikrokontrollerin sisäisellä A/D-muuntimella mitataan myös lämpötilan arvoa. Tähän käytetään eri kanavaa.

Virran mittausalue on 0 - 95 ampeeria. Mittausalue riittää mökin valaistukselle hyvin. Mittausalue riittää, vaikka valaistusta tulevaisuudessa lisättäisiin. Mikäli mittausalue ylittyy, mittari ei hajoa. Se ei vain kykene mittaamaan suurempia virtoja. Myöskään polttimoiden kytkemisestä aiheutuvat virtapiikit eivät riko mittaria.

Ampeerituntimittarissa on LCD-näyttö. Näytölle tulostetaan hetkellisen virran arvo, kertyneet ampeeritunnit, lämpötila, sekä raja-arvo, jolla mittari ja valot sammutetaan.

Käyttäjä voi asettaa kahta painiketta käyttämällä mittariin raja-arvon, jolla mittari sammuttaa itsensä sekä katkaisee ohjausjännitteen valaistusta ohjaavilta releiltä. Tällä estetään akun syväpurkautuminen. Käyttäjä voi myös sammuttaa mittarin suoraan pitämällä toista painiketta pohjassa 6 sekuntia. Jälleen mittarin sammuessa sammuvat myös mökin valot. Mittaria voidaan käyttää siis myös pääkytkimenä. Kun mittari ei ole päällä, virrankulutus on vain joitain mikroampeereita.

Kertyneet ampeeritunnit, Ah-raja ja kalibrointiparametrit tallennetaan mikrokontrollerin EEPROM-muistiin, joten ne säilyvät, vaikka mittari on sammutettuna.

3.1 Ampeerituntimittarin käyttäminen

Ampeerituntimittari laitetaan päälle painikkeella S4. Piirilevyn ensimmäisessä versiossa painike S4 on piirilevyn ulkopuolella, mutta lopullisessa piirilevykuvassa se sijaitsee painikkeiden S1 ja S2 alapuolella. Kuvassa 3 ampeerituntimittari on hetkeä aiemmin laitettu päälle painamalla kytkintä S4. Näytölle tulostuu perusnäkymä, jonka mukaan ampeeritunteja on kertynyt kahdeksan ja raja-arvo on asetettu 23 ampeerituntiin. Kytkinten S1 ja S2 sijainnit on merkitty kuvaan.

Kuva 3 - Ampeerituntimittari perustilassa

Jos ampeeritunteja kertyy raja-arvoksi asetettu 23 ampeerituntia, mittari sammuu ja sammuttaa myös mökin valot. Kun mittari käynnistetään seuraavan kerran, raja-arvoksi asetettu 23 Ah vilkkuu. Vilkkuminen loppuu, jos raja-arvo asetetaan suuremmaksi tai kertyneet ampeeritunnit nollataan.

3.1.1 Raja-arvon asettaminen

Raja-arvo asetetaan painamalla painikkeita S1 ja S2. On samantekevää, kumpaa painiketta painetaan ensin. Jokainen painikkeen S2 painaminen siirtää kursoria yhden numeron oikealle. Kolmannen numeron kohdalta kursori palaa takaisin ensimmäiseen. Jokainen painikkeen S1 painaminen kasvattaa sitä lukua yhdellä, jonka kohdalla kursori

sijaitsee. Näytön tulostus on kuvan 4 mukainen, kun painiketta S2 on painettu kerran. Kursori on tullut näkyviin toisen numeron kohdalle. Kun kytkintä painetaan uudelleen, näkymä muuttuu kuvan 5 mukaiseksi.

Kuva 4 - Näkymä painikkeen S2 painamisen jälkeen

Kuva 5 - Näkymä painikkeen S2 toisen painamiskerran jälkeen

Seuraavaksi raja-arvoksi asetetaan 412 Ah. Raja-arvo voi olla mikä tahansa luku väliltä 0 - 999. Kursoria siirretään painikkeella S2 ja numeroa muutetaan painikkeella S3. Kuvissa 6 - 8 on näytön tulostukset jokaisen numeron asettamisen jälkeen.

Kuva 6 - Näkymä ensimmäisen numeron asettamisen jälkeen

Kuva 7 - Näkymä toisen numeron asettamisen jälkeen

Kuva 8 - Näkymä kolmannen numeron asettamisen jälkeen

Asetettu raja-arvo saadaan tallennettua pitämällä painiketta S2 pohjassa 2 sekuntia. Mikäli mitään painiketta ei paineta, mittari palaa kuvan 3 mukaiseen perusnäkymään ja raja-arvoa ei tallenneta muistiin.

Mikäli painiketta S2 pidetään pohjassa 6 sekuntia, mittari sammuu ja raja-arvoa ei tallenneta. Seuraavan kerran kun mittari käynnistetään, näytölle tulee kuvan 3 mukainen perusnäkymä.

Jos mittari sammutetaan vasta raja-arvon asettamisen jälkeen, raja-arvo tallentuu muistiin.

3.1.2 Kertyneiden ampeerituntien nollaus

Kertyneiden ampeerituntien määrä saadaan nollattua pitämällä painiketta S1 painettuna 2 sekunnin ajan. Mikäli nollaus tehdään, kun raja-arvon asetus on kesken, raja-arvo jää tallentamatta. Kuvassa 9 on näytön tulostus, kun Ah-rajaksi on asetettu 412 Ah ja kertyneet ampeeritunnit on nollattu.

Kuva 9 - Näkymä Ah-rajan tallennuksen ja Ah-lukeman nollauksen jälkeen

3.2 Ampeerituntimittarin kalibrointivalikon käyttäminen

Ampeerituntimittari kalibroidaan säätämällä kuutta piirilevyllä sijaitsevaa trimmeripotentiometriä, joista kahdella vaikutetaan lämpötilamittaukseen ja lopuilla

neljällä vaikutetaan virtamittaukseen. Näiden potentiometrien lisäksi kalibrointiasetuksia muutetaan kalibrointivalikon avulla.

Kalibrointivalikon avulla voidaan muuttaa kahta EEPROM-muistissa sijaitsevaa muuttujaa, jotka ovat T_0 ja T_kerroin. T_0-muuttujaan asetetaan kalibrointilämpötila, eli se lämpötila, jossa virtamittauksen kalibrointi on tehty. T_kerroin-muuttujan arvoa muuttamalla vaikutetaan lämpötilamittauksen tulokseen.

Kalibrointivalikkoon päästään painamalla piirilevyn komponenttipuolella sijaitsevaa painiketta S3. Kuvassa 10 on näytön tulostus, kun painiketta S3 on painettu.

Kuva 10 - Näkymä painonapin S3 painamisen jälkeen

Näyttöön tulostuu teksti ”Aseta T0” ja vasemmalle tulostuu muuttujan T_0 arvo, joka kuvassa 10 on 22,00 $^{\circ}C$. T_0-muuttujan arvoa saadaan pienennettyä painamalla painiketta S2 ja suurennettua painamalla painiketta S1.

Mikäli painiketta S3 painetaan uudelleen, näytölle tulostuu teksti ”Aseta T” kuvan 11 mukaisesti. Nyt vasemmalla puolella näkyy mitatun lämpötilan arvo, joka kuvassa 11 on 20,74 $^{\circ}C$. T_kerroin-muuttujan arvoa voidaan nyt vähentää painamalla painiketta S2 ja suurentaa painamalla painiketta S1.

Kuva 11 - Näkymä painonapin S3 toisen painamiskerran jälkeen

Asetetut arvot saadaan tallennettua pitämällä painiketta S3 painettuna 2 sekunnin ajan. Tallennuksen jälkeen näyttöön tulee kuvan 3 mukainen perusnäkymä. Mikäli mitään

painiketta ei paineta 6 sekuntiin, mittari palaa perusnäkymään ja tehdyt muutokset eivät tallennu muistiin.

4 MIKROKONTROLLERI JA KEHITYSALUSTA

Ampeerituntimittarin kehityksessä käytettiin Digilent Inc:n valmistamaa Minicon-kehitysalustaa, joka valittiin halvan hintansa vuoksi. Toinen merkittävä syy oli se, että siinä on Atmelin AVR-sarjan mikrokontrolleri, johon oli mielenkiintoa tutustua. On myös hyvin todennäköistä törmätä tämän sarjan mikrokontrollereihin työelämässä.

4.1 Atmel AVR

Atmelin AVR-sarjan mikrokontrollerit ovat 8-bittisiä ja käyttävät RISC-arkkitehtuuria, jonka avulla mikrokontrolleri pystyy suorittamaan yhden käskyn yhden kellojakson aikana. AVR-sarjan mikrokontrollerit kuluttavat vähän tehoa, ja niissä on laaja käyttöjännite- ja kellotaajuusalue. Lisäksi niissä on runsaasti erilaisia toimintoja, jotka voidaan tarvittaessa ottaa käyttöön.

AVR-sarja jakautuu edelleen neljään tuoteryhmään, jotka ovat AVR CPU, tinyAVR, megaAVR ja Xmega. Tuoteryhmät eroavat toisistaan varustelutasossa. Kaikki AVR-sarjan prosessorit käyttävät kuitenkin samaa ydinarkkitehtuuria, joten sama ohjelmakoodi toimii kaikissa sarjan mikrokontrollereissa. Tämä tietysti edellyttää, että kohdeprosessorissa on koodissa käytetyt toiminnot (A/D-muunnin tms.).
(Atmel, 2009)

4.2 Digilent Minicon

Ampeerituntimittariin valittiin ATmega168 siitä syystä, että prototyypin kehittelyssä käytettiin Digilentin valmistamaa Minicon-kehitysalustaa, jossa on myös sama mikrokontrolleri. Näin ohjelmaa päästiin kehittämään projektin alusta asti yhtä aikaa kytkennän kanssa. Alustaa käyttämällä pystyttiin myös kytkennän osia kokeilemaan ennen lopullisen piirilevyversion tekemistä.

Minicon on yksinkertainen laite, joka sisältää mikrokontrollerin, neljä lediä, regulaattorin, ESD-suojauksen, reset-painikkeen sekä runsaasti liittimiä. Liitinten kautta pystytään kytkeytymään helposti mikrokontrollerin I/O-liittimiin. ESD-suojaus on toteutettu kytkemällä I/O-nastoihin zener-diodit.

4.3 Mikrokontrollerin ohjelmointi

Minicon-kehitysalustassa olevaa mikrokontrolleria ohjelmoidaan SPI-liitynnän (Serial Peripheral Interface) kautta. Kyseessä on sarjamuotoinen ohjelmointi, jossa ohjelmoitava mikrokontrolleri voi olla juotettuna piirilevylle. Näin esimerkiksi jo käytössä olevan laitteen ohjelmiston päivitys on helppoa. Tällaista menetelmää kutsutaan myös nimellä ISP (In-System Programming).

Ohjelmointiin käytettiin Digilentin valmistamaa ohjelmointilaitetta JTAG-USB, joka toimii tietokoneen USB-liitynnän kautta (kuva 12). Ohjelmointilaite on hyvin pieni ja näyttääkin ulkoapäin pelkältä välijohdolta tietokoneen ja mikrokontrollerin välillä. Samaa ohjelmointilaitetta käytettiin myös ampeerituntimittarin ohjelmointiin.

Kuva 12 - JTAG-USB-ohjelmointilaite

Ohjelmointilaitetta käytetään Digilentin valmistamalla ohjelmalla, jonka saa ilmaiseksi yrityksen www-sivuilta. Ohjelman nimi on AVR Programmer. Se tukee suurinta osaa AVR-sarjan mikrokontrollereista ja toimii Windows-käyttöjärjestelmissä. Sivuilla on myös hyvät ohjeet ohjelman käyttöön.

5 KYTKENTÄ JA KOMPONENTIT

Ampeerituntimittarin kytkentäkaavio on liitteessä 1. Kytkentäkaavio piirrettiin Eagle-ohjelmalla. Kytkentä koostuu useista toiminnallisista lohkoista. Piirikaavio piirrettiin siten, että nämä lohkot erottuvat omina kokonaisuuksinaan ja kytkentäkaaviota on siten helpompi lukea.

Suunnitteluvaiheessa kytkentää kokeiltiin koekytkentäalustalla lohko kerrallaan. Suurempien kokonaisuuksien rakentaminen koekytkentäalustalle olisi ollut hankalaa, joten kytkentää kokeiltiin kokonaisuudessaan vasta piirilevyllä. Kytkentään tehtiin vielä muutoksia ja lisäyksiä, joten piirilevyä jouduttiin muokkaamaan käsin.

Kytkennässä päätettiin käyttää läpiladottavia komponentteja siitä syystä, että koekytkentä voitiin tehdä samoilla komponenteilla. Lisäksi piirilevyversion muokkaaminen on helpompaa läpiladottavia komponentteja käytettäessä.

Ampeerituntimittarissa käytetyt vastukset ovat toleranssiltaan ± 5 % hiilikalvovastuksia. Operaatiovahvistimen OP1 yhteydessä olevat vastukset ovat sovitettuja pareja siten, että vastusten R8 ja R9, R10 ja R11, R17 ja R18, R19 ja R20 resistanssit ovat mahdollisimman samansuuruiset. Myös vastukset R35 ja R36 sovitettiin samalla tavalla. Sovittaminen tehtiin siten, että kolmenkymmenen vastuksen joukosta valittiin mahdollisimman samanarvoiset parit. Vastusten resistanssit mitattiin yleismittarilla.

Operaatiovahvistimen OP1 yhteydessä olevat vastukset sovitettiin siitä syystä, että erovahvistimina toimivien vahvistinten OP1A ja OP1B yhteismuotoisen jännitteen vaimennussuhde (CMRR) olisi mahdollisimman suuri. Vastukset R35 ja R36 sovitettiin siksi, että kyseisen vahvistuskytkennän vahvistukseksi saatiin mahdollisimman tarkkaan -1.

Kaikki ampeerituntimittarissa käytetyt operaatiovahvistimet ovat tyyppiä TL062IP. TL062IP kuluttaa vähän tehoa, siinä on suuri tuloimpedanssi ja se toimii lämpötila-alueella -40 °C…+85 °C. Muita vaihtoehtoja joita kokeiltiin, olivat TL072, TL082, MC33272A. Mittauskokeilun perusteella TL062IP kuitenkin sopi tähän sovellukseen parhaiten. (TL062 datalehti 2004, 1)

Kaikki kytkennässä olevat 100 nF kondensaattorit ovat keraamisia kondensaattoreita, jotka pystyvät luovuttamaan virtaa hetkellisesti suuren määrän. Kondensaattorien C19 ja C20 tehtävänä on vähentää kytkimien S1 ja S2 aiheuttamia kytkentäpiikkejä. Loput 100 nF:n kondensaattorit toimivat kunkin kondensaattorin lähellä olevan piirin virtavarastona. Tällä menetelmällä pyritään vähentämään koko kytkennän häiriöitä.

Kuten liitteen 1 kytkentäkaaviosta voidaan havaita, kytkennässä on erikseen digitaalinen maataso (DGND) ja analoginen maataso (GND). Maatasot yhdistettiin piirilevyllä vain yhdessä pisteessä.

Digitaalinen ja analoginen maataso pidetään erillään siitä syystä, että sillä pyritään vähentämään digitaalisten laitteiden aiheuttamien häiriöiden pääsyä analogiseen maatasoon. Maatasot pitää kuitenkin yhdessä kohdassa yhdistää, sillä digitaaliset ja analogiset laitteet käyttävät samaa virtalähdettä.

Liitteen 1 kytkentäkaavion kohdassa 8D olevat kytkimet S1, S2 ja S3 ovat piirilevylle juotettavia painokytkimiä. Kytkimillä S1 ja S2 on omat ylösvetovastuksensa R38 ja R39, mutta kytkin S3 käyttää mikrokontrollerin sisäistä ylösvetovastusta. Sinänsä ei ole väliä, käyttääkö kyseisessä kohdassa sisäistä vai ulkoista ylösvetovastusta. Ulkoisia vastuksia käyttämällä vähennetään hieman mikrokontrollerin I/O-portin kuormitusta.

Liitteen 1 kytkentäkaavion kohdassa 2A sijaitsevan transistorin Q2 tehtävänä on toimia kytkennässä puolijohdekytkimenä. Kun mittari on pois päältä, transistorin läpi pääsee mittarille vain todella pieni vuotovirta. Kun painokytkintä S4 painetaan, transistorin kantavirta kulkee vastuksen R42 ja painokytkimen S4 kautta ja mittari saa käyttöjännitteen. Mikroprosessorin ohjelmassa asetetaan lähtö PC4 1-tilaan ja transistori Q1 alkaa johtaa. Nyt transistori Q2 kantavirta kulkee vastuksen R42 ja transistorin Q1 kollektorin kautta ja kytkin S4 voidaan vapauttaa.

Kun mittari pitää sammuttaa, ohjelmassa asetetaan lähtö PC4 0-tilaan ja Q1 ei johda enää. Nyt transistorin Q2 kantavirralle ei ole enää kulkureittiä ja koko mittarilta katkeaa käyttöjännite.

5.1 Virtamittauksen operaatiovahvistinkytkentä

Ampeerituntimittarin virtamittaus perustuu syöttökaapelin yli vaikuttavan jännitehäviön mittaamiseen. Syöttökaapeli on kuvattu vastuksena R liitteen 1 kytkentäkaavion kohtaan 3B.

Koska syöttökaapelin resistanssi on erittäin pieni, myös kaapelin yli vaikuttava jännitehäviö on pieni. Tämä jännitetieto pitää vahvistaa suuremmaksi, jotta A/D-muuntimen resoluutio saadaan mahdollisimman hyvin hyödynnettyä. Jännitetiedon vahvistaminen tehtiin operaatiovahvistinkytkennällä.

5.1.1 Syöttökaapeli

Monissa laitteissa virran mittaaminen on toteutettu vastaavalla tavalla, mutta syöttökaapelin jännitehäviön sijaan mitataan piiriin lisätyn vastuksen jännitehäviötä. Akun ja sulaketaulun väliin ei kuitenkaan haluttu lisätä yhtään ylimääräistä liitosta tai vastusta, sillä ne olisivat vain lisänneet tehohäviöitä. Tehohäviöiden huomiointi on erityisen tärkeää tällaisessa kohteessa, jossa virtalähteenä toimii akku.

Valitun menetelmän etu on sekin, että nyt virta voi kasvaa suureksi ja silti se kyetään mittaamaan. Jos kaapelin sijasta käytettäisiin vastusta, siinä pitäisi olla hyvä tehonkesto suuria virtoja varten. Koska kaapelin resistanssi on erittäin pieni, se alkaa lämmetä vasta erittäin suurilla virran arvoilla.

Syöttökaapeli on monisäikeinen kuparijohto, joka on poikkipinta-alaltaan $21\ mm^2$ ja pituudeltaan 2,9 m. Kaapelin toisessa päässä on akkukenkä, ja sen toinen pää kytketään keskuksen sulakepesän liittimelle. Kaapelin molemmista päistä lähtee poikkipinta-alaltaan $1{,}5\ mm^2$ johdot operaatiovahvistinkytkennälle. Nämä johdot liitettiin syöttökaapeliin tinajuotoksella. Syöttökaapelin suuren poikkipinta-alan ja pituuden vuoksi lämpö pääsee helposti johtumaan ja juottamiseen tarvitaan siksi suurta lämpötilaa. Juottaminen tehtiin kaasukäyttöisellä juotoskynällä.

Suunnittelun alkuvaiheessa laskettiin syöttökaapelin resistanssi. Johtimen resistanssi R saadaan laskettua kaavalla 2, jossa l = johtimen pituus metreinä, A = johtimen pinta-ala neliömetreinä ja ρ = resistiivisyys (kuparilla $17{,}2 \cdot 10^{-9}\,\Omega\text{m}$).
(Mäkelä ym. 2000, 120 ja 177)

$$R = \rho \cdot \frac{l}{A} \qquad (2)$$

Kun kaavaan 1 sijoitetaan tunnetut arvot, saadaan resistanssi laskettua:

$$R = 17{,}2 \cdot 10^{-9}\,\Omega\text{m} \cdot \frac{2{,}9\,\text{m}}{0{,}000021\,\text{m}^2} = 0{,}00237\ \Omega$$

Käytettyjen lähtöarvojen epätarkkuudesta johtuen laskettu resistanssin arvo ei ole äärimmäisen tarkka. Se ei kuitenkaan haittaa, sillä tätä arvoa käytetään operaatiovahvistinten vahvistuksen laskemiseksi ja vahvistusta voidaan säätää.

5.1.2 Kaksi mittausaluetta

Virran mittausalue päätettiin jakaa kahteen osaan, joista ensimmäisellä mitataan virtaa alueella 0…25 A ja jälkimmäisellä 0…100 A. Molemmille mittausalueille on omat operaatiovahvistinkytkentänsä ja sisäänmenonsa A/D-muuntimelle. Mikrokontrollerin ohjelmassa vaihdetaan A/D-muuntimen kanavaa mitattavan virran suuruuden mukaan.

25 A:n mittausalueen kytkentä muodostuu operaatiovahvistimista OP1A ja OP2A oheiskomponentteineen sekä A/D-muuntimen sisäänmenosta PC0. 100 A:n mittausalueen kytkentä muodostuu vastaavasti operaatiovahvistimista OP1B ja OP2B oheiskomponentteineen sekä A/D-muuntimen sisäänmenosta PC2.

Mittausalueiden kytkentöjen välinen ero on vahvistuksen suuruudessa. 100 A:n mittausalueella tarvitaan vähemmän vahvistusta, kuin 25 A:n mittausalueella. Tämä johtuu siitä, että suurempi virta aiheuttaa syöttökaapelin yli suuremman jännitehäviön. Tällöin jännite on lähempänä A/D-muuntimen sisääntulon jännitealuetta ja vahvistusta tarvitaan vähemmän.

Kun tunnetaan syöttökaapelin resistanssi, saadaan laskettua, paljonko syöttökaapelin yli vaikuttavaa jännitettä pitää vahvistaa, ennen kuin se syötetään A/D-muuntimen sisäänmenoon. A/D-muuntimen referenssijännitteeksi Vref asetettiin 4,66 volttia, joten muuntimen sisään syötettävän jännitteen pitää olla välillä 0…4,66 volttia. 25 A:n mittausalueen jännitevahvistuksen arvo saadaan laskettua seuraavalla tavalla:

$$A_{25} = \frac{4{,}66V}{(25A \cdot 0{,}00237\,\Omega)} = 78{,}6$$

100 A:n mittausalueen jännitevahvistuksen arvo saadaan laskettua vastaavasti:

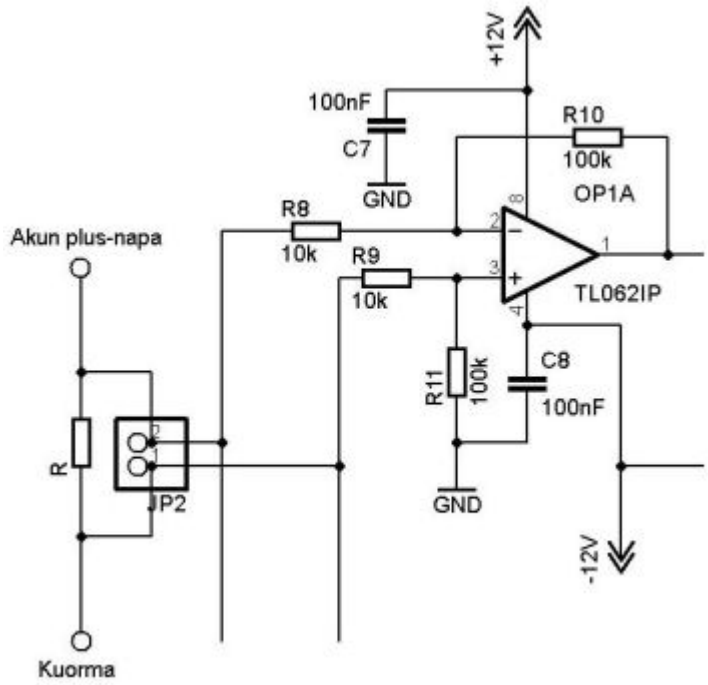
$$A_{100} = \frac{4{,}66V}{(100A \cdot 0{,}00237\,\Omega)} = 19{,}6$$

Jännite vahvistetaan kahdessa vahvistinasteessa, joista tarkemmin seuraavaksi.

5.1.3 Erovahvistin

Syöttökaapelin päihin juotetut johdot kytketään ampeerituntimittarin piirilevylle liittimeen JP2. Tästä jännitetieto haarautuu molempien mittausalueen ensimmäisille vahvistinasteille. Ensimmäinen vahvistinaste on tyypiltään erovahvistin.

Erovahvistimen lähdöstä saatava jännite on sisäänmenojännitteiden erotus kerrottuna vahvistuksella. 25 A:n mittausalueen erovahvistimen kytkentä on kuvassa 13. Jotta operaatiovahvistin toimisi erovahvistimena, pitää vastusten R8 ja R9 sekä vastusten R10 ja R11 olla pareittain samansuuruiset.



Kuva 13 - 25 A:n mittausalueen erovahvistinkytkentä

Akun plus-napa on korkeammassa jännitepotentiaalissa kuin kuormaan kytketty syöttökaapelin pää. Koska akun plus-napa on kytketty operaatiovahvistimen invertoivaan sisäänmenoon, vahvistinkytkentä toimii invertoivana vahvistimena. Jälkimmäinen vahvistin on myös invertoiva, joten näiden kahden vahvistinasteen jälkeen A/D-muuntimelle menee positiivinen jännite.

Erovahvistimen vahvistukseksi valittiin 10. Vahvistus määräytyy vastusten R9 ja R10 välisellä suhteella. Erovahvistimena toimivaa operaatiovahvistinta käytetään $\pm$ 12 V käyttöjännitteellä.

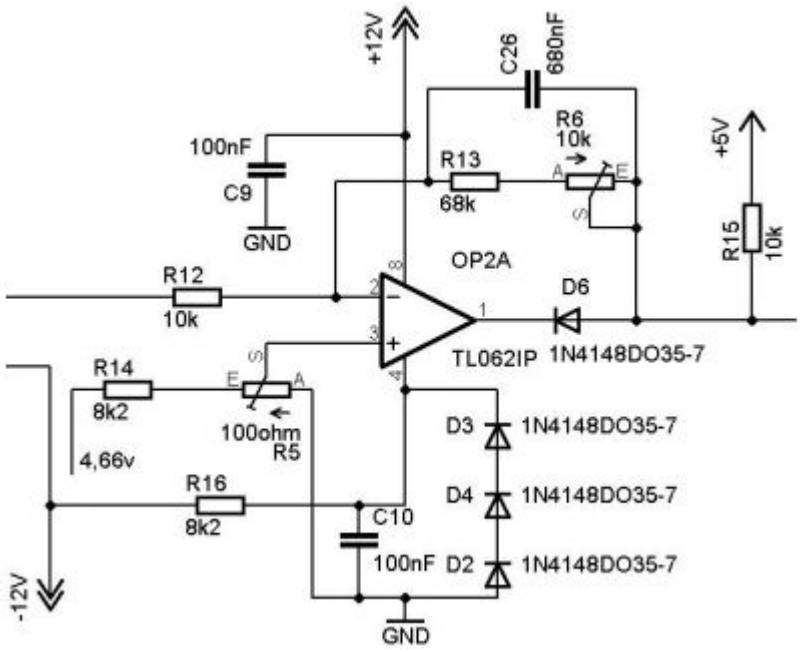
100 A:n mittausalueen erovahvistin eroaa 25 A:n mittausalueen erovahvistimesta pelkästään siten, että sen vahvistus on 2,2.

5.1.4 Invertoiva vahvistin

25 A:n mittausalueen ensimmäisen vahvistinasteen vahvistus on 10, joten jälkimmäisen vahvistimen vahvistus A_2 saadaan laskettua seuraavalla tavalla:

$$A_2 = \frac{78,6}{10} = 7,8$$

Laskettu arvo pitikin yllättävän hyvin paikkansa, sillä todellinen tarvittavan vahvistuksen arvo jäi vain hieman lasketun arvon alapuolelle. Vahvistusta pystyy säätämään trimmeripotentiometrillä R6. Säätöä tarvitaan kalibroitaessa mittaria. Jälkimmäisen vahvistinasteen kytkentä on kuvassa 14.



Kuva 14 - 25 A:n mittausalueen jälkimmäisen vahvistinasteen kytkentä

Jälkimmäisen vahvistinasteen negatiivinen käyttöjännite rajoitetaan diodeilla D2 - 4 ja vastuksella R16 arvoon -1,5 V. Ensin diodeita oli vain kaksi, mutta koska operaatiovahvistin ei toiminut oikein, piti negatiivisen käyttöjännitteen suuruutta nostaa lisäämällä kolmas diodi.

Negatiivinen käyttöjännite on rajoitettu pieneksi siitä syystä, että jos syöttökaapelilta liittimelle JP2 tulevat johdot kytketään ristiin tai virran suunta syöttökaapelissa olisi akulle päin, jälkimmäisen vahvistinasteen ulostulo olisi negatiivinen. Nyt ulostulon negatiivinen jännite rajoittuu n. -0,4 volttiin.

Mikrokontrollerin datalehden mukaan tulojännitteen minimiarvo on -0,5 V. Tulojännite pysyy siis sallituissa rajoissa eikä mittari rikkoudu, vaikka se kytkettäisiin väärin.

Diodin D6 tehtävä on estää virran kulku operaatiovahvistimelta lähdön suuntaan. Lähdössä tarvittava positiivinen jännite saadaan +5 V käyttöjännitteestä vastuksen R15

kautta. Tämän kytkennän tarkoituksena on rajoittaa ulostulon suurin mahdollinen jännite viideksi voltiksi.

Mittauskokeilujen perusteella ulostulo on kuitenkin suurimmillaan n. 4,5 volttia. Ulostulon jännite on sitä suurempi, mitä suurempi takaisinkytkentävastuksen suuruus on vastukseen R15 verrattuna. Tämä johtuu siitä, että ulostulojännite muodostuu näiden vastusten jännitejaosta. A/D-muuntimen referenssijännite oli alun perin 5 volttia, mutta se laskettiin 4,66 volttiin edellä mainitusta syystä.

Koska jälkimmäisen vahvistinasteen suurin ulostulojännite on vähemmän kuin referenssijännite, on selvää, että 25 A:n mittausalueen yläraja on vähemmän kuin 25 ampeeria. Mikrokontrollerin ohjelmassa on määritetty siten, että mittausalue vaihtuu, jos virta on suurempi kuin 23 ampeeria. Näin ollen todellinen mittausalue on 0…23A

100 A:n mittausalueen jälkimmäisen vahvistinasteen kytkentä on täsmälleen samanlainen, kuin 25 A:n mittausalueen. Tästä syystä myös 100 A:n mittausalue loppuu todellisuudessa n. 95 ampeeriin.

Vahvistuskytkennän offset saadaan säädettyä nollaksi trimmeripotentiometrillä R5. Sopivat arvot vastuksille R14 ja R5 löytyivät mittauskokeilun perusteella. Sopivat arvot riippuvat käytetystä operaatiovahvistimesta. Myös 100 A:n mittausalueen vahvistinkytkennässä on vastaavat vastukset R3 ja R23.

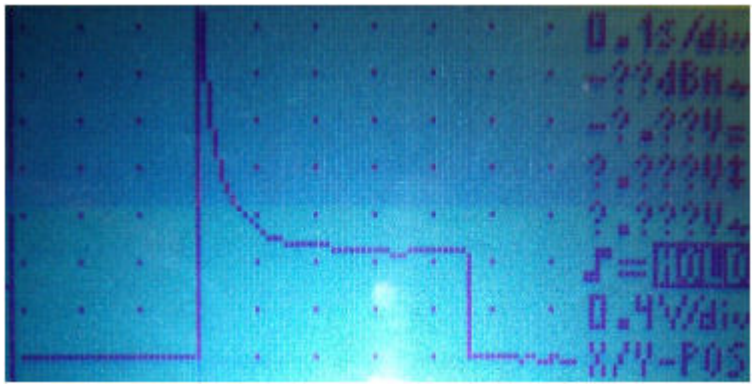
5.1.5 Alipäästösuodatus

Molempien mittausalueiden jälkimmäisiin vahvistinasteisiin lisättiin alipäästösuodatusta, jonka tarkoituksena on suodattaa virtapiikkien aiheuttamia lyhytkestoisia jännitepiikkejä A/D-muuntimen sisäänmenossa.

Koska mittaus suoritetaan vain 100 ms välein, mittaus ei todennäköisesti osu juuri piikin kohdalle. Mikäli mittaus osuu piikin kohdalle, mittausalue ylittyy, eikä piikin todellista arvoa saada mitattua. Suodatus hidastaa jännitteen muuttumisnopeutta, joten jännitepiikistä tulee loivemmin nouseva, lyhyempi ja leveämpi. Tällainen piikki on helpommin mitattavissa, koska piikki ei ylitä mittausaluetta. On myös todennäköisempää, että mittaus osuu piikin kohdalle.

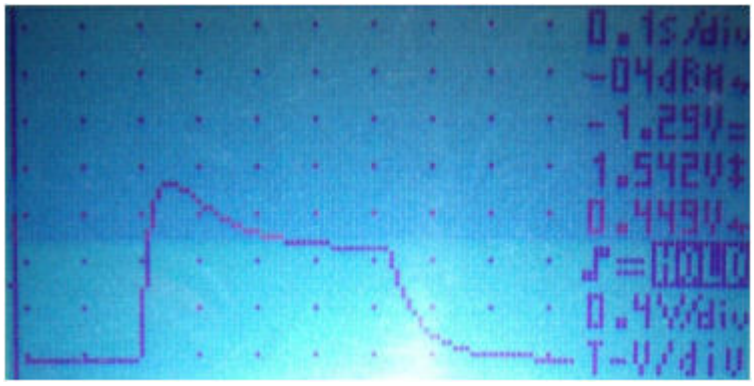
Alipäästösuodatus toteutettiin lisäämällä kummankin mittausalueen jälkimmäisen vahvistinasteen negatiiviseen takaisinkytkentään kapasitanssia. 25 A:n mittausalueen vahvistinasteeseen lisättiin C26 ja 100 A:n mittausalueen vahvistinasteeseen lisättiin C27. Nyt nämä vahvistinasteet toimivat aktiivisina alipäästösuodattimina. Kondensaattorit C26 ja C27 ovat samanlaiset ja sopiva arvo löydettiin mittauskokeilun perusteella.

Kuvassa 15 on oskilloskoopilla mitattu jännite mikrokontrollerin sisäänmenossa 25 A:n mittausalueella, kun kytkennässä ei ole alipäästöä ja kytkentäpiikin aiheuttaa kuormaksi kylmänä kytketty H1-polttimo.



Kuva 15 - Jännite mikrokontrollerin sisäänmenossa PC0, kun kondensaattoria C26 ei ole kytketty

Kuvassa 16 on oskilloskooppikuva, kun sama H1-polttimo aiheuttaa kytkentäpiikin, mutta nyt kytkennässä on alipäästösuodatusta.



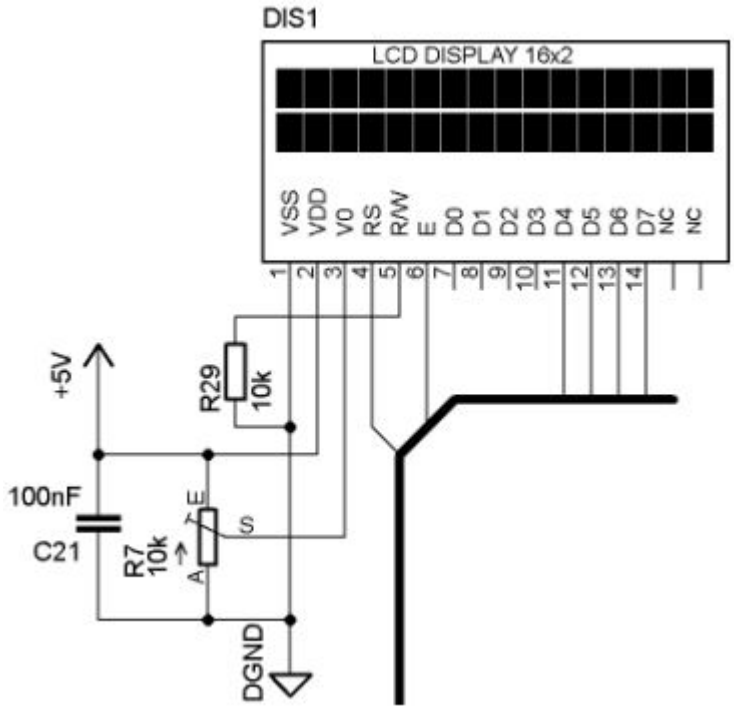
Kuva 16 - Jännite mikrokontrollerin sisäänmenossa PC0, kun kondensaattori C26 on kytketty

Kuvassa 16 jännitepiikki on lyhyempi ja leveämpi kuin kuvassa 15. Nyt jännite myös laskee loivasti, siten osa kytkentäpiikin virrasta mitataan vasta kun polttimo irrotetaan virtapiiristä. Molemmissa kuvissa kahden pisteen väli on 100 ms. Jännitehuippu laskee vain n. 0,2 volttia 100 ms aikana. Koska jännitteen muutokset tapahtuvat hitaammin, mittari ehtii mitata jännitteen tarkemmin. Tällä keinolla mittaustarkkuutta saatiin hieman parannettua, joskin virtapiikkejä ei kyetä vieläkään mittaamaan kokonaan. Sillä ei kuitenkaan ole suurta merkitystä, sillä kytkentäpiikkien osuus kokonaisampeerituntimäärästä on pieni.

5.2 LCD-näyttömoduuli

Yksi ampeerituntimittarin ydinkomponenteista on LCD-näyttö, jolla näytetään kertyneet ampeeritunnit, virran hetkellisarvo, ampeerituntiraja sekä lämpötila.
Näytöksi valittiin YJ162-näyttömoduuli, joka sisältää kaksirivisen nestekidenäytön, sekä KS0066U- tai vastaavan ohjaimen. KS0066U on LCD-näyttöjen ohjaukseen suunniteltu mikropiiri. Tämän moduulin valintaan päädyttiin, koska sen ohjaamisen katsottiin olevan yksinkertaista. Moduuli on myös hinnaltaan huokea.

Moduulista on saatavilla niukasti tietoa. Kytkennät ja ohjelma tehtiin pitkälti KS0066U-ohjaimen datalehden tietojen mukaan. LCD-näyttömoduulin kytkentä on kuvassa 17. Vastus R7 on kytkennässä kontrastin säätöä varten. Piirilevyllä olevien säätövastusten määrää haluttiin kuitenkin pienentää, joten vastus R7 korvattiin $10\,\Omega$ vastuksella, joka yhdistettiin moduulin nastasta V0 maapotentiaaliin. Tähän arvoon päädyttiin mittauskokeilun perusteella.



Kuva 17 - LCD-näyttömoduulin kytkentä

Tiedonsiirtoon KS0066U-ohjaimen ja mikrokontrollerin välillä voidaan käyttää joko 4- tai 8-bittistä rinnakkaismuotoista dataväylää. 4-bittinen väylä valittiin mittariin johdinten pienemmän lukumäärän takia. Käytettäessä 4-bittistä väylää, datalinjat D0-D3 jätetään kytkemättä ja data siirretään datalinjoilla D4 - 7.

Dataväylällä siirtyvän tiedon suunta valitaan R/W-signaalilla siten, että suunta on 1-tilassa ohjaimelta mikrokontrollerille ja 0-tilassa päinvastoin. Näytön ohjaus toteutettiin siten, että KS0066U-ohjainta ohjataan mikrokontrollerilla käyttämällä vakioaikaisia viiveitä käskyjen välillä. Dataa siirtyy siis pelkästään mikrokontrollerilta ohjaimen suuntaan. Tästä syystä R/W-nasta on kytketty vastuksen R29 kautta maapotentiaaliin. (Kuni 2006, 31 - 32)

Ohjaimen RS-signaalilla määritetään, laitetaanko sisään syötetty data ohjaimen näyttömuistiin vai käskymuistiin. Näyttömuistiin kirjoitetaan merkit, joita näytöllä halutaan näyttää ja käskymuistiin kirjoitetaan ohjauskomennot. Ohjauskomennoilla voidaan määritellä mm. kursorin näkyminen, kursorin vilkutus, näytön tyhjennys, kirjoitussuunta ja kursorin palautus alkuun.

Ohjaimen E-signaalilla sallitaan datan lukeminen ohjaimelta, tai kirjoittaminen ohjaimeen. Data luetaan väylältä ohjaimelle E-signaalin laskureunalla.

Ohjaimen nastat D4 - 7, RS ja E on suoraan kytketty mikrokontrollerin nastoihin. Tämä on mahdollista siksi, että mikrokontrollerin ulostulo ja ohjaimen sisäänmeno ovat TTL-yhteensopivia, eli niiden 0- ja 1-tilojen jännitetasot sopivat yhteen.

5.3 Atmel AVR ATmega168 -mikrokontrolleri

Atmel AVR ATmega168 -mikrokontrolleri sisältää 8-kanavaisen 10-bittisen A/D-muuntimen, kaksi 8-bittistä laskuria, 16-bittisen laskurin, USARTin, SPI-liitynnän, TWI-liitynnän, kuusi PWM-kanavaa, komparaattorin, watchdog-kellon omalla sisäisellä oskillaattorillaan sekä runsaasti muita oheislaitteita. Muita ominaisuuksia ovat mm. sisäinen oskillaattori, sisäisten ja ulkoisten keskeytysten käyttö, viisi erilaista virransäästötilaa, resetointi käynnistettäessä sekä ns. brown-out-tunnistus. (ATmega168 datalehti 2007, 1)

Kyseinen mikrokontrolleri siis sopii moniin erilaisiin sovelluksiin. Myös muistitilaa on melko paljon: 16 kilotavua flash-ohjelmamuistia, 1 kilotavua SRAM-muistia sekä 512 tavua EEPROM-muistia.

Ampeerituntimittarin mikrokontrollerin kellotaajuus on 8 MHz. Kellolähteeksi valittiin ulkoinen kide, jotta mittausvirhe saatiin mahdollisimman pieneksi. Kide on kytketty datalehden suosituksen mukaisesti 20 pF kondensaattoreilla C22 ja C23 maatasoon. Mikrokontrollerin kytkentä on liitteen 1 piirikaavion kohdassa 5C.

Mikrokontrollerilla tehdään A/D-muunnos kolmesta eri kanavasta. Näistä kanavista mitattavat jännitteet kuvaavat kahdessa ensimmäisessä kanavassa virtaa ja kolmannessa lämpötilaa. A/D-muuntimen käyttämä referenssijännite on kytketty Aref-nastaan. Mikrokontrollerin kytkennässä ja ohjelmassa on tehty valmistajan suosittamat kohinan vähennystoimet. Näihin sisältyy käyttöjännitteen kytkeminen AVCC-nastaan LC-suotimen kautta. Tämä on toteutettu kelalla L1 ja kondensaattorilla C24.

Ohjelmointi tapahtuu Digilentin USB-JTAG-ohjelmointilaitteella. Ohjelmointia varten kytkentään lisättiin liitin, johon ohjelmointilaite kytketään. SPI-signaalien lisäksi ohjelmointilaitteen 6-napaiseen liittimeen on kytketty maataso ja 5 voltin käyttöjännite. Ohjelmointilaite toimii jännitealueella 1,8 – 5,5 V.

Ohjelmoinnin aikana käynnistyspainike S4 täytyy pitää pohjassa, sillä mikroprosessori saa käyttöjännitteen transistorin Q2 kautta ja transistorin kantavirta kulkee painikkeen S4 kautta maatasoon. Painikkeen painamisen sijaan liittimeen JP3 voi laittaa oikosulkupalan ohjelmoinnin ajaksi.

5.4 Lämpötilan mittaus

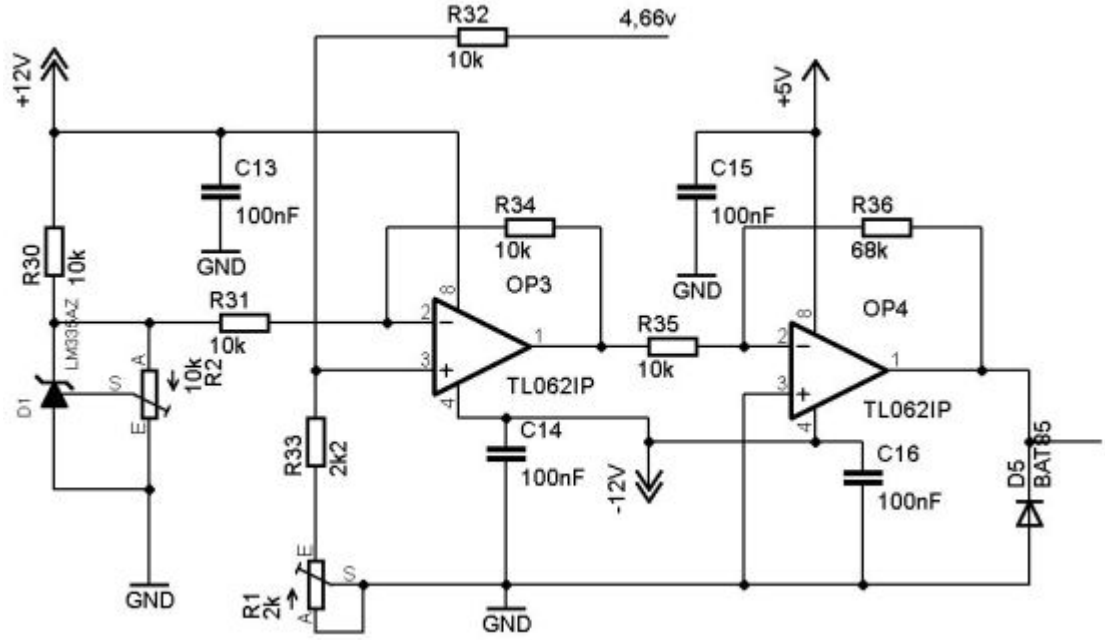
Ampeerituntimittarissa tarvitaan lämpötilan mittausta, jotta lämpötilavaihtelun aiheuttama virran mittausvirhe saadaan kompensoitua. Lämpötilan mittaus voidaan tehdä usealla eri tavalla. Käytetty tekniikka perustuu pn-liitoksen kynnysjännitteen muuttumiseen lämpötilan funktiona.

Lämpötila-anturiksi valittiin LM335, joka on pakattu TO-92-koteloon. LM335 toimii ulospäin kuin zener-diodi, jonka läpilyöntijännite on suorassa suhteessa lämpötilaan. Läpilyöntijännite kasvaa 10 millivolttia, kun lämpötila kasvaa yhden celsiusasteen. Tämän jännitteen muutos on lineaarinen.

LM335 sisältää useita transistoreita ja vastuksia sekä muutaman kondensaattorin. Käytännössä tätä kokonaisuutta voidaan kuitenkin pitää zener-diodina. Tämän zener-diodin kahden nastan (V+ ja V-) lisäksi anturilla on ohjausliitäntä Vadj, jolla zenerjännite voidaan asettaa. Zenerjännitteen asettamisen jälkeen anturin ulostulojännitteen virhe on enintään 2 $^{\circ}$C.

Koska anturin ulostulojännite on lineaarisessa suhteessa ympäröivään lämpötilaan, kalibrointi yhdellä lämpötilan arvolla korjaa anturin ulostulojännitteen virheen myös muilla lämpötilan arvoilla.

Lämpötilan mittauskytkennässä käytettiin datalehden esimerkin mukaista kytkentää (kuva 18).



Kuva 18 - Lämpötilan mittauskytkentä

LM335-lämpötila-anturi toimii datalehdessä luvattujen arvojen mukaisesti, kun estosuuntainen virta on 450 µA …5 mA. Etuvastus R30 on mitoitettu tämän virta-alueen mukaisesti. Mitta-anturin asetettu zenerjännite on +25 °C lämpötilassa 2,982V. Lämpötila-anturin ulostulojännitteen arvo lämpötila-alueen pienimmällä arvolla, eli lämpötilassa -35 °C saadaan laskettua seuraavalla tavalla:

$$U_{min} = 2{,}982\,\text{V} - ((25 + 35) \cdot 0{,}01\,\text{V}) = 2{,}382\ \text{V}$$

Vastaavasti ulostulojännitteen arvo lämpötilassa +35 °C on:

$$U_{max} = 2{,}982\,\text{V} - ((35 - 25) \cdot 0{,}01\,\text{V}) = 3{,}082\ \text{V}$$

Etuvastuksen R30 minimiarvo on:

$$R30_{min} = \frac{12\,\text{V} - 2{,}382\,\text{V}}{0{,}005\,\text{A}} = 1923\ \Omega$$

Etuvastuksen R30 maksimiarvo on:

$$R30_{max} = \frac{12\,\text{V} - 3{,}082\,\text{V}}{450 \cdot 10^{-6}\,\text{A}} = 19817\ \Omega$$

Etuvastukseksi valittiin 10 kΩ vastus, sillä se on laskettujen rajojen välissä ja rajoittaa näin zener-diodin läpi kulkevan virran sopivaan arvoon.

Lämpötila-anturin jälkeisen operaatiovahvistinkytkennän tarkoitus on sovittaa anturin lähtöjännite sopivaksi A/D-muuntimen sisäänmenolle: Lämpötila-alueen pienimmällä arvolla ulostulon jännite on 0 V ja suurimmalla arvolla enintään 4,66 V.

Ensimmäinen lämpötila-anturin jälkeinen operaatiovahvistinaste ei vahvista sisään syötettyä jännitettä. Vastukset R31 ja R34 ovat saman suuruisia, joten vahvistus on -1. Vahvistinasteen tarkoituksena on siirtää ulostulon jännitealue siten, että kun lämpötila on -35 °C, vahvistinasteen ulostulo on 0 V. Jotta tämä tapahtuisi, pitää ei-invertoivaan sisäänmenoon U_{in+} syöttää tietyn suuruinen jännite. Tämä jännite saadaan laskettua seuraavalla tavalla.

Oletetaan, että ei-invertoivaan sisäänmenoon U_{in+} ei mene yhtään virtaa.
Mikäli sarjaan kytkettyjen vastusten R1 ja R33 yhteisresistanssi on saman suuruinen kuin vastuksen R32, vahvistinasteen sisäänmenoon U_{in+} menee puolet stabiloidusta jännitteestä:

$$U_{in+} = \frac{4{,}66\ \text{V}}{2} = 2{,}33\ \text{V}$$

Aiemmin lasketun mukaisesti, lämpötila-anturin ulostulon jännite on -35 °C lämpötilassa 2,382V. Ensimmäisen vahvistinasteen ulostulo olisi näillä jännitteiden arvoilla seuraava:

$$2{,}33\ \text{V} - (2{,}382\ \text{V} - 2{,}33\ \text{V}) = 2{,}278\ \text{V}$$

Jotta ulostulon arvo saataisiin lasketun 2,278 voltin sijaan 0 volttiin, pitää jännitettä U_{in+} muuttaa. Oikea U_{in+} arvo saadaan, kun edellisen yhtälön oikea puoli asetetaan nollaksi:

$$U_{in+} - (2{,}382\ \text{V} - U_{in+}) = 0$$

Kun yhtälö ratkaistaan tuntemattoman jännitteen U_{in+} suhteen, saadaan jännite laskettua seuraavalla tavalla:

$$U_{in+} = \frac{2{,}382\,\mathrm{V}}{2} = 1{,}191\,\mathrm{V}$$

Eli kun ei-invertoivaan sisäänmenoon syötetään 1,191 voltin jännite, ulostulo -35 °C lämpötilassa on 0 volttia. Kun oletetaan, että operaatiovahvistimen ei-invertoivaan sisäänmenoon ei mene yhtään virtaa, jännite saadaan muodostettua yksinkertaisella kahden vastuksen jännitejaolla.

Kun vastuksen R32 arvoksi valitaan 10 kΩ, saadaan sarjaan kytkettyjen vastusten R1 ja R33 yhteisresistanssi laskettua seuraavalla tavalla:

$$1{,}191\,\mathrm{V} = 4{,}66\,\mathrm{V} \cdot \frac{(\mathrm{R1} + \mathrm{R33})}{(\mathrm{R1} + \mathrm{R33}) + 10\,\mathrm{k\Omega}}$$

$$(\mathrm{R1} + \mathrm{R33}) = \frac{1{,}191\,\mathrm{V}}{4{,}66\mathrm{V} - 1{,}191\,\mathrm{V}} \cdot 10\,\mathrm{k\Omega} = 3433\,\Omega$$

Vastukseksi R33 valittiin kiinteä 2,2 kΩ vastus ja vastukseksi R1 valittiin trimmeripotentiometri. Ensimmäisen vahvistinasteen ei-invertoivan sisäänmenon jännite voidaan nyt säätää täsmälleen oikeaksi trimmeripotentiometrillä R1.

Jälkimmäinen vahvistinaste vahvistaa lämpötila-anturin jännitteen. A/D-muuntimen sisäänmenon jännitteen maksimiarvo on 4,66 volttia ja lämpötila-alueeksi määritettiin -35 °C…+35 °C. Kun lisäksi tiedetään, että lämpötila-anturin ulostulojännite suurenee 10mV/°C, saadaan sopiva arvo jännitevahvistukselle A_U laskettua seuraavalla tavalla:

$$A_U = \frac{4{,}66\,\mathrm{V}}{(35 + 35) \cdot 0{,}01\,\mathrm{V}} = 6{,}65$$

Jos kytkentään haluttaisiin tarkalleen lasketun suuruinen vahvistus, pitäisi takaisinkytkentävastuksen olla säädettävä. Vahvistus päätettiin pyöristää seuraavaan suurempaan arvoon. Kiinteistä vastuksista löytyy arvot 10 kΩ ja 68 kΩ, joilla vahvistukseksi tulee:

$$\frac{68\,\mathrm{k\Omega}}{10\,\mathrm{k\Omega}} = 6{,}8$$

Vahvistukseksi valittiin 6,8, joka määräytyy kytkennän vastuksilla R35 ja R36. Pyöristyksen johdosta mitta-alueen yläraja jää hieman alle +35 °C.

Jälkimmäisen vahvistinasteen käyttöjännitteeksi asetettiin 5 V, jotta A/D-muuntimen sisäänmenojännite ei koskaan ylittyisi. Ulostulojännitteen todettiin testimittausten perusteella nousevan korkeintaan 4,5 volttiin. Tämän johdosta mitta-alueen yläraja laski jälleen hieman. Kytkennällä mitattavissa oleva maksimilämpötila on siis:

$$T_{max} = \frac{4{,}5\,\mathrm{V}}{0{,}068} - 35\,^{\circ}\mathrm{C} = 31^{\circ}\mathrm{C}$$

Jälkimmäisen vahvistinasteen lähdöstä kytkettiin diodi D5 maatasoon. Sen tarkoituksena on rajoittaa lähdön negatiivinen jännite diodin kynnysjännitteen suuruiseen arvoon. Diodin kynnysjännite on valmistajan datalehden mukaan enintään 0,5 volttia, kun virta on 30 mA. (BAT85 datalehti 2000, 3)

Lähtöön pääsee negatiivinen jännite, mikäli lämpötila on vähemmän kuin -35 °C, tai jos potentiometrit R1 tai R2 on säädetty väärin. Lähtöön ei siis pitäisi päästä negatiivista jännitettä sen jälkeen, kun mittarin alkukalibroinnit on tehty ja mittaria käytetään lämpötila-alueella -35°C…+35 °C.

5.5 Käyttöjännitteen +5V stabilointi

Mikrokontrolleri ja LCD-näyttömoduuli toimivat +5 V jännitteellä, joka saadaan L7805ABD2T-regulattorilla akkujännitteestä. Lähtöjännitteen minimiarvo on 4,8 volttia ja maksimiarvo on 5,2 volttia, kun tulojännite vaihtelee 7,5 voltista 20 volttiin ja virta vaihtelee 5 milliampeerista yhteen ampeeriin.
(L7805ABD2T datalehti 2008, 7)

Käytännössä lähtöjännitteen vaihteluväli on kuitenkin paljon pienempi. Tämä johtuu siitä, että sisääntulojännite ja kuormituksen virta vaihtelevat kytkennässä huomattavasti vähemmän. Toisaalta, vaikka ulostulon jännite muuttuisi 4,8 voltista 5,2 volttiin, se ei haittaisi, sillä mikrokontrollerin tai LCD-näyttömoduulin käyttöjänniterajat eivät ylittyisi.

Regulaattorikytkentä tehtiin datalehden mallin mukaisesti. Kytkentä on liitteen 1 piirikaavion kohdassa 2A.

5.6 Referenssijännitteen stabilointi

Mikrokontrollerin A/D-muuntimen käyttämä referenssijännite vaikuttaa suuresti A/D-muunnoksen tuloksen tarkkuuteen. Tämä taas vaikuttaa suoraan koko ampeerituntimittarin mittaustarkkuuteen. Muita jännitteitä, joiden pitäisi pysyä mahdollisimman vakiona, ovat operaatiovahvistinkytkentöjen offset-jännitteet. Näiden jännitteiden muutokset vaikuttavat myös suoraan koko mittarin mittaustarkkuuteen.

Ampeerituntimittariin päätettiin laittaa erillinen regulaattori näitä kriittisiä jännitteitä varten. Regulaattoriksi valittiin LM723, jonka lähtöjännite on asetettavissa ulkoisilla vastuksilla 2 voltista 37 volttiin. Tulojännitteen maksimiarvo on 40 V ja sen pitää olla vähintään 9,5 V.
(LM723 datalehti 2007, 5-7)

Akun napajännite saattaa laskea melko paljon suurilla kuormituksilla. Tästä syystä piirin tulojännitteen miniarvo on syytä huomioida. Tässä tapauksessa 9,5 V katsottiin

kuitenkin riittäväksi, sillä akkujännite tuskin laskee näin alas käytettävillä kuormituksilla.

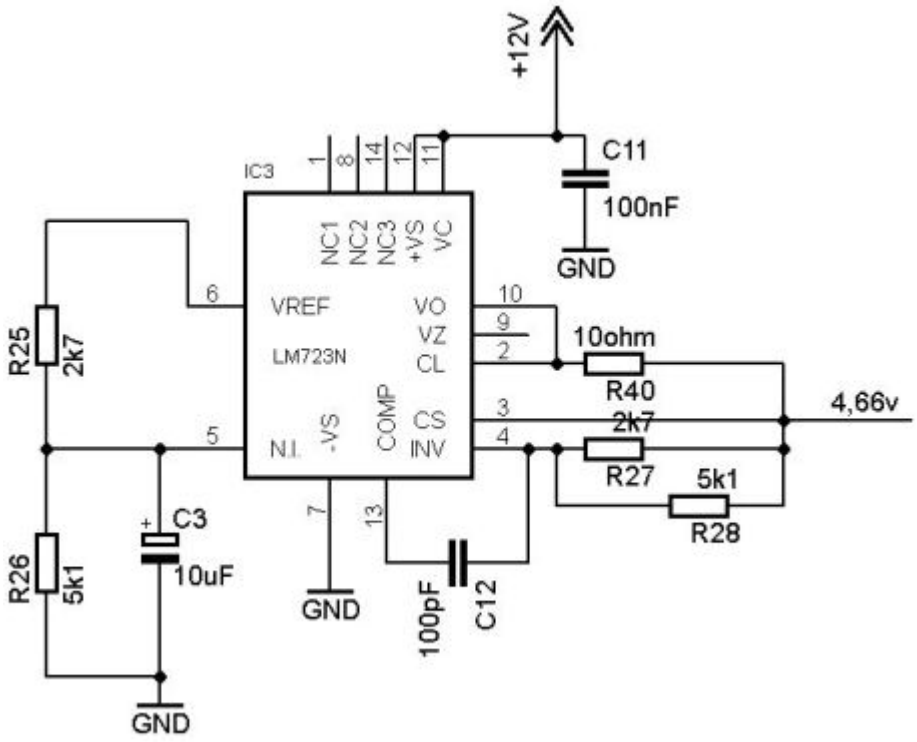
Syöttöjänniteregulointi on valmistajan datalehden mukaan tyypillisesti 0,01 % ja enintään 0,1 %. Tämä tarkoittaa sitä, että tulojännitteen vaihtelun ollessa 3,3 V, lähdön jännitevaihtelu on 0,0033 V. Tulojännitteen vaihtelu ei voisi olla enempää, sillä:

$$12{,}8\ \text{V} - 9{,}5\ \text{V} = 3{,}3\ \text{V}$$

Kaavassa 12,8 V on akun napajännite, kun akku on täynnä. Ulostulon jännitevaihtelu on siis enimmillään 0,003 V ja tyypillisesti 0,0003 V. Kuormareguloinnin vaikutusta ei laskussa huomioitu, sillä kuormavirta oletettiin vakioksi. Regulaattorin kuormana on operaatiovahvistimien ei-invertoivat sisäänmenot jännitejaon kautta sekä mikrokontrollerin referenssijännitteen sisäänmeno. Virtojen määrän vaihtelu on siis todella pientä. Tämä oli yksi syy, miksi kytkennässä haluttiin käyttää kahta eri regulaattoria: Mikroprosessorin ja LCD-näyttömoduulin ottamat virrat muuttuvat jatkuvasti.

Tärkeä ominaisuus tässä sovelluksessa on myös regulaattorin lämpötilastabiilius. Datalehden mukaan lähtöjännitteen lämpötilaryömintä on enintään $150\ \text{ppm}/^{\circ}\text{C}$. Tämä on erittäin vähän, joten jännitteen voidaan ajatella pysyvän vakiona lämpötilasta riippumatta.

Kytkentä tehtiin valmistajan datalehden esimerkkikytkennän mukaisesti. Datalehdessä oli taulukoitu valmiiksi vastusten arvoja erilaisille lähtöjännitteille. 5 V jännitteelle vastusten arvot ovat $2{,}15\ \text{k}\Omega$ ja $5{,}11\ \text{k}\Omega$. Jännitettä haluttiin kuitenkin hieman laskea, joten arvoiksi vaihdettiin $2{,}7\ \text{k}\Omega$ ja $5{,}1\ \text{k}\Omega$, jolloin lähtöjännite on 4,66V. Valmis kytkentä on kuvassa 19.



Kuva 19 - LM723-piirin kytkentä

Kuvan 19 kytkennässä nastasta 4 lähtöön kytkettävän vastuksen kokonaisresistanssin suuruus pitää olla vastusten R25 ja R27 rinnankytkennän suuruinen. Tämän vastuksen tarkoituksena on minimoida lämpötilaryömintä. Kytkennässä käytettiin kahta vastusta, jotta päästiin mahdollisimman lähelle haluttua arvoa.

Vastus R40 rajoittaa lähtövirran arvoon n. 65 mA. (LM723 datalehti 2007, 9) Lähtövirta on normaalisti joitain milliampeereita, joten rajoituksesta on hyötyä lähinnä vikatilanteessa.

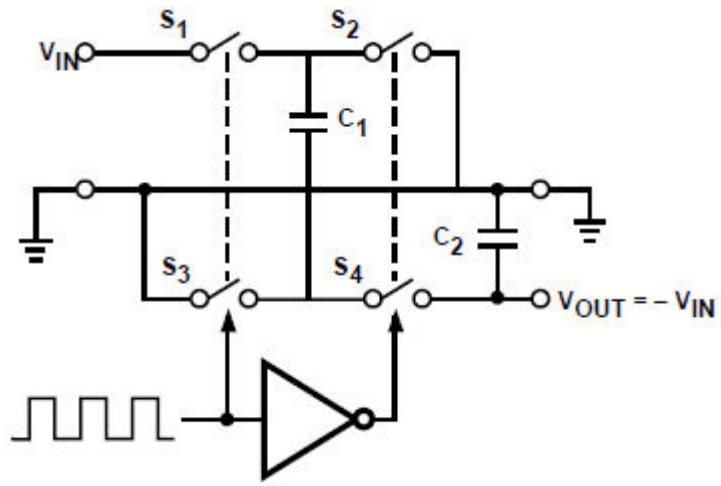
5.7 Negatiivisen jännitteen muodostaminen

Jotta mittarin maatasoa ei tarvinnut jättää kelluvaksi, päätettiin kytkennän operaatiovahvistimia käyttää kaksipuolisella käyttöjännitteellä. Positiivisen käyttöjännitteensä operaatiovahvistimet OP1-3 saavat suoraan akkujännitteestä sulakkeen, suotokondensaattorin C25 ja kytkintransistorin Q2 jälkeen. Negatiivinen jännite muodostetaan kondensaattoripumpulla, joka koostuu kahdesta elektrolyyttikondensaattorista sekä Microchipin TC7662B-piiristä.

Tähän ratkaisuun päädyttiin siitä syystä, että operaatiovahvistinkytkennän kuluttama virta on hyvin pieni. Kondensaattoripumpun kytkentä on myös hyvin yksinkertainen ja osat halpoja.

5.7.1 Kondensaattoripumpun toimintaperiaate

Kuvassa 20 on ideaalisen kondensaattoripumpun toiminta kytkimillä kuvattuna. Kytkentä toimii siten, että ensin kytkimet S1 ja S3 ovat suljettuna ja kytkimet S2 ja S4 ovat auki. Kondensaattori C1 varautuu jännitteeseen Vin. Seuraavaksi kytkimet S1 ja S3 aukeavat ja kytkimet S2 ja S4 sulkeutuvat. Nyt kondensaattorin positiivisesti varautunut pää kytkeytyy maapotentiaaliin ja negatiivisesti varautunut pää kytkeytyy ulostuloon. Näin ulostuloon on siirtynyt sisäänmenon jännite negatiivisena. Kytkinten asentojen vaihto toistuu korkealla taajuudella. (TC7662B datalehti 1996, 3)



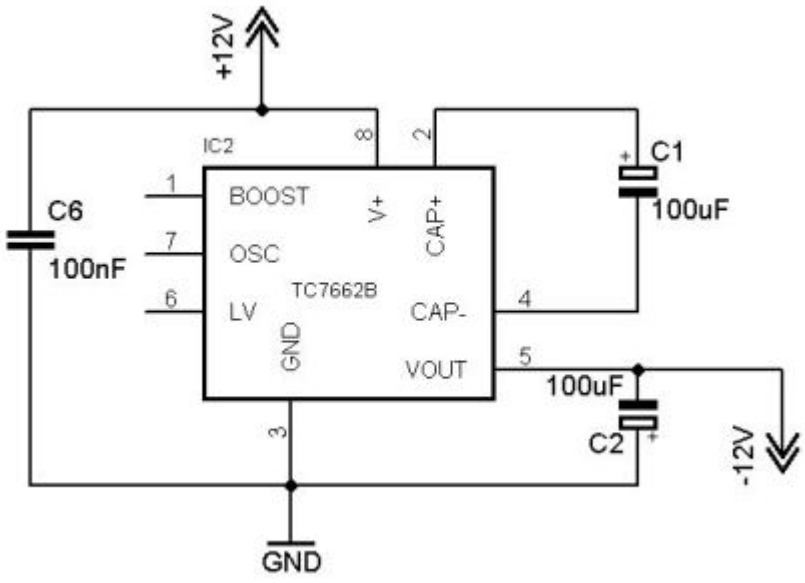
Kuva 20 - Negatiivisen jännitteen muodostaminen kondensaattoripumpulla (TC7662B datalehti 1996, 3)

TC7662B-piirissä kytkimet S1 - 4 ovat MOS-tehokytkimiä, joissa tapahtuu hieman tehohäviöitä. Piirissä on myös oskillaattori, sisäinen jänniteregulaattori, jakaja, logiikkaverkko sekä jännitetason muunnin. Logiikkaverkon ja jännitetason muuntimen tarkoituksena on ylläpitää kytkimillä oikeanlaisia estojännitteitä, jotta piiri toimisi moitteettomasti myös käynnistysvaiheessa sekä lähdön oikosulkutilanteessa. (TC7662B datalehti 1996, 1-3)

5.7.2 TC7662B-piirin kytkentä oheiskomponentteineen

TC7662B-piiriä voidaan käyttää myös muihin tarkoituksiin jännitteen käännön lisäksi. Valmistajan datalehdessä on useita esimerkkikytkentöjä. Ampeerituntimittarissa käytetty kytkentä on suoraan datalehdestä sillä erotuksella, että kondensaattorien arvot ovat 100 µF. Kondensaattorien kapasitanssia suurentamalla vähennettiin lähtöjännitteen vaihtelua.

Käytetyssä kytkennässä varausta siirretään kondensaattorien välillä 10 kHz taajuudella ja sisäinen regulaattori on toiminnassa. Regulaattori pitää kytkeä pois päältä, mikäli käytetään alle 3,5 voltin jännitteitä, sillä regulaattorin aiheuttama jännitehäviö on liian suuri käytettävään jännitteeseen verrattuna. Regulaattori saadaan pois päältä, kun LV-nasta kytketään maatasoon. Kuvassa 21 on mittarissa käytetty negatiivisen jännitteen muodostava kytkentä. (TC7662B datalehti 1996, 3)



Kuva 21 - Kytkentä negatiivisen jännitteen muodostamiseksi

5.7.3 Negatiivisen jännitteen jännitevaihtelu

Kytkennästä saatavan negatiivisen jännitteen jännitevaihteluun pystytään vaikuttamaan komponenttivalinnoilla. Jännitevaihtelu on sitä pienempää, mitä pienempi on käytettyjen kondensaattorien ESR (efektiivinen sarjaresistanssi). Vastaavasti vaihtelu on sitä pienempää, mitä suurempi kapasitanssi käytetyissä kondensaattoreissa on. Käytetyissä 100 μF kondensaattoreissa ESR on alle 1 Ω.
(TC7662B datalehti 1996, 3)

Jännitevaihtelun määrään vaikuttavat myös lähtövirran suuruus ja TC7662B-piirin oskillaattorin taajuus. Taajuutta ei kuitenkaan saa lisätä liikaa, sillä jossain vaiheessa taajuus on liian suuri kondensaattorin kapasitanssiin nähden ja kondensaattori ei ehdi enää varautua täysin. (TC7662B datalehti 1996, 3)

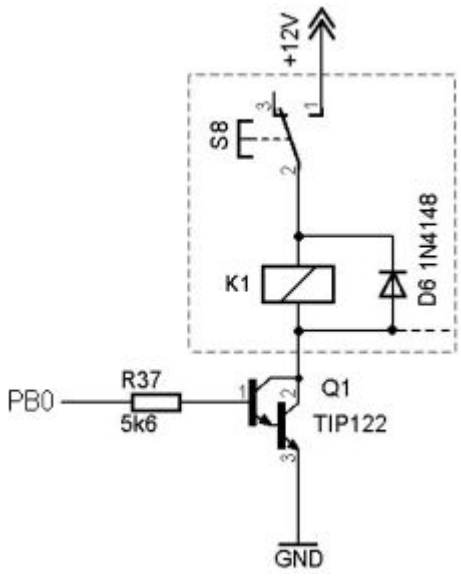
Kytkennän kuormana toimiva operaatiovahvistinkytkentä toimii pienellä virralla, joka on milliampeeriluokkaa. Negatiivista jännitettä mitattiin valmiissa kytkennässä. Syöttöjännitteen ollessa 12,48 V, piirin lähtöjännite on -12,41 V. Jännitevaihtelu on n.10 mV.

5.8 Avokollektorilähtö

Ampeerituntimittarilla voidaan ohjata kesämökin valaistusta. Valot saadaan päälle, mikäli mittari on päällä. Valot sammuvat, mikäli mittari sammutetaan tai asetettu ampeerituntimäärä ylittyy.

Tämä on toteutettu kytkennän osalta siten, että mikrokontrollerin lähtö PB0 ohjaa transistoria Q1. Transistorin emitteri on maapotentiaalissa ja kollektori on kytketty lähtöliittimen JP1 nastaan 1, josta edelleen ohjausreleille.

Mökin valaistuksen ohjausta varten sähkökeskuksessa on 10 relettä. Jokaisen releen kela on kytketty kuvan 22 mukaisesti.



Kuva 22 - Mökin valaistuksen ohjausreleiden kytkentä

Jokaisen releen kelan läpi kulkee 50 milliampeerin virta. Käytettävän darlingtontransistorin TIP122 virtavahvistus on 1000, suurin sallittu kollektorivirta 5A ja kanta-emitterijännite 1,4 V. (TIP122 datalehti 2008, 3-4)

Kun mikrokontrollerin lähtöportin jännite 1-tilassa on datalehden mukaan 4,2 V virran ollessa 2 mA, saadaan transistorin kantavastuksen resistanssin maksimiarvoksi:

$$\mathrm{R37} = \frac{4{,}2\ \mathrm{V} - 1{,}4\ \mathrm{V}}{\left(\frac{500\ \mathrm{mA}}{1000}\right)} = 5{,}6\ \mathrm{k\Omega}$$

6 PIIRILEVYN SUUNNITTELU

Piirikaavio ja piirilevy suunniteltiin Eagle-ohjelman Light Edition-versiolla, jonka voi ladata ilmaiseksi CadSoftin internetsivuilta. Kyseistä versiota ei saa käyttää kaupallisiin suunnitteluprojekteihin.

Ilmaisversio on täysin samanlainen täysversion kanssa, lukuun ottamatta seuraavia rajoituksia: Piirilevyn koko voi olla enintään 100 mm*80 mm, piirilevyllä voi olla enintään kaksi kerrosta ja piirikaavio voi olla vain yhden arkin kokoinen. Arkin koko tosin voidaan määritellä suureksikin. Ampeerituntimittarin piirikaavio piirrettiin A3-kokoiselle arkille.

Ampeerituntimittaria varten tehtiin Eaglella oma komponenttikirjasto, johon kerättiin työssä tarvittavat komponentit. Suurin osa komponenteista saatiin tehtyä yhdistelemällä komponenttikirjastosta sopivia symboleita sopiviin koteloihin. Vain muutama komponentti piirrettiin kokonaan itse.

6.1 Kotelointi

Piirilevyn kooksi valittiin suurin mahdollinen ja osat saatiin juuri sopivasti mahtumaan levylle. LCD-näyttömoduuli sekä käyttöliittymän painonapit S1, S2 ja S4 sijoitettiin piirilevyn juotospuolelle. Näin saatiin enemmän tilaa piirilevyn komponenttipuolelle.

Piirilevy suunniteltiin siten, että se voidaan kiinnittää suoraan kotelon kanteen. Kanteen tehdään reiät näyttöä ja painikkeita varten. Koska muut komponentit ovat toisella puolella, piirilevy saadaan mahdollisimman lähelle kantta. Kun piirilevy on tarpeeksi lähellä kantta, näyttö ja painikkeet tulevat kotelosta hieman ulkopuolelle.

Koska painike S3 jätettiin piirilevyn komponenttipuolelle, se jää kotelon sisäpuolelle. Tämä tehtiin tarkoituksella, sillä silloin käyttäjä ei vahingossa pääse muuttamaan kalibrointiasetuksia. Mikäli mittari halutaan kalibroida, kotelo pitää avata.

6.2 Osasijoittelu ja häiriöiden minimointi

Piirilevyn osasijoittelukuva on liitteessä 2. Osasijoittelukuvan punaiset viivat ovat komponenttipuolen kuparivetoja ja siniset viivat ovat juotospuolen kuparivetoja. Juotospuolella sijaitsevat komponentit näkyvät osasijoittelukuvassa peilikuvana.

Mikrokontrolleri ja LCD-näyttömoduuli asetettiin mahdollisimman lähelle toisiaan ja etäälle operaatiovahvistinten kytkennöistä. Myös mikrokontrollerin ohjelmointiliitin asetettiin mahdollisimman lähelle mikrokontrolleria. Kaikki häiriönpoistoon tarkoitetut 100 nF kondensaattorit asetettiin mahdollisia häiriöpiikkejä aiheuttavien komponenttien lähelle.

Transistori Q1 asetettiin lähelle reunaa, sillä se säteilee hieman lämpöä. Myös lämpötila-anturi D1 asetettiin lähelle reunaa, jotta muiden komponenttien tuottama lämpö ei vääristäisi mittaustulosta.

Kaikki johdinten vedot pyrittiin pitämään mahdollisimman lyhyinä ja johdinten käännöskohdissa vältettiin teräviä kulmia. Piirikaavion suunnittelussa ei käytetty automaattista osasijoittelua eikä automaattista johdinten vetoa. Ylä- ja alatason tyhjäksi jääneet osat täytettiin maatasolla. Suurin osa piirilevyn ylä- ja alapuolesta on siis maatasoa.

Syöttökaapelin päistä tulevat johtimet voidaan liittää joko liittimeen JP2 tai juottaa suoraan piirilevyyn liittimen viereen. Mikäli johdot kytketään liittimeen JP2, saattaa liitoksen resistanssi muuttua ajan myötä. Juotetussa liitoksessa resistanssi muuttuu vähemmän.

Analogisten signaalien reitit pyrittiin pitämään mahdollisimman etäällä digitaalisten signaalien reiteistä.

7 LÄMPÖTILAMUUTOKSEN VAIKUTUS VIRTAMITTAUKSEN TULOKSEEN

Syöttökaapelin resistanssi vaikuttaa syöttökaapelin yli vaikuttavan jännitehäviön suuruuteen. Koska syöttökaapeli on kuparia, sen resistanssi muuttuu lämpötilan vaikutuksesta.

Kuparin resistiivisyys eli ominaisvastus muuttuu lämpötilan muuttuessa kaavan 3 mukaisesti.

$$\rho = \rho_0 \cdot (1 + \alpha_R \cdot (T - T_0)) \qquad (3)$$

Kaavassa 3 ρ = resistiivisyys, ρ_0 = resistiivisyys lämpötilassa T_0, α_R = resistiivisyyden lämpötilakerroin, T = Loppulämpötila ja T_0 = alkulämpötila. Kuparin lämpötilakerroin on 0,00039 $\frac{1}{K}$ ja resistiivisyys on +20 °C lämpötilassa $17{,}2 \cdot 10^{-9}\,\Omega m$.
(Mäkelä ym. 2000, 120;177)

Yhdistämällä kaava 3 kaavaan 2, saadaan tulokseksi kaava 4, jolla voidaan laskea johtimen resistanssi eri lämpötiloissa.

$$R = \rho_0 \cdot (1 + \alpha_R \cdot (T - T_0)) \cdot \frac{1}{A} \qquad (4)$$

Kaavaa 4 tarkastellessa havaitaan, että erona kaavaan 2 on sulkujen sisällä oleva lämpötilan mukaan muuttuva kerroin. Siis mikäli tunnetaan alkuperäinen resistanssi ja lämpötila, saadaan resistanssi laskettua jossain muussa lämpötilassa. Kerroin vaikuttaa ohmin lain mukaisesti myös jännitteen ja virran arvoon.

Mikäli syöttökaapelin läpi kulkeva virta on 25 A ja ympäröivä lämpötila on 20 °C matalampi kuin mittarin kalibrointilämpötila, se näyttäisi arvoa:

$$I = \left(1 + 0{,}0039 \frac{1}{K} \cdot (-20)\right) \cdot 25\,A = 23{,}05\,A$$

Lämpötilamuutoksen aiheuttama mittausvirhe on siis huomattava.

Mittausvirhe korjataan mikrokontrollerin ohjelmassa siten, että mitatun virran arvo kerrotaan seuraavalla kertoimella, jossa T = lämpötila mittaushetkellä ja T_0 = lämpötila, jossa mittari on kalibroitu:

$$I = \frac{1}{\left(1 + 0{,}0039\frac{1}{K}\cdot\left(T - T_0\right)\right)}\cdot I_{mitattu}$$

8 OHJELMA

Mikrokontrollerin ohjelma tehtiin C-kielellä ja kääntämiseen käytettiin AVR-GCC -kääntäjää. C-kielinen lähdekoodi on kommentteineen liitteessä 3. Koko ohjelma on yhdessä lähdetiedostossa main.c

Ohjelmaa suunniteltaessa tavoitteena oli toimivan ja selkeän koodin tuottaminen asetetuissa aikarajoissa. Tehokkuutta voidaan parantaa myöhemmissä kehitysversioissa. Lisäksi ohjelma tehtiin sellaiseksi, että ampeerituntimittarin käyttäminen olisi mahdollisimman helppoa.

8.1 Pääohjelma

Pääohjelman alussa määritellään I/O-porttien asetukset, haetaan tarvittavat tiedot EEPROM-muistista ja tehdään tarvittavat alustukset.

Pääohjelma sisältää pääohjelmasilmukan, jonka sisältöä suoritetaan 10 ms välein niin kauan, että painiketta S2 pidetään pohjassa 6 sekuntia, tai asetettu Ah-raja ylittyy. Kun silmukasta poistutaan, tallennetaan kertyneet ampeeritunnit, Ah-raja ja kalibrointiparametrit EEPROM-muistiin ja mittari sammutetaan asettamalla lähtö PB0 nollaksi.

Pääohjelmasilmukan ajoitus toimii siten, että timer1-keskeytys laukeaa 10 ms välein ja keskeytysohjelmassa asetetaan lippu_10ms-muuttuja 1-tilaan. Silmukan suorituksen ehtona on, että merkkilippu on 1-tilassa. Pääohjelmasilmukassa luetaan painikkeiden tiloja ja päivitetään näyttö kunkin tilanteen mukaiseksi. Näytön ohjaukseen liittyvät toiminnot jaettiin aliohjelmiksi, joita kutsutaan pääohjelmasta. Näyttö päivitetään 200 ms välein.

Painikkeiden asentoa luetaan 10 ms välein. Tällä keinolla vältetään kytkinten painamisesta aiheutuvien häiriöpiikkien aiheuttamat virhetilat. Lisäksi pystytään toteamaan, onko painiketta juuri painettu, vai pidetäänkö sitä pohjassa. Painikkeiden S1 ja S2 painamisen kesto vaikuttaa suoritettavaan toimintoon.

8.2 Aliohjelmat

Kaikille aliohjelmille kirjoitettiin lähdekoodiin kommentteina oma tunnistekenttänsä, joka sisältää aliohjelman kuvauksen, parametrit ja paluuarvon. Aliohjelmat käyttävät joko globaaleja muuttujia tai omia paikallisia muuttujiaan. Ohjelmaa voitaisiin kehittää siten, että aliohjelmat käyttäisivät vain globaaleja muuttujia, jolloin suoritusnopeus olisi suurempi.

Tallenna_uusi_raja-niminen aliohjelma muuttaa käyttäjän asettaman Ah-rajan ASCII-koodista numeroksi ja tallentaa sen ah_raja-muuttujaan. Kaikki muut aliohjelmat ovat LCD-näytön ohjaamista varten. Jokaiselle LCD-näytölle tulostettavalle suureelle tehtiin oma aliohjelmansa.

LCD_kirjoita_8bittia-aliohjelmaa käytetään eniten, sillä sitä kutsutaan muista aliohjelmista sekä pääohjelmasta. Se kirjoittaa näytölle yhden merkin tai käskyrekisteriin yhden käskyn 4-bittisen dataväylän kautta. 8-bittisestä datasta siirretään ensin 4 eniten merkitsevää bittiä, jonka jälkeen 4 vähiten merkitsevää bittiä.

Aliohjelmissa käytetyt viiveet muodostetaan delay_basic-kirjastotiedoston sisältämällä _delay_loop_1 tai _delay_loop_2-funktiolla. Molempien viive-funktioiden toiminta perustuu asm-kielellä tehtyyn silmukkaan, jota käydään läpi parametrina annettu määrä. Yhden _delay_loop_1-silmukan suoritukseen kuluu kolme kellojaksoa ja yhden _delay_loop_2-silmukan suoritukseen kuluu neljä kellojaksoa. Lisäksi aikaa kuluu itse silmukkaan siirtymiseen ja siitä poistumiseen. Kokonaisviiveen laskeminen tarkasti on siksi hankalaa. Aliohjelmien viiveiden tarkkuudella ei ole suurta merkitystä.

Numerot muutetaan liukuluvuista ASCII-koodiksi stdio-kirjastotiedoston sisältämällä sprintf-funktiolla. Funktion parametrina asetetaan, monenko desimaalin tarkkuudella luku halutaan esittää. Parametrina välitetään lisäksi muutettava liukuluku sekä taulukon alkuosoite, jonne tulos tallennetaan.

8.3 Timer1-keskeytys

Timer1-keskeytyksellä ohjataan pääohjelman ajoituksen lisäksi A/D-muunnoksen ajoitusta. Kun keskeytysohjelmaa suoritetaan kymmenettä kertaa, aikaa on kulunut 100 ms. Tällöin virtamittauksen arvo kerrotaan 0,1 sekunnilla, tulos muutetaan ampeerisekunneista ampeeritunneiksi ja tallennetaan. Seuraavaksi aloitetaan uusi virtamittaus A/D-muuntimella.

Kertyneet ampeeritunnit tallennetaan muuttujaan Ah_vajaat. Mikäli muuttujan arvo on yksi tai enemmän, tuloksesta vähennetään yksi ja Ah_kokonaiset-muuttujan arvoa kasvatetaan yhdellä. Kertyneiden ampeerituntien kokonaismäärä on siis jaettu kahteen muuttujaan. Mikäli tulos tallennettaisiin vain yhteen muuttujaan, mittarin mittaustarkkuus olisi huonompi. Tämä johtuu desimaalipilkun paikan siirtymisestä liukuluvussa.

255 keskeytyksen välein aikaa on kulunut 2,55 sekuntia. Virtamittausta ei suoriteta, sillä se on tehty 50 ms sitten. Tällöin edelliseen virtamittaukseen käytetty kanava tallennetaan ja aloitetaan lämpötilamittaus. Lämpötilamittauksen jälkeen saatu tulos tallennetaan ja kanavaksi vaihdetaan ennen mittausta käytössä ollut kanava. Koska mittausalue määräytyy virran suuruuden mukaan, kanava täytyy säilyttää samana.

8.4 A/D-muunnos

Osana valmistajan suosittamia toimia kohinan vähentämiseksi A/D-muunnos suoritetaan, kun CPU on pysähtyneenä. Muunnos käynnistyy sleep-käskyn jälkeen. Muunnoksen valmistuttua CPU ja A/D-muunnoksen keskeytysohjelma käynnistyy. Ensimmäisellä kerralla muunnos kestää 400 μs ja normaalisti 208 μs . Pääohjelmaa ehditään siis suorittaa pitkä aika ennen seuraavaa A/D-muunnosta.

Muunnostulos tallennetaan keskeytysohjelmassa. Oikea muuttuja valitaan muunnoskanavan perusteella. Mikäli mittaus tehdään kanavasta PC0 tai PC2, tulos kerrotaan vielä lämpötilamuutoksen aiheuttaman mittausvirheen korjauskertoimella. Keskeytysohjelmassa vaihdetaan myös virran mittauskanavaa mittausalueen mukaan.

Muunnostulos on 10-bittinen, josta käytetyllä referenssijännitteellä ja muunnosnopeudella virhe on 2 vähiten merkitsevää bittiä. A/D-muunnoksen aiheuttama virhe virtamittaukseen 25 A:n mittausalueella on:

$$I_{virhe} = 25\ \text{A} - \frac{25\ \text{A} \cdot 1111111100_b}{1111111111_b} = 0{,}07\ \text{A}$$

100 A:n mittausalueella virhe on vastaavasti:

$$I_{virhe} = 100\ \text{A} - \frac{100\ \text{A} \cdot 1111111100_b}{1111111111_b} = 0{,}29\ \text{A}$$

Kohina ja häiriöt vaikuttavat A/D-muunnoksen tulokseen, joten myös piirilevyn suunnittelulla on suuri merkitys koko mittarin tarkkuuteen. Kokeilun perusteella kokonaisvirhe on vain hieman enemmän, kuin 2 vähiten merkitsevää bittiä.

9 YHTEENVETO

Työ oli haastava ja kartutti tietämystä sekä digitaali- että analogiaelektroniikasta. Asetetut tavoitteet saavutettiin, vaikka aikaa kului reilusti alkuperäistä suunnitelmaa enemmän. Se selittyy pääosin sillä, että mittariin lisättiin projektin edetessä joitain ominaisuuksia ja toimintoja. Testausvaiheeseen ja ohjelman viimeistelyyn kului myös yllättävän paljon aikaa.

Useat piirilevyn virheet johtuivat huolimattomuusvirheistä piirikaaviossa. Jostain puuttui vetoja tai oli väärässä paikassa. Jännitesyöttö operaatiovahvistimen OP2 offset-jännitteen säätöön otettiin ensin akkujännitteestä. Koska akkujännite muuttuu, myös offset-jännite muuttui. Jännitesyötöksi vaihdettiin 4,66 V stabiloitu jännite.

Virtaa mitattiin ensin liian pienellä mittausvälillä. Kun mittausväli on pieni, myös yhdellä mittauskerralla kertyvä ampeerituntimäärä on pieni. Tällöin ohjelmassa joudutaan käsittelemään todella pieniä lukuja ja lukujen tarkkuus huononee. Mittausväliksi vaihdettiin 100 ms ja kertyneet ampeeritunnit jaettiin kahteen muuttujaan siten, että toisessa on yhtä pienemmät ja toisessa kokonaiset ampeeritunnit.

Lukujen käsittelyn tarkkuutta voitaisiin parantaa jakamalla kertyneet ampeeritunnit vielä useampaan muuttujaan tai kasvattamalla mittausväliä. Mitä vähemmän mitattavassa virrassa on vaihtelua, tai mitä pidempään mittarilla mitataan yhtäjaksoisesti, sitä pidemmäksi mittausväli kannattaa asettaa. Mikäli lukujen käsittelytarkkuus halutaan vielä tarkemmaksi, pitää mittariin vaihtaa mikrokontrolleri, joka kykenee käsittelemään tarkempia liukulukumuuttujia, tai muuttaa laskenta kokonaislukupohjaiseksi

Vaikka mittaustarkkuus on jo riittävä käyttökohteeseen, tulevissa kehitysversioissa voitaisiin keskittyä mittaustarkkuuden parantamiseen. Lisäksi trimmeripotentiometrit voitaisiin korvata kiinteillä vastuksilla, jolloin kalibrointi tehtäisiin täysin ohjelmallisesti. Tällä tavoin mittarin luotettavuutta saataisiin parannettua, sillä trimmeripotentiometrin resistanssi muuttuu hieman ajan myötä, jolloin kalibrointiasetukset muuttuvat. Uudessa versiossa kannattaisi käyttää ulkoista A/D-muunninta ja vielä tarkempaa referenssijännitettä mittaustarkkuuden parantamiseksi.

LÄHTEET

ATmega168 datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

Atmel. [www-sivu].[Viitattu 17.03.2009] Saatavissa:
http://www.atmel.com/

BAT85 datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.nxp.com/acrobat_download/datasheets/BAT85_4.pdf

Bebek. [www-sivu].[Viitattu 17.03.2009] Saatavissa:
http://www.bebek.fi

Berndt, D. Maintenance-Free Batteries. 3rd edition. Baldock, Hertfordshire, England: Research studies press ltd, 2003.

Digilent Inc. [www-sivu].[Viitattu 17.03.2009] Saatavissa:
http://www.digilentinc.com/

KS0066U datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.lcd-module.de/eng/pdf/zubehoer/ks0066.pdf

Kuni, Timo. PIC16F690-pohjainen lämpömittari ja tiedonkeräin. Tutkintotyö. Tampereen ammattikorkeakoulu. Tietokonetekniikan koulutusohjelma. Tampere, 2006.

L7805ABD2T datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.datasheetcatalog.org/datasheet/stmicroelectronics/2144.pdf

LM723 datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.national.com/ds/LM/LM723.pdf

Mäkelä, Mikko; Soininen, Lauri; Tuomola, Seppo; Öistämö, Juhani. Tekniikan kaavasto. 3. Painos. Tampere: Tammertekniikka, 2001.

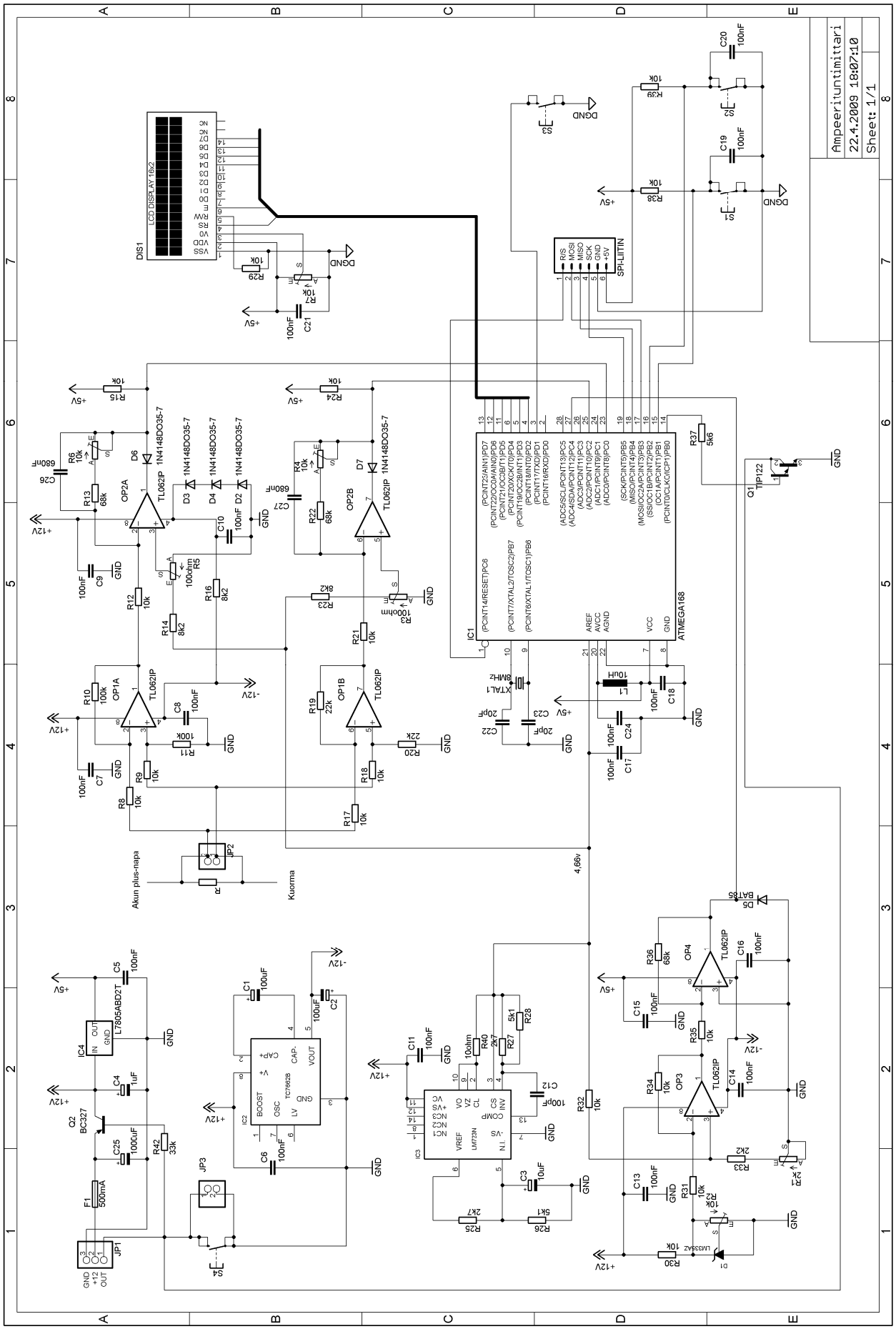
TC7662B datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://ww1.microchip.com/downloads/en/DeviceDoc/21469a.pdf

TIP122 datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.datasheetcatalog.org/datasheet/stmicroelectronics/4128.pdf

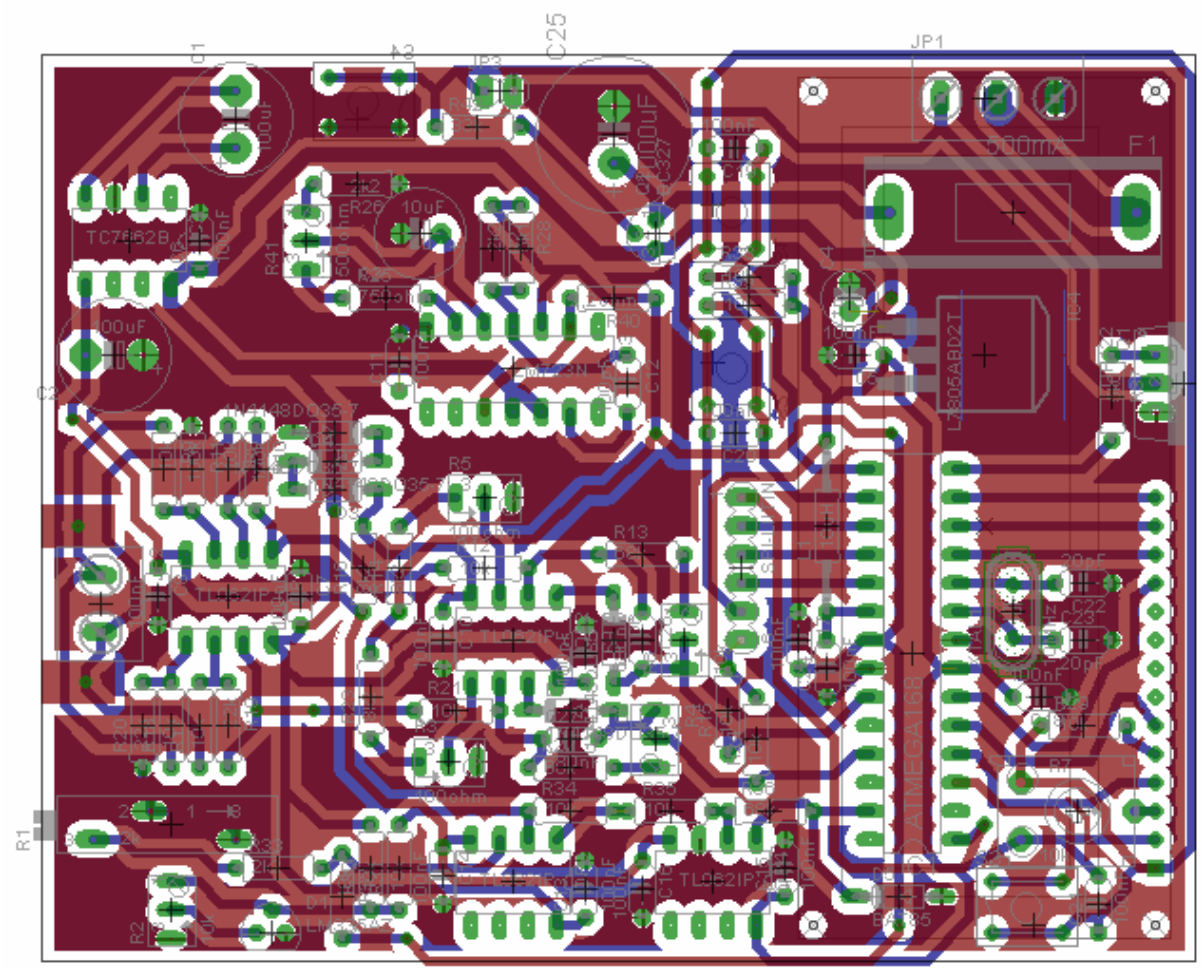
TL062 datalehti. [pdf-tiedosto]. [Viitattu 17.03.2009] Saatavissa:
http://www.datasheetcatalog.org/datasheet/stmicroelectronics/2294.pdf

LIITTEET

LIITE 1 - Piirikaavio



LIITE 2 - Osasijoittelukuva



LIITE 3 - Ohjelman lähdekoodi

```
/*_____________________________________________________________________
 * Tampereen ammattikorkeakoulu
 * Tietotekniikan koulutusohjelma
 * Sulautetut järjestelmät
 * Jukka Viinamäki
 *
 * Kääntäjä: AVR-GCC
 ************************Ampeerituntimittari**********************
 *_____________________________________________________________________*/


//    ** sisällytetään tarvittavat kirjastojen otsikkotiedostot  **

#include<avr/io.h>
#include<inttypes.h>
#include<util/delay_basic.h>          //sisältää viive-funktiot
#include<avr/interrupt.h>             //sisältää ADC- ja Timer1-keskeytykset
#include<avr/sleep.h>                 //sisältää virransäästötilojen asettamiseen
#include<stdint.h>                    //tarvittavat rutiinit
#include<stdio.h>                     //sisältää sprintf-funktion
#include<math.h>                      //sisältää round-funktion
#include<avr/eeprom.h>                //sisältää eepromin käsittelyyn tarvittavat
                                      //rutiinit

//    ** globaalien muuttujien määrittelyt **

volatile float virta0_100A=0;         //mitatun virran arvo
volatile float Ah_vajaat=0;           //kertyneet ampeeritunnit. Tänne
                                      //tallennetaan yhtä Ah:ta pienemmät arvot
volatile unsigned int Ah_kokonaiset=8; //kertyneet ampeeritunnit. Tänne
                                      //tallennetaan kokonaiset ampeeritunnit
volatile float lampotila=0;           //lämpötilan arvo
volatile float T_kalib = 22.0;        //kalibrointilämpötila
volatile float T_kerroin = 65.529;    //parametri, jolla vaikutetaan
                                      //lämpötilamittauksen tulokseen
volatile unsigned int  Ah_raja = 60;  //raja-arvo, jolla lähtö asetetaan nollaksi ja
                                      //mittari sammutetaan
volatile unsigned char lippu_10ms=0;  //lippu asetetaan ykköseksi 10 ms välein
volatile unsigned char lippu_100ms=0; //lippu asetetaan ykköseksi 100 ms välein
volatile unsigned char lippu_2s55=0;  //lippu asetetaan ykköseksi 2550 ms välein
volatile unsigned char kursorin_sij=0x8B;//kursorin sijainti LCD-näytöllä
volatile unsigned char ADC_edellinen; //A/D-muuntimen kanava, jota käytettiin
                                      //edellisellä muunnoskerralla
volatile unsigned char kaynnissa=1;   //pääohjelmasilmukan ehto-muuttuja. Kun tämä
                                      //menee nollaksi, pääohjelman suoritus loppuu

unsigned char Ah_uusi_raja[4]="000";  //uuden raja-arvon asetus
unsigned char ekarivi[]="C Raja:   Ah"; //ensimmäisen rivin teksti perusnäkymässä
unsigned char tokarivi[]="Ah Hetk.    A";//toisen rivin teksti perusnäkymässä
```

```
//   ** A/D-muunnoksen keskeytyspalvelurutiini **

ISR (ADC_vect)
{
   if((ADMUX<<4) == 0)              //mikäli A/D-muunnos tehtiin kanavalla PC0,
   {                                //tulos muutetaan ampeereiksi, saatu arvo
                                    //kerrotaan korjauskertoimella ja tulos
                                    //tallennetaan virta0_100A-muuttujaan

      virta0_100A=(((ADC*25.0)/1024)*(1/(1+0.0039*(lampotila-T_kalib))));

      if(virta0_100A > 23)          //mikäli tämän mitta-alueen raja ylittyy,
      {                             //vaihdetaan A/D-muuntimen kanavaksi PC2
         ADMUX &= 0xF0;
         ADMUX |= 0x02;
       }
   }
   else if(ADMUX & 0x02)            //mikäli AD-muunnos tehtiin kanavalla PC2,
   {                                //tulos muutetaan ampeereiksi, saatu arvo
                                    //kerrotaan korjauskertoimella ja tulos
                                    //tallennetaan virta0_100A-muuttujaan

      virta0_100A=(((ADC*100.0)/1024)*(1/(1+0.0039*(lampotila-T_kalib))));

      if(virta0_100A < 22)          //mikäli tämän mitta-alueen raja alittuu,
      {                             //vaihdetaan A/D-muuntimen kanavaksi PC2
         ADMUX &= 0xF0;
      }
   }

   if(ADMUX & 0x04)                 //mikäli A/D-muunnos tehtiin kanavalla PC4,
   {                                //tulos muutetaan celsiusasteiksi ja
                                    //tallennetaan lampotila-muuttujaan

      lampotila=(((ADC*T_kerroin)/1024)-35);
      ADMUX = ADC_edellinen;        //A/D-muuntimen kanavaksi vaihdetaan ennen
                                    //lämpötilamittausta käytössä ollut kanava

   }
}


//   ** Timer1 keskeytyspalvelurutiini **

SIGNAL(TIMER1_COMPA_vect)        //tämä suoritetaan 10 ms välein
{
   lippu_10ms = 1;               //asetetaan 10 ms lippu
   lippu_100ms++;                //kasvatetaan 100 ms lipun arvoa yhdellä
   lippu_2s55++;                 //kasvatetaan 2550 ms lipun arvoa yhdellä

   if(lippu_100ms > 9)           //mikäli aikaa on kulunut 10*10 ms=100 ms,
   {
      lippu_100ms=0;             //nollataan 100 ms lippu
```

```
                                        //kerrotaan mitattu virran arvo 0,1 sekunnilla
                                        //ja muutetaan tulos ampeeritunneiksi
                                        //tallennetaan saatu arvo Ah_vajaat-muuttujaan

      Ah_vajaat = (Ah_vajaat + ((virta0_100A * 0.1) / 3600));

      if(Ah_vajaat >= 1)                //mikäli Ah_vajaat-muuttujassa on yksi
      {                                 //kokonainen ampeeritunti
         Ah_kokonaiset++;               //kasvatetaan Ah_kokonaiset-muuttujan arvoa
                                        //yhdellä
         Ah_vajaat = Ah_vajaat - 1.0;   //ja vähennetään sama määrä, eli 1Ah
                                        //muuttujasta Ah_vajaat
      }
      sei();                            //keskeytesten sallinta
      sleep_mode();                     //sammutetaan muu toiminta ja aloitetaan
   }                                    //A/D-muunnos


   if(lippu_2s55 > 254)                 //mikäli aikaa on kulunut 255*10 ms=2,55 s,
   {
      lippu_2s55=0;                     //nollataan 2,55 s lippu
      ADC_edellinen = ADMUX;            //tallennetaan kanava, josta A/D-muunnosta on
      ADMUX &= 0xF0;                    //viimeksi tehty ja vaihdetaan kanavaksi PC4,
      ADMUX |= 0x04;                    //josta saadaan mitattua lämpötila
      sei();                            //keskeytysten sallinta
      sleep_mode();                     //sammutetaan muu toiminta ja aloitetaan
                                        //A/D-muunnos
   }
}


//   ** Aliohjelmien prototyypit **

void LCD_alustus(void);
void LCD_kirjoita_8bittia(unsigned char data, unsigned char rekisteri);
void LCD_kirjoita_teksti(unsigned char *teksti);
void LCD_alkutekstit(void);

void LCD_kirjoita_T(void);
void LCD_kirjoita_T_tarkka(void);
void LCD_kirjoita_Ah(void);
void LCD_kirjoita_I(void);
void LCD_kirjoita_T_kalib(void);
void LCD_kirjoita_Ah_raja(void);

void Tallenna_uusi_raja(unsigned char *tekstina);


//   ** Pääohjelma **

int main (void)
{
   //   ** I/O-porttien määrittelyt **
```

```
// I/O:t joita ei käytetä, määritellään sisäänmenoiksi ja
// asetetaan sisäisen vastuksen avulla 1-tilaan

//PORTB
DDRB = 0x01;                    //PB0 ohjaa transistoria Q1
PORTB = 0xF8;                   //PB1 on kytkimen S1 sisäänmeno
                                //PB2 on kytkimen S2 sisäänmeno
//PORTD
DDRD = 0xFC;                    //PD1 on kytkimen S3 sisäänmeno
PORTD = 0x03;                   //PD2 ohjaa LCD-näytön sisäänmenoa RS
                                //PD3 ohjaa LCD-näytön sisäänmenoa E
                                //PD4 ohjaa LCD-näytön sisäänmenoa D4
                                //PD5 ohjaa LCD-näytön sisäänmenoa D5
                                //PD6 ohjaa LCD-näytön sisäänmenoa D6
                                //PD7 ohjaa LCD-näytön sisäänmenoa D7
//PORTC
DDRC = 0x00;                    //PC0 on 25 A:n mittausalueen sisäänmeno
PORTC = 0x6A;                   //PC2 on 100 A:n mittausalueen sisäänmeno
                                //PC4 on lämpötilatiedon sisäänmeno

//    ** haetaan tiedot EEPROM-muistista **

unsigned int testi_2tavua=0;    //luodaan muuttujat muistin sisällön
float testi_4tavua=0;           //oikeellisuuden tarkistamiseksi

                                //mikäli EEPROM-muistista haettu arvo ei
                                //ole sillä arvoalueella kuin pitäisi, sitä
                                //ei tallenneta vaan muuttujan arvoksi jätetään
                                //oletusarvo, joka on määritetty alustuksessa

                                //haetaan arvo ulkoisesta muistista muuttujalle
                                //Ah_kokonaiset
testi_2tavua = eeprom_read_word((unsigned int*)0);
if((testi_2tavua >= 0) && (testi_2tavua <1000))
   Ah_kokonaiset = testi_2tavua;

                                //haetaan arvo ulkoisesta muistista muuttujalle
                                //Ah_raja
testi_2tavua = eeprom_read_word((unsigned int*)2);
if((testi_2tavua >= 0) && (testi_2tavua <1000))
   Ah_raja = testi_2tavua;

                                //haetaan arvo ulkoisesta muistista muuttujalle
                                //T_kalib
eeprom_read_block((void*)&testi_4tavua, (const void*)4, 4);
if((testi_4tavua >= 0.0) && (testi_4tavua <100.0))
   T_kalib = testi_4tavua;

                                //haetaan arvo ulkoisesta muistista muuttujalle
                                //T_kerroin
eeprom_read_block((void*)&testi_4tavua, (const void*)8, 4);
if((testi_4tavua >= 0.0) && (testi_4tavua <100.0))
```

```
   T_kerroin = testi_4tavua;

_delay_loop_2(60000);          //60ms viive (2*36000) lcd-moduulia varten
LCD_alustus();                 //näytön alustus
LCD_alkutekstit();             //kirjoitetaan näyttöön alkutekstit
LCD_kirjoita_Ah_raja();        //kirjoitetaan näyttöön Ah-raja


//   ** A/D-muuntimen määrittelyt **

DIDR0 = 0x15;                  //C-portin ADC0, 2 ja 4 -kanavien
                               //digitaalisisäänmeno pois päältä
ADMUX = 0x00;                  //sisäinen Vref pois päältä, käytetään ulkoista.
                               //ADC-kanavan valinta: kanava 0 käyttöön
ADCSRA = 0x8F;                 //ADC-keskeytyksen sallinta, ei automaattista
                               //liipaisua, 128 esijakaja, f=62,5 kHz
ADCSRB = 0x00;                 //free running mode, ei vaikutusta sillä liipaisu
                               //on pois päältä


//   ** Timer1-kellon määrittelyt **

TCCR1B = 0x0A;                 //Output Compare A, esijakaja = /8
OCR1A = 9999;                  //(1/(8 MHz/8))*10000=10 ms
TIMSK1 |= 1<<OCIE1A;           //sallitaan laskuri

PORTB |= 0x01;                 //asetetaan transistorin Q1 ohjaus ykköseksi.
                               //Nollautuu vasta, jos Ah-raja ylittyy tai mittari
                               //sammutetaan

set_sleep_mode(SLEEP_MODE_ADC);//asetetaan virransäästötilaksi ADC
                               //A/D-muunnos käynnistyy heti, kun
                               //annetaan sleep-käsky


//   ** pääohjelman muuttujien määrittelyt **

unsigned char aika_200ms=0;    //200 ms lippu
unsigned char aika_2s=0;       //2 s lippu
unsigned int  aika_6s=0;       //6 s lippu
unsigned char painettu_nap1=0; //painikkeen1 tila 10 ms sitten
unsigned char painettu_nap2=0; //painikkeen2 tila 10 ms sitten

unsigned char painettu_nap3=0; //kalibrointipainikkeen S3 tila 10 ms sitten
unsigned char kalib_teksti[]="Aseta T   ";//kalibrointivalikon teksti
unsigned char kalib_tila=1;    //kalibroinnin tilatieto
unsigned char asetustila=0;    //sisältää tiedon, onko menossa kalibrointi,
                               //raja-arvon asetus vai ollaanko perustilassa
unsigned char sammutus=0;      //muuttuu ykköseksi, jos painiketta 1 pidetään
                               //pohjassa. Sammutus on mahdollisesti
                               //menossa.
unsigned int vanha_Ah_raja=60; //sisältää uutta Ah-rajaa edeltäneen arvon
unsigned char raja_ylittynyt=0; //menee ykköseksi jos Ah-raja ylittyy
```

```
unsigned char vilkutus=0;          //menee ykköseksi jos Ah-raja on ylitetty
                                   //ja mittari on sammunut

float vanha_T_kalib=T_kalib;       //uutta kalibrointiarvoa edeltänyt arvo
float vanha_T_kerroin=T_kerroin;   //uutta kalibrointiarvoa edeltänyt arvo

if(Ah_kokonaiset >= Ah_raja)       //mikäli Ah-raja on ylittynyt, asetetaan
  raja_ylittynyt=1;                //raja_ylittynyt-muuttuja ykköseksi

sei();                             //keskeytysten sallinta


//   ** Pääohjelmasilmukka alkaa **

while(kaynnissa)                   //ohjelma suoritus
{                                  //loppuu vasta, jos Ah-raja-arvo ylittyy tai
                                   //painiketta S2 pidetään pohjassa 6sekuntia
  if(lippu_10ms == 1)
  {
      if(asetustila != 2)          //tähän if-rakenteeseen mennään vain, mikäli
      {                            //kalibrointitila ei ole päällä
          aika_200ms++;
          if(asetustila==1)
            aika_6s++;
          if(!(PINB & 0x04))       //mikäli painiketta S2 on painettu
          {
              if(!painettu_nap1)   //mikäli 10ms sitten painiketta ei oltu painettu
              {
                  asetustila=1;    //siirrytään raja-arvon asetustilaan
                  LCD_kirjoita_8bittia(0x0E,0); //kursori näkyviin
                  painettu_nap1=1;
                  aika_2s=0;
                  aika_6s=0;
                  sammutus=0;
                  switch(kursorin_sij)          //ehtona kursorin sijainti
                  {
                      case 0x8B:
                        kursorin_sij=0x8C;      //siirretään kursoria oikealle
                        break;

                      case 0x8C:
                        kursorin_sij=0x8D;      //siirretään kursoria oikealle
                        break;

                      case 0x8D:
                        kursorin_sij=0x8B;      //siirretään kursori alkuun
                        break;
                  }
                  LCD_kirjoita_8bittia(0x8B,0);//siirretään kursori kohtaan 0B
                  LCD_kirjoita_teksti(Ah_uusi_raja);//tulostetaan asetusnäkymä
              }
              else if(painettu_nap1 && !painettu_nap2) //mikäli painiketta pidetään
```

```
        {                                           //pohjassa
            aika_2s++;
            if((aika_2s>199) && !sammutus)//mikäli painiketta on
            {                                       //painettu 10 ms*200=2 s
                aika_2s=0;
                aika_6s=0;
                sammutus=1;
                asetustila=0;
                vanha_Ah_raja = Ah_raja;      //tallennetaan vanha raja-arvo
                Tallenna_uusi_raja(Ah_uusi_raja); //tallennetaan uusi
                                              //asetettu raja-arvo
                LCD_kirjoita_Ah_raja();  //tulostetaan uusi raja-arvo
                Ah_uusi_raja[0]='0';     //asetetaan Ah-rajan
                Ah_uusi_raja[1]='0';     //asetus-muuttuja
                Ah_uusi_raja[2]='0';     //nollaksi
                kursorin_sij=0x8B;       //siirretään kursori kohtaan 08
                LCD_kirjoita_8bittia(0x0C,0); //kursori pois näkyvistä
            }

            else if((aika_2s>199) && sammutus)//mikäli painiketta on
            {                                 //painettu(10 ms*200)*2=4 s
                Ah_raja=vanha_Ah_raja;  //korvataan uusi asetettu raja-arvo
                                        //vanhalla raja-arvolla, sillä nyt
                kaynnissa=0;            //mittari sammutetaan
            }
        }
}

if(PINB & 0x04)                 //jos painiketta S2 ei ole painettu,
{                               //nollataan
   painettu_nap1=0;             //painikkeen painamista kuvaava
   sammutus=0;                  //muuttuja, sekä
}                               //sammutus-muuttuja

if(!(PINB & 0x02))              //mikäli painiketta S1
{                               //on painettu, mutta
if(!painettu_nap2)              //10ms sitten
   {                            //painiketta ei oltu painettu
      asetustila=1;             //siirrytään raja-arvon asetustilaan
      LCD_kirjoita_8bittia(0x0E,0);  //kursori näkyviin
      painettu_nap2=1;
      sammutus=0;
      aika_2s=0;
      aika_6s=0;
      switch(kursorin_sij)      //ehtona kursorin sijainti
      {
         case 0x8B:             //kasvatetaan kursorin kohdalla
            if(Ah_uusi_raja[0]==0x39)  //olevaa numeroa. Luvun 9
               Ah_uusi_raja[0]='0';    //jälkeen tulee 0
            else
               Ah_uusi_raja[0]++;
               break;
         case 0x8C:
```

```
            if(Ah_uusi_raja[1]==0x39)
              Ah_uusi_raja[1]='0';
            else
              Ah_uusi_raja[1]++;
              break;
          case 0x8D:
            if(Ah_uusi_raja[2]==0x39)
              Ah_uusi_raja[2]='0';
            else
              Ah_uusi_raja[2]++;
              break;
      }
LCD_kirjoita_8bittia(0x8B,0);          //siirretään kursori kohtaan 0B
LCD_kirjoita_teksti(Ah_uusi_raja);     //tulostetaan asetusnäkymä
   }
   else if(painettu_nap2 && !painettu_nap1) //mikäli painiketta S1
   {                                       //pidetään pohjassa ja painiketta
                                           //S2 ei ole painettu
      aika_2s++;
      if(aika_2s>199)                      //jos painiketta on painettu
      {                                    //10 ms*200=2 s
         aika_2s=0;
         aika_6s=0;
         sammutus=0;
         asetustila=0;
         Ah_vajaat=0;                      //nollataan kertyneet
         Ah_kokonaiset=0;                  //ampeeritunnit ja
         Ah_uusi_raja[0]='0';              //asetetaan Ah-rajan
         Ah_uusi_raja[1]='0';              //asetus-muuttuja
         Ah_uusi_raja[2]='0';              //nollaksi
         kursorin_sij=0x8B;                //siirretään kursori kohtaan 08
         LCD_kirjoita_8bittia(0x0C,0);     //kursori pois näkyvistä
      }
   }
}
if(PINB & 0x02)                          //jos painiketta S1 ei ole painettu,
painettu_nap2=0;                         //nollataan painikkeen painamista
                                         //kuvaava muuttuja
if(aika_6s>599)                          //mikäli painikkeita ei ole
{                                        //painettu 600*10ms = 6 sekuntiin
   asetustila=0;                         //palataan perustilaan ja
   aika_6s=0;                            //nollataan asetuksessa käytetyt
   aika_2s=0;                            //muuttujat
   sammutus=0;
   painettu_nap1=0;
   painettu_nap2=0;
   Ah_uusi_raja[0]='0';
   Ah_uusi_raja[1]='0';
   Ah_uusi_raja[2]='0';
   kursorin_sij=0x8B;                    //siirretään kursori kohtaan 08
   LCD_kirjoita_8bittia(0x0C,0);         //kursori pois näkyvistä
   LCD_kirjoita_Ah_raja();
}
```

```
    if((aika_200ms>19) && (asetustila==0))//näyttö päivitetään
    {                                     //20*10 ms = 200 ms välein
      aika_200ms=0;
      vilkutus++;
      LCD_kirjoita_I();
      LCD_kirjoita_T();
      LCD_kirjoita_Ah();
      if(Ah_kokonaiset >= Ah_raja)        //mikäli Ah-raja on ylittynyt
      {
        if(raja_ylittynyt)                //mikäli mittari on jo kerran
        {                                 //sammutettu raja-arvon
          if(vilkutus > 4)                //ylittymisen vuoksi, vilkutetaan
          {                               //raja-arvoa näytöllä
            LCD_kirjoita_Ah_raja();       //tulostamalla näyttöön
            if(vilkutus > 9)              //vuoroin raja-arvoa,
              vilkutus=0;                 //vuoroin välilyöntejä
          }
          if(vilkutus < 5)
          {
            LCD_kirjoita_8bittia(0x8B,0);
            LCD_kirjoita_8bittia(' ',1);
            LCD_kirjoita_8bittia(' ',1);
            LCD_kirjoita_8bittia(' ',1);
          }
        }
        if((!raja_ylittynyt) && (!sammutus))//mittari sammutetaan, mikäli
          kaynnissa=0;                    //Ah-raja on ylittynyt
      }                                   //käynnistämisen jälkeen

      if(Ah_kokonaiset < Ah_raja)         //mikäli Ah-rajaa on nostettu
      {                                   // tai kertyneet ampeeritunnit
        LCD_kirjoita_Ah_raja();           // on nollattu, nollataan myös
        raja_ylittynyt=0;                 // raja_ylittynyt-muuttuja
      }
    }
}                                         //Ah-rajan asettamisen if-rakenne
                                          //loppuu

if(asetustila != 1)                       //kalibrointivalikon
{                                         //if-rakenne alkaa. Rakenteeseen
                                          //mennään, mikäli Ah-rajan
                                          //asetus ei ole kesken
    if(!(PIND & 0x02))                    //mikäli kalibrointipainonappia
    {                                     //S3 on painettu, mutta 10ms
      if(!painettu_nap3)                  //sitten ei oltu painettu
      {
        painettu_nap3=1;
        aika_2s=0;
        aika_6s=0;
        if(asetustila != 2)               //jos kalibrointitilan
```

```
        {                                   //if-rakenteeseen mennään
                                            //ensimmäistä kertaa
           asetustila=2;                    //asetetaan asetustilaa
                                            //kuvaava muuttuja luvuksi 2
           LCD_kirjoita_8bittia(0x86,0);    //siirretään kursori kohtaan 06
           LCD_kirjoita_teksti(kalib_teksti);//kirjoitetaan kalibrointivalikon
        }          //teksti
        if(kalib_tila == 1)                 //mikäli edellinen tila oli
        {                                   //T_kerroin-muuttujan asetus
           kalib_tila=0;                    //vaihdetaan T:n jälkeinen
           LCD_kirjoita_8bittia(0x8D,0);    //kirjain nollaksi
           LCD_kirjoita_8bittia('0',1);
        }
        else if(kalib_tila == 0)            //mikäli edellinen tila oli
        {                                   //T_kalib-muuttujan asetus
           kalib_tila=1;                    //poistetaan näytöltä
           LCD_kirjoita_8bittia(0x8D,0);    //T:n jälkeinen nolla
           LCD_kirjoita_8bittia(' ',1);
        }
     }
     else if(painettu_nap3)                 //mikäli painiketta S3 pidetään
     {                                      //pohjassa 10 ms*200=2 s
        aika_2s++;
        if(aika_2s>199)
        {
           aika_2s=0;
           aika_6s=0;
           asetustila=0;
           cli();                           //kielletään keskeytykset
           vanha_T_kalib=T_kalib;           //tallennetaan uudet asetetut
           vanha_T_kerroin=T_kerroin;       //kalibrointimuuttujien arvot
           sei();                           //sallitaan keskeytykset
           LCD_alkutekstit();
        }
     }
  }
  if(PIND & 0x02)                           //jos painiketta S3 ei ole painettu,
     painettu_nap3=0;                       //nollataan painikkeen painamista
                                            //kuvaava muuttuja
  if(asetustila == 2)
  {
     aika_2s++;
     aika_6s++;
     aika_200ms++;
     if(!(PINB & 0x04))                     //mikäli painiketta S2 on painettu,
     {                                      //mutta 10 ms sitten ei oltu
        if(!painettu_nap1)                  // painettu
        {
           painettu_nap1=1;
           aika_6s=0;
           if(kalib_tila == 0)              //mikäli kalibrointilämpötilan
           {                                //asetus on menossa
              cli();
```

```
            T_kalib = (T_kalib-0.1);        //vähennetään
            sei();                          //kalibrointilämpötilan arvoa
         }
         if(kalib_tila == 1)                //mikäli T_kerroin-muuttujan
         {                                  //asetus on menossa
            cli();
            T_kerroin = (T_kerroin-0.01); //vähennetään T_kerroin-
            sei();                          //muuttujan arvoa
         }
      }
   }
   if(PINB & 0x04)                          //jos painiketta S2 ei ole painettu,
      painettu_nap1=0;                      //nollataan painikkeen painamista
                                            //kuvaava muuttuja

   if(!(PINB & 0x02))                       //mikäli painiketta S1 on painettu,
   {                                        //mutta 10ms sitten ei oltu
      if(!painettu_nap2)                    //painettu
      {
         painettu_nap2=1;
         aika_6s=0;
         if(kalib_tila == 0)                //mikäli kalibrointilämpötilan
         {                                  //asetus on menossa
            cli();
            T_kalib = (T_kalib+0.1);        //lisätään
            sei();                          //kalibrointilämpötilan arvoa
         }
         if(kalib_tila == 1)                //mikäli muuttujan T_kerroin
         {                                  //asetus on menossa
            cli();
            T_kerroin = (T_kerroin+0.01);//lisätään T_kerroin-
            sei();                          //muuttujan arvoa
         }
      }
   }
   if(PINB & 0x02)                          //jos painiketta S1 ei ole painettu,
   painettu_nap2=0;                         //nollataan painikkeen painamista
                                            //kuvaava muuttuja
   if(aika_6s>599)                          //Mikäli painikkeita ei ole
   {                                        //painettu 600*10ms = 6 sekuntiin
      asetustila=0;                         //palataan perustilaan ja nollataan
      aika_6s=0;                            //kalibroinnissa käytetyt
      aika_2s=0;                            //muuttujat
      painettu_nap1=0;
      painettu_nap2=0;
      painettu_nap3=0;
      cli();
      T_kalib=vanha_T_kalib;                //tallennetaan kalibrointi-
      T_kerroin=vanha_T_kerroin;            //muuttujien arvot
      sei();
      LCD_alkutekstit();
   }
   if(aika_200ms>19)                        //Tulostetaan I, Ah ja
```

```
                {                                       //kalibroitavan muuttujan
                  aika_200ms=0;                         //arvo (T_kalib tai T_kerroin)
                  LCD_kirjoita_I();                     //10ms*20=200ms välein
                  if(kalib_tila == 1)
                     LCD_kirjoita_T_tarkka();
                  if(kalib_tila == 0)
                    LCD_kirjoita_T_kalib();
                  LCD_kirjoita_Ah();
                }
              }
            }                                           //kalibrointivalikon if-rakenne
            lippu_10ms=0;                               //loppuu
          }                                             //10 ms if-rakenne loppuu
        }                                               //pääohjelmasilmukka loppuu

        cli();                                          //ei sallita enää keskeytyksiä
        LCD_kirjoita_8bittia(0x08,0);                   //näyttö pois päältä


        //    ** Muuttujien tallennukset EEPROM-muistiin **

        eeprom_write_word((unsigned int*)0, Ah_kokonaiset); //Ah-arvon tallennus

        eeprom_write_word((unsigned int*)2, Ah_raja);     //raja-arvon tallennus

        eeprom_write_block((const void*)&T_kalib, (void*)4, 4);     //lämpötilakertoimen
                                                                    //tallennus
        eeprom_write_block((const void*)&T_kerroin, (void*)8, 4); //T_kerroin-muuttujan
                                                                    //tallennus

        PORTB &= 0xFE;          //virransyötön ohjaus pois päältä -> koko laite sammuu

      }                         //pääohjelma päättyy
```

```
//   ** Aliohjelmat alkaa **


/*_____________________________________________________________________
 *
 * nimi: LCD_alustus
 * toiminta: alustaa LCD-näytön
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_alustus(void)
{
   LCD_kirjoita_8bittia(0x22,0);
   PORTD &= 0xFB;                  //RS alas, sillä käytetään Käskyrekisteriä
   PORTD &= 0x0F;                  //nollataan bitit D4-D7
   PORTD |= 0xC0;                  //2-rivinen, näyttö päälle

   PORTD |= 0x08;                  //E ylös
   _delay_loop_1(1);               //1 us viive
   PORTD &= 0xF7;                  //E alas
   _delay_loop_2(2000);            //1 ms viive (2*1000)

   LCD_kirjoita_8bittia(0x0C,0);   //näyttö päälle, ei kursoria [kursori näkyy(0E)]
                                   //eikä kursorin vilkutusta
   LCD_kirjoita_8bittia(0x01,0);   //näytön tyhjennys
   _delay_loop_2(4000);            //2 ms viive (2*2000)
   LCD_kirjoita_8bittia(0x06,0);   //kirjoitussuunta oikealle, ei shiftausta
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_8bittia
 * toiminta: kirjoittaa näytölle yhden kirjaimen tai
 *           käskyrekisteriin yhden käskyn. Data-parametri
 *           sisältää siirrettävän kirjaimen tai käskyn ja
 *           rekisteri-parametrilla valitaan näyttö-
 *           tai käskyrekisteri
 * parametri(t): data ja rekisteri
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_8bittia(unsigned char data, unsigned char rekisteri)
{
   if(rekisteri == 1)
   {
      PORTD |= 0x04;               //RS ylös, sillä käytetään Datarekisteriä
   }
   else
   {
      PORTD &= 0xFB;               //RS alas, sillä käytetään Käskyrekisteriä
   }

   _delay_loop_1(1);               //1 us viive
```

```
   PORTD &= 0x0F;                  //nollataan bitit D4-D7
   PORTD |= (data & 0xF0);         //siirretään ensin 4 eniten merkitsevää bittiä

   PORTD |= 0x08;                  //E ylös
   _delay_loop_1(1);               //1 us viive
   PORTD &= 0xF7;                  //E alas
   _delay_loop_2(2000);            //1 ms viive (2*1000)

   PORTD &= 0x0F;                  //nollataan bitit D4-D7
   PORTD |= ((data<<4) & 0xF0);    //shiftataan neljä bittiä vasemmalle, sillä nyt
                                   //sillä nyt siirretään neljä vähiten merkitsevää
                                   //bittiä

   PORTD |= 0x08;                  //E ylös
   _delay_loop_1(1);               //1 us viive
   PORTD &= 0xF7;                  //E alas
   _delay_loop_2(2000);            //1 ms viive (2*1000)
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_teksti
 * toiminta: kirjoittaa näytölle merkkijonon
 * parametri(t): teksti
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_teksti(unsigned char *teksti)
{
   unsigned char *teksti2=teksti;
   while(*teksti != '\0')          //kirjoitetaan lukujonosta merkki kerrallaan
   {                               //näytölle, kunnes '\0'-merkki
      LCD_kirjoita_8bittia(*teksti,1);  //tulee vastaan
      teksti++;
   }
   teksti=teksti2;
   LCD_kirjoita_8bittia(kursorin_sij,0); //siirretään kursori
}


/*_____________________________________________________________________
 *
 * nimi: LCD_alkutekstit
 * toiminta: kirjoittaa näytölle tekstit, jotka ovat
 *           näytöllä mittarin ollessa perustilassa
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_alkutekstit(void)
{
   LCD_kirjoita_8bittia(0x82,0);   //siirretään kursori kohtaan 02
   LCD_kirjoita_8bittia(' ',1);    //kirjoitetaan välilyönti
```

```
   LCD_kirjoita_8bittia(0x83,0);          //siirretään kursori kohtaan 03
   LCD_kirjoita_8bittia(0xDF,1);          //kirjoitetaan aste-merkki
   LCD_kirjoita_teksti(ekarivi);          //kirjoitetaan loput ensimmäisestä rivistä
   LCD_kirjoita_8bittia(0xC3,0);          //siirretään kursori kohtaan 43
   LCD_kirjoita_teksti(tokarivi);         //kirjoitetaan toinen rivi
   LCD_kirjoita_8bittia(0xCB,0);          //siirretään kursori kohtaan 4B
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_T
 * toiminta: kirjoittaa näytölle kokonaisluvuksi
 *              pyöristetyn lämpötilan
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_T(void)
{
   char mystr[5];
   unsigned char desimaaliesitys[4];
   float kopio=0;
   LCD_kirjoita_8bittia(0x80,0);                  //siirretään kursori kohtaan 00
   kopio = round(lampotila);                      //pyöristetään mitatun lämpötilan
                                                  //tarkka arvo lähimpään
                                                  //kokonaislukuun
   sprintf(mystr, "%.0lf", (double)kopio);        //floating point -> char
   desimaaliesitys[0] = (unsigned char)mystr[0];  //char -> unsigned char
   desimaaliesitys[1] = (unsigned char)mystr[1];  //char -> unsigned char
   desimaaliesitys[2] = (unsigned char)mystr[2];  //char -> unsigned char
   desimaaliesitys[3] = '\0';                     //merkkijono päättyy
   LCD_kirjoita_teksti(desimaaliesitys);          //tulostus näytölle
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_T_tarkka
 * toiminta: kirjoittaa näytölle lämpötilan kahden desimaalin
 *              tarkkuudella
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_T_tarkka(void)
{
   char mystr[6];
   unsigned char desimaaliesitys[6];
   LCD_kirjoita_8bittia(0x80,0);                  //siirretään kursori kohtaan 00
   sprintf(mystr, "%.2lf", (double)lampotila);    //floating point -> char
                                                  //katkaistaan mitatun lämpötilan
                                                  //arvo kahden desimaalin
                                                  //tarkkuuteen
   desimaaliesitys[0] = (unsigned char)mystr[0];  //char -> unsigned char
```

```
   desimaaliesitys[1] = (unsigned char)mystr[1];      //char -> unsigned char
   desimaaliesitys[2] = (unsigned char)mystr[2];      //char -> unsigned char
   desimaaliesitys[3] = (unsigned char)mystr[3];      //char -> unsigned char
   desimaaliesitys[4] = (unsigned char)mystr[4];      //char -> unsigned char
   desimaaliesitys[5] = '\0';                         //merkkijono päättyy
   LCD_kirjoita_teksti(desimaaliesitys);              //tulostus näytölle
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_Ah
 * toiminta: kirjoittaa näytölle kertyneet kokonaiset
 *           ampeeritunnit
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_Ah(void)
{
   char mystr[6];
   unsigned char desimaaliesitys[4];
   LCD_kirjoita_8bittia(0xC0,0);                      //siirretään kursori kohtaan 40
   sprintf(mystr, "%.0lf", (double)Ah_kokonaiset);    //unsigned int -> char
   desimaaliesitys[0] = (unsigned char)mystr[0];      //char -> unsigned char
   desimaaliesitys[1] = (unsigned char)mystr[1];      //char -> unsigned char
   desimaaliesitys[2] = (unsigned char)mystr[2];      //char -> unsigned char
   desimaaliesitys[3] = '\0';                         //merkkijono päättyy
   LCD_kirjoita_teksti(desimaaliesitys);              //tulostus näytölle
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_I
 * toiminta: kirjoittaa näytölle virran arvon
 *           kahden desimaalin tarkkuudella
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_I(void)
{
   char mystr[6];
   unsigned char desimaaliesitys[5];
   LCD_kirjoita_8bittia(0xCB,0);                      //siirretään kursori kohtaan 4B
   sprintf(mystr, "%.2lf", (double)virta0_100A);      //floating point -> char
                                                      //katkaistaan mitatun virran
                                                      //arvo kahden desimaalin
                                                      //tarkkuuteen
   desimaaliesitys[0] = (unsigned char)mystr[0];      //char -> unsigned char
   desimaaliesitys[1] = (unsigned char)mystr[1];      //char -> unsigned char
   desimaaliesitys[2] = (unsigned char)mystr[2];      //char -> unsigned char
   desimaaliesitys[3] = (unsigned char)mystr[3];      //char -> unsigned char
   desimaaliesitys[4] = '\0';                         //merkkijono päättyy
```

```
   LCD_kirjoita_teksti(desimaaliesitys);          //tulostus näytölle
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_T_kalib
 * toiminta: kirjoittaa näytölle asetetun kalibrointilämpötilan
 *              arvon kahden desimaalin tarkkuudella
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_T_kalib(void)
{
   char mystr[6];
   unsigned char desimaaliesitys[6];
   LCD_kirjoita_8bittia(0x80,0);                  //siirretään kursori kohtaan 00
   sprintf(mystr, "%.2lf", (double)T_kalib);      //floating point -> char
                                                  //katkaistaan
                                                  //kalibrointilämpötilan arvo
                                                  //kahden desimaalin tarkkuuteen
   desimaaliesitys[0] = (unsigned char)mystr[0];  //char -> unsigned char
   desimaaliesitys[1] = (unsigned char)mystr[1];  //char -> unsigned char
   desimaaliesitys[2] = (unsigned char)mystr[2];  //char -> unsigned char
   desimaaliesitys[3] = (unsigned char)mystr[3];  //char -> unsigned char
   desimaaliesitys[4] = (unsigned char)mystr[4];  //char -> unsigned char
   desimaaliesitys[5] = '\0';                     //merkkijono päättyy
   LCD_kirjoita_teksti(desimaaliesitys);          //tulostus näytölle
}


/*_____________________________________________________________________
 *
 * nimi: LCD_kirjoita_Ah_raja
 * toiminta: kirjoittaa näytölle asetetun Ah-rajan
 * parametri(t): ei parametreja
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void LCD_kirjoita_Ah_raja(void)
{
   unsigned int luku=Ah_raja;
   unsigned char Ah_raja_teksti[4]="000";
   unsigned char i=0;
   while(luku>99)                                 //tallennetaan sadat
   {                                              //lukujonoon Ah_raja_teksti
      luku=luku-100;
      i++;
   }
   if(i==0)
      Ah_raja_teksti[0]=' ';                      //tulostetaan välilyönti, jos luku
   else                                           //on pienempi kuin 100. Muussa
      Ah_raja_teksti[0]=(i+0x30);                 //tapauksessa luku muutetaan
                                                  //ASCII-merkiksi
```

```
   i=0;
   while(luku>9)                          //tallennetaan kymmenet
   {                                      //lukujonoon Ah_raja_teksti
      luku=luku-10;
      i++;
   }
   if((i==0) & (Ah_raja_teksti[0]==' '))
      Ah_raja_teksti[1]=' ';              //tulostetaan välilyönti, jos luku
   else                                   //on pienempi kuin 10. Muussa
      Ah_raja_teksti[1]=(i+0x30);         //tapauksessa luku muutetaan
                                          //ASCII-merkiksi
   i=0;
   while(luku>0)                          //tallennetaan kymmentä
   {                                      //pienemmät lukujonoon
      luku--;                             //Ah_raja_teksti
      i++;
   }
   Ah_raja_teksti[2]=(i+0x30);            //luku muutetaan ASCII-merkiksi
   LCD_kirjoita_8bittia(0x8B,0);          //siirretään kursori kohtaan 0B
   LCD_kirjoita_teksti(Ah_raja_teksti);   //tulostus näytölle
}


/*_____________________________________________________________________
 *
 * nimi: Tallenna_uusi_raja
 * toiminta: tallentaa käyttäjän asettaman Ah-rajan
 * parametri(t): tekstimuotoinen raja-arvo
 * paluuarvo(t): ei paluuarvoa
 *_____________________________________________________________________*/
void Tallenna_uusi_raja(unsigned char *tekstina)
{
   unsigned char luku= *tekstina;
   unsigned char *tekstina2=tekstina;

   Ah_raja = ((luku-0x30)*100);           //tallennetaan sadat
   tekstina++;
   luku= *tekstina;
   Ah_raja = (Ah_raja+((luku-0x30)*10));  //tallennetaan kymmenet
   tekstina++;
   luku= *tekstina;
   Ah_raja = (Ah_raja+(luku-0x30));       //tallennetaan kymmentä
   tekstina=tekstina2;                    //pienemmät
}
```