Antti Aitta

Ultraäänikäyttöinen vesimääräanturi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

25.4.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Antti Aitta Ultraäänikäyttöinen vesimääräanturi 33 sivua + 5 liitettä 25.4.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	sulautetut järjestelmät
Ohjaaja	koulutuspäällikkö Harri Airaksinen

Tämän insinöörityön tavoitteena oli selvittää erilaisia vaihtoehtoja vesimäärän mittaamiseen vesisäiliössä ja kehittää tehdyn tutkimuksen perusteella aurinkoenergialla toimiva kustannustehokas ja yksinkertainen laite vesimäärän mittaamiseen. Kehitetty prototyyppi suunniteltiin siten, että se kykenee toimimaan olosuhteissa, joissa lämpötila voi nousta 40 °C ja laskea lähelle 0 °C:een. Itse anturin on kestettävä 100 % kosteutta vesisäiliössä tapahtuvan haihtumisen vuoksi ja ulkopuolella olevien osien on kestettävä vesisade sekä muut sääolosuhteet. Laitteen toiminnan on myös oltava varmaa ja vakaata, siedettävä mahdollisia häiriöitä sekä oltava mahdollisimman itsenäistä ja käyttäjästä riippumatonta.

Työ suunniteltiin Sisiliaan yksityishenkilöiden käytössä oleviin vesisäiliöihin. Sisilian vesihuolto on rakennettu siten, että asuinrakennusten katoille tai ullakoille on sijoitettu juomavesisäiliö. Vesisäiliöt ovat tilavuudeltaan tyypillisesti muutaman kuutiometrin vetoisia ja ne täytetään kahdesta kolmeen kertaan viikossa. Vesihuollon ongelmia ovat kuitenkin vedensaantiin liittyvä ajoittainen epävarmuus ja vesisäiliössä olevan veden määrän epäselvyys.

Työn tuloksena on suunniteltu ja rakennettu edellä mainittuun tarpeeseen ultraäänianturia käyttävä mittalaite, joka mittaa vesimäärää ja toimii aurinkoenergialla. Laite on suunniteltu toiminnaltaan täysin häiriösietoiseksi ja varmatoimiseksi.

Avainsanat	Ultraääni, etäisyyden mittaus, vesisäiliö, vesimäärä

Author(s) Title Number of Pages Date	Antti Aitta Ultrasonic water quantity measuring device 33 pages + 5 appendices 25 April 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Ubiquitous Systems
Instructor	Harri Airaksinen, Head of Degree Programme

The goal for this Bachelor's thesis was to research different methods for measuring quantity of water in water tank. The goal included developing a solar powered, simple and cost efficient device to measure quantity of water in water tanks based on study. Developed prototype was planned to operate in conditions, where temperature may rise to 40 °C and decrease to near 0 °C. The probe itself is supposed to cope with 100 % humidity inside the water tank due to evaporation. Also parts located outside the water tank are supposed to cope with outside environments, such as rain and other weather conditions. Operation of the device has to be stable and it has to be immune to errors. The operation must need minimal user interaction.

This Bachelor's thesis was designed for drinking water tanks owned by Sicilian individuals. The water supply at Sicily is designed such that there are water tanks on rooftops and attics of houses. Water tanks can typically hold a few cubic meters of water and they are filled two to three times a week. Problems in the water supply are occasional uncertainty on supply and obscurity of water quantity in water tank.

This Bachelor's thesis resulted in a prototype of measurement device that works with solar power and uses ultrasonic sensor. The device itself has been designed to be as immune as possible to errors and stable.

Keywords	Ultrasonic, sonar, rangefinder, water quantity, water tank

Sisällys

Liitteet

Liite 1. Loogiset piirikaaviot

Liite 2. Fyysiset piirikuvat

Liite 3. Laitteen lohkokaavio

Liite 4. Ohjelmakoodin vuokaavio

Liite 5. Ohjelmakoodi

Lyhenteet ja käsitteet

ADC	*Analogic to Digital Converter*, analogisesta digitaaliseen muuntaja.
GPIO	*General Purpose Input Output*, eli yleiskäyttöinen syöte ja tuloste.
I^2C	Myös *Eye-to-see*, IIC ja I2C. Philipsin kehittämä kaksijohtiminen sarjamuotoinen väylä.
Insolaatio	*Incoming Solar Radiation*, auringon säteilystä maanpinnalle saapuva energia. Suureen SI-tunnus on E, yksikkönä $\frac{Wh}{m^2}/d$.
PSoC	*Programmable System on a Chip* tarkoittaa Cypress Semiconductorsin valmistamaa ohjelmoitavaa integroitua piiriä.
SCL	*Serial Clock*, I^2C-väylän kellosignaali.
SDA	*Serial Data*, I^2C-väylän datasignaali.
USB	*Universal Serial Bus*. Yleisin käytetty tiedonsiirtoväylä tietokoneen ja oheislaitteen välillä.

1 Johdanto

Sisiliassa käytössä oleva vesihuoltojärjestelmä on rakennettu siten, että rakennusten katoille tai ullakoille on sijoitettu juomavesisäiliö. Vesisäiliöt ovat tilavuudeltaan tyypillisesti muutaman kuutiometrin, ja ne täytetään kahdesta kolmeen kertaan viikossa. Vesihuollossa ongelmana on kuitenkin ajoittainen epävarmuus veden saannissa ja riittävyydessä sekä vesisäiliöissä olevan vesimäärän mittaamisen vaikeus. Mikäli asukas haluaa selvittää säiliössä olevan vesimäärän, on hänen fyysisesti mentävä säiliön luokse ja tarkistettava vesimäärä. Tämä aiheuttaa etenkin ikääntyneelle väestölle huomattavia riskejä. Tilanteet, joissa esimerkiksi pumppuasemien rikkoontuminen aiheuttaa vedentulon katkeamisen, eivät saisi päästä yllättämään väestöä. Veden määrän väheneminen on nähtävä ennalta, jotta siihen voidaan kunnolla varautua. Kuvasta 1 nähdään kaksisäiliöinen toteutustapa ja sen sijoittaminen rakennuksen ullakolle. Kuvasta nähdään myös täyttöputki säiliön yläosassa.

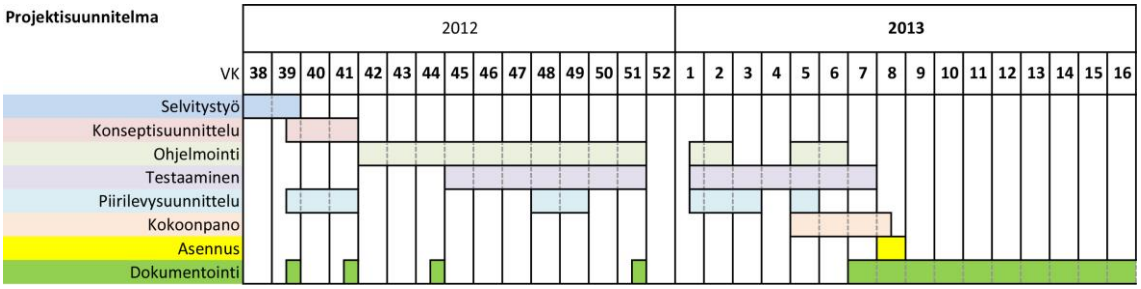
Kuva 1. Vesisäiliö rakennuksen ullakolla [1].

Tämän insinöörityön aiheena on selvittää erilaisia vaihtoehtoja vesimäärän mittaamiseen vesisäiliössä ja kehittää tehdyn tutkimuksen perusteella aurinkoenergialla toimiva kustannustehokas ja yksinkertainen laite vesimäärän selvittämiseen. Kehitettävä prototyyppi on suunniteltava siten, että se kykenee toimimaan olosuhteissa, joissa lämpötila voi nousta 40 °C:een ja laskea lähelle 0 °C:ta. Itse anturin on kestettävä 100 % kosteutta vesisäiliössä tapahtuvan haihtumisen vuoksi, ja ulkopuolella olevien osien on kestettävä vesisade ja muut sääolosuhteet. Laitteen toiminnan on myös oltava varmaa ja itsenäistä,

ja sen on oltava käyttäjästä riippumatonta. Insinöörityön on tilannut Metropolia Ammattikorkeakoulun Leppävaaran toimipisteen mediatekniikan koulutuspäällikkö Harri Airaksinen.

2 Työn aikataulu

Aloitin insinöörityön tekemisen välittömästi, kun olin saanut toimeksiannon. Työn aikatauluttaminen oli tärkein alkuun tehtävä asia, sillä se auttoi havainnoimaan käytettävissä olevaa aikaa ja arvioimaan tehtävänä olevia asioita. Kuvasta 2 nähdään projektin aikataulutus alkaen vuoden 2012 viikosta 38 ja päättyen vuoden 2013 huhtikuun lopulle, viikkoon 16.



Kuva 2. Projektin aikataulu.

Työn tekemiseen sisältyi seuraavia vaiheita:

- Selvitystyö

Selvitystyön aikana tutkin erilaisia etäisyyden mittaamisen menetelmiä ja jo olemassa olevia anturiratkaisuja, joita työssä olisi mahdollista käyttää. Selvitystyö valmistui todellisuudessa jo hieman aiemmin kuin suunnitelmassa, viikon 39 alussa.

- Konseptisuunnitelma

Konseptisuunnitelmassa ideoin ja suunnittelin karkeasti laitteen toimintaperiaatteen ja sen osat. Laitteen ensimmäinen lohkokaavio valmistui konseptisuunnittelun tuloksena. Lohkokaavio on liitteessä 3. Konseptisuunnitelma valmistui ajallaan.

- Ohjelmointi

Ohjelmointi oli kestoltaan pisin osa-alue, ja se jatkui projektin loppuun asti. Ohjelmoinnin alussa suunnittelin laitteen vuokaavion. Ohjelmointi oli jaksotettu siten, että ohjelman eri osa-alueet valmistuvat vuoden loppuun mennessä, minkä jälkeen niitä oli tarkoitus korjailla ja optimoida. Ohjelmoinnin suurin haaste oli saada käytettävä ultraäänianturi vastaamaan mittauskomentoon. Ongelman sain ratkaistua vasta viikolla 2, mikä aiheutti osaltaan kiirettä muun ohjelmakoodin valmiiksi saattamiseen.

- Testaaminen

Testaukselle olin aikatauluttanut aikaa koko työn ajaksi. Käytännössä testaaminen alkoi välittömästi, kun olin saanut kirjoitettua ensimmäiset rivit ohjelmakoodia.

- Piirilevysuunnittelu

Aloitin piirilevysuunnittelun välittömästi, kun ohjelmakoodin asettamat tarpeet tulivat selville. Piirilevysuunnittelu jatkui projektin loppuun asti.

- Kokoonpano

Kokoonpanovaihe piti sisällään piirilevyjen jyrsimiset ja komponenttien juottamisen niihin kiinni. Kokoonpanovaihe piteni Leppävaaran toimipisteen piirilevyjyrsimen vikaannuttua. Kokoonpano tapahtui viikolla 13 suunnitellun viikon 8 sijaan.

- Asennus

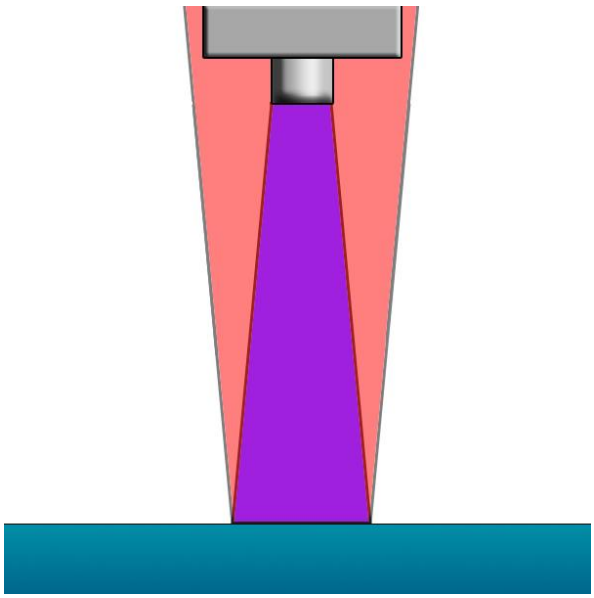
Asennusvaihe siirtyi viikolta 8 myöhemmin määrittelemättömään ajankohtaan kesällä, koska kokoonpanovaihe siirtyi yllättäen.

- Dokumentointi

Kirjoitin dokumentaatiota aina, kun jonkin osa-alueen työ tuli päätökseen. Varsinaisen kirjoitusurakan aloitin viikolla 8, kun työn intensiivisin vaihe päättyi.

3 Etäisyyden mittaamisen menetelmiä

Soveltuvaa etäisyyden mittaustapaa tutkiessani olen keskittynyt elektronisiin sovelluksiin. Tarkastelusta on siis jätetty pois erilaiset mekaaniset ratkaisut sekä optiikkaan ja trigonometriaan perustuvat ratkaisut. Tarkastellut menetelmät perustuvat heijastuman vastaanottamiseen. Kuva 3 osoittaa heijastuman vastaanottamiseen perustuvan mittaustavan toimintaa. Kuvassa sininen kappale on mitattava pinta, violetti alue lähetetty mittaussignaali ja punaiset alueet mittauspinnasta takaisin palaava heijastuma.



Kuva 3. Heijastumaan perustuva mittaustapa.

Anturikomponentti aloittaa ajan mittaamisen, kun se lähettää mittauksen ja lopettaa, kun se ottaa sen jälleen vastaan. Etäisyys on mahdollista laskea seuraavalla kaavalla:

$$x = \frac{t}{2} \cdot c$$

missä x on mitattu etäisyys,
t on anturin mittaama aika ja
c on lähetetyn mittaussignaalin nopeus väliaineessa.

3.1 Elektro-optinen mittaustapa

Optinen mittaustapa perustuu usein infrapunaan. Mittaus perustuu valon heijastumiseen ja heijastuman vastaanottamiseen. Infrapunamittaus häiriintyy helposti auringonvalosta

ja on tarkka mitattavan pinnan väristä. [2.] Koska infrapunavalo läpäisee vedenpinnan osittain ja vain osa heijastuu takaisin, ei infrapunamittaustapa sovellu hyvin tähän työhön.

3.2 Akustinen mittaustapa

Akustinen mittaustapa perustuu äänen heijastumiseen ja heijastuman vastaanottamiseen. Lähetetyn ja vastaanotetun äänen taajuusalue on yli 20 kHz, koska tämä on ihmiskorvalle kuulumaton alue. Kyseisellä taajuusalueella ei käytännössä kuulu muuta kuin mittauslaitteiden tuottamaa ääntä, eli mittaaminen on mahdollista toteuttaa häiriövapaasti.

Ultraäänimittaus soveltuu parhaiten litteisiin, laajoihin ja huonosti läpäistäviin kohteisiin, kuten nesteisiin. Ultraäänimittaus soveltuu huonosti monimuotoisten kappaleiden etäisyyksien mittaamiseen, koska niistä aiheutuvat heijastukset voivat saapua eri aikoihin mittapäähän. [3]. Tästä johtuen esimerkiksi nesteen vaahtoaminen voi aiheuttaa mittaustuloksessa tilapäisen vääristymän. Säiliön täyttämisen yhteydessä tapahtuva vaahtoaminen voi siis aiheuttaa tilapäisen mittausvirheen. Käytettävän vesisäiliön kannalta olisikin parasta, jos se olisi mahdollista täyttää säiliön alareunasta. Edellä esittämieni perustelujen vuoksi päädyin käyttämään insinöörityössäni akustista mittaustapaa.

4 Laitteen suunnittelu

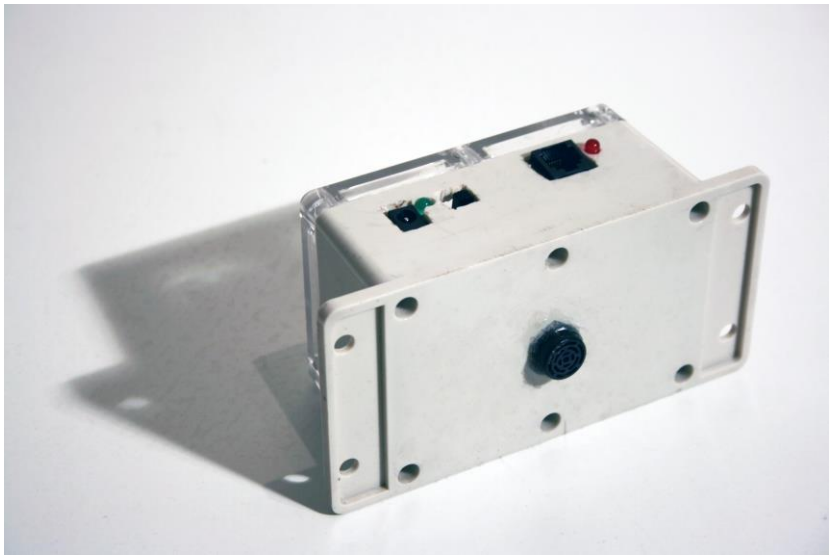
Laitteen suunnittelussa lähdin kaikilta osin siitä lähtökohdasta, että sen on täytettävä sille asetetut vaatimukset ja mielellään ylitettävä ne selkeästi. Laitteen olisi kyettävä mittaamaan luotettavasti 0 °C – 40 °C:een lämpötiloissa ja 100 % kosteudessa. Laitteen olisi myös kestettävä ulkoiset olosuhteet, kuten sade ja muut luonnonilmiöt.

4.1 Mekaaninen suunnittelu

4.1.1 Kotelointi

Päätin koteloida laitteen vähintään IP65-standardoituihin rasioihin, jotta vaadittu kosteudenkesto esimerkiksi sadetta vastaan saavutettaisiin. IP65-standardoitu rasia on pölyn

ja vesisuihkun kestävä [4, s.1]. Kotelointeihin oli kuitenkin tehtävä läpivientejä liitäntöjä varten. Anturirasiassa nämä kyseiset liitännät alentavat suojaustasoa merkittävästi, sillä prototyyppilaitteeseen ei otettu säänkestäviä liittimiä datalle ja virralle. Läpiviennit kuitenkin on tiivistetty kotelon rakenteeseen ja riittävä kosteudenkesto on näin saavutettu. Kuvasta 4 nähdään käyttöön valitun anturirasian rakenne. Vasemmassa reunassa on virtaliitin, virtatilaa osoittava vihreä LED-valo ja virtakytkin. Oikeassa reunassa on RJ45-liitin LED-paneelille ja häiriötilaa osoittava punainen LED-valo. Laitteen pohjassa on SRF02-ultraäänianturi, jonka lähetin-vastaanotinta varten on porattu rasian pohjaan reikä.



Kuva 4. Anturirasia.

Aurinkopaneeli koteloitiin omaan kehykseensä ja paneelin pinta suojattiin läpinäkyvällä akryylimuovipleksillä. Aurinkopaneelin kehys on kiinnitetty IP65-standardoituun koteloon, ja johtimia varten on koteloon porattu reikä, joka on myöhemmin tiivistetty epoksiliimalla. Myös aurinkopaneelin juotokset on suojattu epoksiliimalla, jotta ei syntyisi oikosulkua sisään mahdollisesti vuotavan veden seurauksena. Suihkutesteissä aurinkopaneelin sisäpuolelle ei päässyt lainkaan kosteutta. Kuvasta 5 nähdään aurinkopaneelin kotelointi. Kotelon taustassa on kiinni suuntaussarana ja kuvassa oikealla näkyy DC-virtaliitin.

Kuva 5. Aurinkopaneeli.

LED-paneeli koteloitiin tavanomaiseen IP-standardoimattomaan muovirasiaan. LED-paneelin en katsonut tarvitsevan erityistä kosteudenkestoa, koska sen sijoituspaikka on kuitenkin sisätiloissa. Kuvasta 6 nähdään LED-paneelin kotelo. Kotelon oikealla kyljellä on RJ45-liitin ja vasemmassa reunassa veden määrää osoittavat keltaiset LED-valot. Oikeassa ylälaidassa on häiriötilaa osoittava punainen LED-valo ja virtatilaa osoittava vihreä LED-valo.

Kuva 6. LED-paneeli.

4.1.2 Kaapelointi

Aurinkopaneelilta laitteelle tuleva virtajohto on toteutettu tyypillisellä ovaalilla kaksikaapelisella liitäntäjohdolla. Yksittäisen kaapelin halkaisija on 0,19 mm ja johtimia kaapeliparissa on 12 [5]. Virtajohdossa on käytetty myös 2,1/5,5 mm DC-liitintä. Liitinvalinta on maailman yleisimpiä, ja laitteelle on mahdollista hankkia erillinen verkkovirtalaite [6].

Anturirasian ja LED-paneelin välinen informaatio kulkee rinnakkaismuotoisena liikenteenä kahdeksanjohtimisessa suoraan kytketyssä CAT-5-verkkokaapelissa. Verkkokaapelin käyttö takaa häiriöttömyyden ja helpon asennuksen, sillä verkkokaapelit ovat tyypillisesti häiriösietoisia ja kaapeli on mahdollista ostaa juuri halutun pituisena. Tämä myös mahdollistaa olemassa olevien käyttämättömien kaapelointien tehokkaan hyödyntämisen.

4.2 Piirilevysuunnittelu

Piirilevysuunnittelu tehtiin Mentor Graphicsin PADS Logic- ja PADS Layout -ohjelmistoilla. Loogiset piirikaaviot (liite 1) ja piirilevyjen fyysiset piirrokset (liite 2) ovat liitteinä. Piirilevysuunnittelu alkoi loogisen piirikaavion piirtämisellä. Ensimmäisten piirrettyjen piirilevyjen perusteella valitsin käytettävän kotelon mittalaitteelle ja sen perusteella oli mahdollista piirtää tarkempi, lopullinen piirilevy. Suurin osa piirilevyn johtimista on toteutettu 24 millsin leveydellä, mutta ahtaissa kohdissa johtimen leveys oli kavennettava 12 millsiin. Myös RJ45-liittimeltä lähtevät johtimet oli toteutettava 12 millsin leveydellä. Koska laitteessa ei kulje suuria virtamääriä, ovat johdinleveydet täysin riittävät.

Piirilevysuunnittelun loppupuolella tehdyissä koelevyissä havaitsin muutamia virheitä, jotka korjasin lopulliseen levyyn. Koelevyjen perusteella valmistui lopullinen piirilevy, jota ei päästy jyrsimään ajoissa Leppävaaran toimipisteen piirilevyjyrsimeen tulleen vian vuoksi. Tästä johtuen laitteen luovutus Sisiliassa tehtäviä testejä varten viivästyi.

Prototyyppilevyjä ja pieniä piirilevysarjoja valmistavalla Prinelillä piirilevyn tuottaminen olisi maksanut noin 200 €, joten piirilevy oli jyrsittävä Metropolian Albertinkadun toimipisteessä. Albertinkadun toimipisteen laitteet eivät tosin aluksi ymmärtäneet luotuja piirilevytiedostoja, mutta laboratorioinsinööri Teemu Mahrbergin avustuksella levy saatiin jyrsittyä.

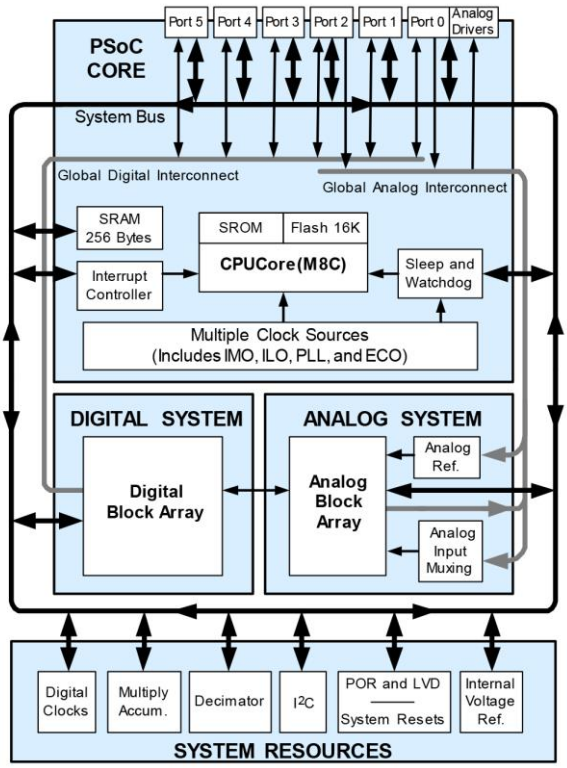
Valitsin piirilevylle sijoitettavat komponentit laitteelle asetettujen vaatimusten perusteella. Alhaisin lämpötila, jossa kaikki komponentit toimivat, on -30 °C ja korkein kaikkien komponenttien kestämä lämpötila on 70 °C. Taulukkoon 1 kirjattujen käyttölämpötilojen perusteella laite kestää vaaditun 0 °C–40 °C lämpötilat.

Taulukko 1. Komponenttien käyttölämpötilat [7, s. 20; 8, s. 4; 9, s. 2; 10, s. 1].

Komponentti	**Käyttölämpötila**
CY8C27443PXI	-40 °C–105 °C
IRF9530	-55 °C–150 °C
LM78xxCT	-40 °C–125 °C
MIC4427CN	-40 °C–85 °C
SRF02	-30 °C–70 °C

4.2.1 Mikrokontrolleri

Laitteen mikrokontrolleriksi valitsin Cypressin PSoC CY8C27443PXI:n. CY8C27443PXI on Cypress Semiconductorsin valmistama ohjelmoitava integroitu piiri. CY8C27443PXI sisältää 8-bittisen Harvard-arkkitehtuuriin pohjautuvan 24 MHz-prosessorin ja kahdeksan digitaalista ja 12 analogista moduulia.



Kuva 7. PSoC-arkkitehtuuri [7, s. 1].

Kuvasta 7 nähdään PSoC-arkkitehtuuriin kuuluvat neljä pääaluetta: PSoC-ydin, analoginen ja digitaalinen järjestelmä sekä järjestelmäresurssit. PSoC-ytimeen kuuluu 8-bittinen Harvard-arkkitehtuuriin pohjautuva 24MHz-prosessori, kellot, muisti ja GPIO-pinnit.

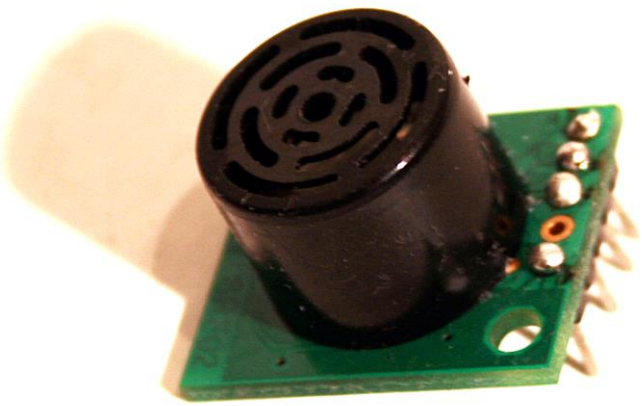
Digitaalinen järjestelmä koostuu kahdeksasta digitaalisesta PSoC-palikasta, joista kukin toimii kahdeksanbittisenä resurssina. PSoC-palikoita voi yhdistellä ja yhdistelmillä voi muodostaa 16-, 24- ja 32-bittisiä moduuleita. Digitaaliset PSoC-palikat voidaan myös kytkeä GPIO-pinneihin järjestelmäväylän kautta. Mittalaitteessa on käytössä seitsemän digitaalista PSoC-palikkaa, joista kuusi on kahdelle erilliselle AD-muuntimelle ja yksi ajastimelle.

Analoginen järjestelmä koostuu 12 muokattavasta PSoC-palikasta, jotka kukin sisältävät operaatiovahvistinkytkennän, joita voidaan kytkeä analogisiin sisääntuloihin. Mittalaitteessa on käytössä neljä analogista moduulia, joista kaksi on AD-muuntajille ja kaksi AD-muuntajia varten oleville vahvistimille.

Järjestelmäresursseihin kuuluu muun muassa kellojakajat ja I^2C-moduuli. Kellojakajilla on mahdollista jakaa sisäisten kellojen kellotaajuuksia siten, että niitä on helpompi sovittaa eri moduulien käyttöön. I^2C-moduulia on mahdollista käyttää nopeuksilla 100b kbit/s ja 400 kbit/. [7, s. 3–5.]

4.2.2 Ultraäänianturi

Devantech SRF02 on yhtä lähetin-vastaanotinta käyttävä ultraäänianturi. SRF02 tukee sekä I^2C- että sarjaväylää. SRF02 ottaa vastaan mittauskäskyn ja suorittaa mittauksen joko senttimetreissä tai tuumissa. SRF02 valikoitui käytettäväksi ultraäänianturiksi, sillä



se tarjosi valmiin komponentin, vaaditun kosteudenkeston, pienen virrankulutuksen ja toisaalta helpon keinon lukea saatua dataa. SRF02 mittaa tehokkaasti noin 16 cm–600 cm. [11.] Kuitenkin käytännön testeissä havaittiin, että luotettavasti laite mittaa vasta noin 20 cm:n jälkeen. Tämä johtuu niin sanotusta ringing-ilmiöstä. Ilmiössä anturin pietsoelektroninen lähetinvastaanotin jää värähtelemään lähetyksen jälkeen hetkeksi. Tämä aiheuttaa sen, että lähellä olevan kappaleen kaiku sekoittuu värähtelyyn, eikä mittaus onnistu tarkasti. [12, s. 2.] Kuvasta 8 nähdään SRF02-ultraäänianturin yksikomponenttinen lähetinvastaanotin.

Kuva 8. SRF02-ultraäänianturi.

Ilman lämmetessä muuttuu myös äänen nopeus väliaineessa, ja samalla muuttuu mittaustuloksen tarkkuus. SRF02:n suositeltu käyttölämpötila on 20 °C [13]. Äänen nopeus tietyn lämpöisessä ilmassa on mahdollista laskea seuraavalla kaavalla ja tulokset on sijoitettu taulukkoon 2 [14].

$$V_{ÄÄNI} = \sqrt{\frac{\gamma RT}{M}}$$

$$V_{ÄÄNI} = \sqrt{\frac{1{,}4\left(8{,}314\,\frac{J}{mol}\right)T}{0{,}02895\,\frac{kg}{mol}}}$$

Missä ɣ on adiabaattivakio = 1,4

R on kaasuvakio = $8{,}314\,\frac{J}{mol}K$

M on moolimassa = $0{,}02895\,\frac{kg}{mol}$

T on absoluuttinen lämpötila °K

Taulukko 2. Äänen nopeus tietyssä lämpötilassa.

Ilman lämpötila	**Äänen nopeus ilmassa**	**Aika metrillä**
0 °C	331 m/s	3,00 ms
10 °C	337 m/s	2,96 ms
20 °C	343 m/s	2,91 ms
30 °C	349 m/s	2,86 ms
40 °C	355 m/s	2,81 ms

Aikaero metrin matkalla äärilämpötilojen välillä on noin 0,1 ms normaaliin eli 20 °C lämpötilaan.

$$METRI_{40\,°\mathrm{C}} = 343m/s \cdot 2{,}81ms$$
$$METRI_{40\,°\mathrm{C}} = 0{,}96m$$

$$METRI_{0\,°\mathrm{C}} = 343m/s \cdot 3{,}00ms$$
$$METRI_{0\,°\mathrm{C}} = 1{,}03m$$

40 °C:n mittausvirhe on siis -4cm ja 0 °C mittausvirhe 3 cm. Koska mittaustulokset esitetään 10 %:n tarkkuudella ja matalimmalla 160 cm kalibroinnilla tarkkuus on 16 cm, eivät edellä mainitut -4 cm:n ja +3 cm:n erot vaikuta merkittävästi tulokseen. Matalalla säiliöllä ongelma korostuu, kun lähestytään mittauksen ääripäitä eli 100 %:n ja 0 %:n vesimääriä. Kylmällä säällä vettä näyttää olevan hieman vähemmän kuin todellisuudessa, ja kuumalla säällä vettä näyttää olevan enemmän kuin todellisuudessa.

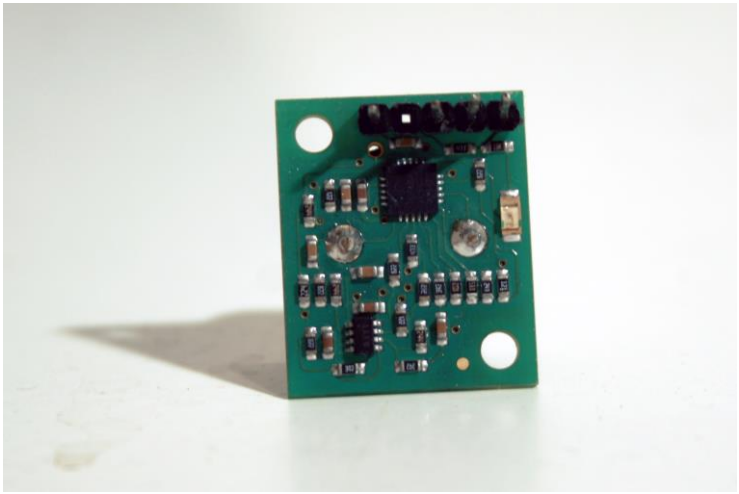
Äänen nopeuteen ilmassa vaikuttaa myös kosteus. Oletan kosteuden säiliössä olevan lähellä 100 %:a. Äänen nopeuksia tietyssä lämpötilassa ja suhteellisessa kosteudessa on kirjattu taulukkoon 3.

Taulukko 3. Äänen nopeus eri kosteusprosenteilla ja lämpötiloilla [15].

Suhteellinen kosteus	**Lämpötila**	**Äänen nopeus**
50 %	0 °C	331,61 m/s
100 %	0 °C	331,76 m/s
50 %	40 °C	356,88 m/s
100 %	40 °C	358,93 m/s

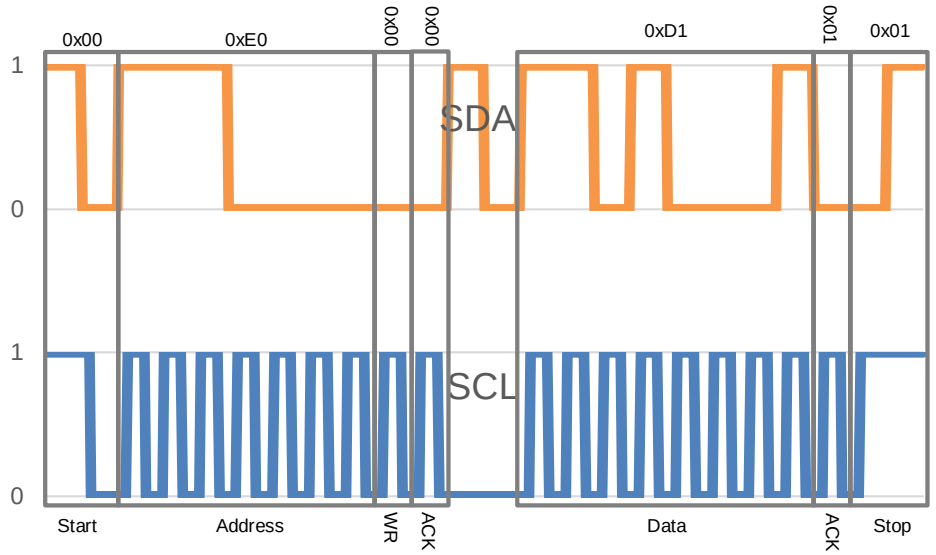
Taulukkoon 2 ja taulukkoon 3 merkittyjä tuloksia vertailemalla on mahdollista nähdä, että ilman kosteus ei ratkaisevasti vaikuta mittaustuloksiin.

SRF02:n käyttämä I^2C-väylä on Philipsin 1980-luvulla kehittämä synkroninen, kaksisuuntainen ja -johtiminen sarjamuotoinen tiedonsiirtoväylä. Väylän johtimet ovat Serial Data (SDA) ja Serial Clock (SCL). [16.]



Kuva 9. SRF02-ultraäänianturi.

Kuvasta 9 nähdään SRF02:n piirilevy. Oikeassa ylälaidassa on piikkirima, jonka pinnit ovat vasemmalta oikealle: 0 V, Mode (sarjaliikennettä varten, kytkemättä), SDA, SCL ja +5 V. SDA-johtimessa kulkee sarjamuotoinen tiedonsiirto ja SCL-väylässä isäntälaitteen kehittämä kellosignaali. I^2C-väylällä voi olla samanaikaisesti useita isäntä- ja orjalaitteita. Väylä havaitsee törmäyksen kahden isäntälaitteen yrittäessä varata väylää samaan aikaan. Väylän törmäyksenhavainnointi myös estää tiedon katoamisen. Väylän laitteet erotetaan toisistaan seitsenbittisellä osoitteella, joka on kullakin laitteella yksilöllinen. Sekä SDA- että SCL-johtimessa on ylösvetovastukset 5 V:iin. Kun isäntälaite haluaa varata väylän käyttöönsä, se kytkee SDA-johtimen maapotentiaaliin. Orjalaitteet eivät voi varata väylää käyttöönsä ja ne ovat toiminnaltaan passiivisia. I^2C:n väylänopeudet ovat 100 kbit/s (Standard mode), 400 kbit/s (Fast-mode) ja 3,4 Mbit/s (High-speed mode).



Kuvio 1. SDA- ja SCL-johtimet I²C-väylällä.

Kuviosta 1 nähdään I²C-väylällä tapahtuva kommunikaatio isäntälaitteen kirjoittaessa osoitteessa 0xE0 sijaitsevalle orjalaitteelle heksaluvun 0xD1. Kommunikaatiossa on seuraavia osia:

- Start
- Address
- R/W
- ACK/NAK
- Data
- Stop

Start-vaiheessa isäntälaite laskee SDA-johtimen maapotentiaaliin, minkä jälkeen isäntälaite alkaa tuottaa SCL-johtimeen kellosignaalia. Address-vaiheessa isäntälaite lähettää seitsenbittisen osoitteen orjalaitteelle. Osoitteen perässä on kirjoitusbitti (R/W), joka kuviossa 1 on asetettu kirjoitukseen (0x00). Luettaessa bitti olisi ylhäällä (0x01). Osoite voidaankin siis nähdä kahdeksanbittisenä, jolloin kirjoitus ja lukeminen tapahtuvat eri osoitteesta: 0xE0 (luku) ja 0xE1 (kirjoitus). Orjalaite vastaa acknowledgement-bitillä (ACK) 0x00, mikä osoittaa isäntälaitteelle tavun saapuneen perille. Kukin väylään kirjoitettu tavu on osoitettava päättyneeksi acknowledgement-bitillä. Acknowledgement-bitin tilalla voi olla myös not-acknowledgement-bitti 0x01 (NAK). Not-acknowledgement-bitti tarkoittaa, että kyseistä orjalaitetta ei ole olemassa, se ei voi ottaa komentoa vastaan tai

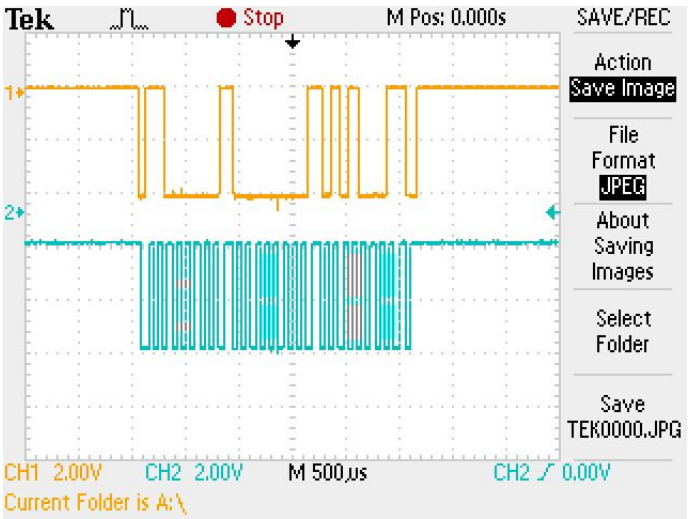
se ei ymmärrä annettua komentoa. Isäntälaite ilmoittaa orjalaitteelle päättävänsä lukemisen not-acknowledgement-bitillä. Acknowledgement-bitin jälkeen SDA-johtimeen kirjoitetaan lähetettävä tavu, joka myös osoitetaan vastaanotetuksi acknowledgement-bitillä. Lopuksi kommunikaatiota päätettäessä isäntälaite luovuttaa väylän vapaaksi nostamalla SDA-linjan ylös 5 V:in, kun SCL-linja on jo ylhäällä 5 V:ssa.

I²C-väylällä on myös mahdollista nähdä toistettu aloitus (engl. repeated start), jolloin orjalaitteet tietävät tulevan viestin olevan osa edellistä lähetystä. Toistettu aloitus tapahtuu, kun väylä on vielä varattu, eikä siihen ole vielä lähetetty pysäytysbittiä. [17, s.4–11.]

SRF02:n ja PSoC:in välinen kommunikointi I²C-väylän ylitse osoittautui varsin haasteelliseksi. En saanut anturia ottamaan komentoja vastaan ennen tammikuuta. Asiaa selviteltäessä olin yhteydessä valmistajaan yrityksen keskustelualueella. Heidän kanssa yhteistyössä selvisi, että joko PSoC tai SRF02 käsittelee ensimmäisiä neljää bittiä väärin.

> ”Zoom in on the first E0, it looks more like C0 to me, with just two bits high instead of three. C0 -> 1100 0000, E0 -> 1110 0000” [18].

Kuvasta 10 nähdään, kuinka ensimmäisessä kahdeksan bitin jonossa on vain kaksi bittiä ykkösenä kolmen sijasta.



Kuva 10. Virheellinen lukukomento I²C väylässä oskilloskoopilla tarkasteltuna

Ongelma oli joko PSoC:in tai S6RF02:n I²C-väylän ominaisuuksissa. SRF02:n osoite on tehdasasetuksilla 0xE0, mutta PSoC:n lähettämä osoite piti muuttaa 0xF0:ksi. Toisin sanoen 0xE0:n bittijonoon 1110 000 lisättiin ensimmäiseen puoliskoon yksi bitti, jolloin siitä tuli 0xF0 eli 1111 0000.

4.2.3 Tehonlähde

Tehonlähdepiiri käyttää Fairchild Semiconductorsin LM7805 ja LM7809 -lineaariregulaattoria. LM7805 muuttaa aurinkopaneelilta tai akulta saadun jännitteen 5 V:iin ja LM7809 muuttaa akulle menevän latausjännitteen 9 V:in [10, s. 3, 6]. Tehonlähdepiirissä on myös jännitteenjaolla toteutettu jännitteenalennus AD-muunnosta varten. Jännitteenjako on laskettu olettamalla akun korkeimmaksi jännitteeksi 9 V ja aurinkopaneelin korkeimmaksi jännitteeksi 15 V. Akun jännitemittauksen vastuksien R1 ja R2 arvoiksi mitoitettiin 900 Ω ja 1 kΩ. Aurinkopaneeliin jännitteenmittauksen vastuksien R3 ja R4 arvoiksi mitoitettiin 1 kΩ ja 1,85 kΩ.

$$V_{adbat} = \frac{R_2 \cdot VCC}{R_1 + R_2}$$
$$V_{adbat} = \frac{1\text{k}\Omega \cdot 9\text{V}}{800\Omega + 1\text{k}\Omega}$$
$$V_{adbat} = 5{,}0V$$

$$V_{adsol} = \frac{R_2 \cdot VCC}{R_1 + R_2}$$
$$V_{adsol} = \frac{1\text{k}\Omega \cdot 9\text{V}}{1{,}85\text{k}\Omega + 1\text{k}\Omega}$$
$$V_{adsol} = 5{,}26V$$

Jännitteen ohjausta varten laitteessa on kaksi IRF9530-mosfetia ja niiden yhteydessä MIC4427CN-mosfet-ajuria. Ajureiden avulla on tarkoitus ohjata aurinkopaneelilta tuleva jännite hilalle. MIC4427CN-ajuria ohjataan mikrokontrollerin signaalilla siten, että kun jännite nostetaan 5 V:iin, laskee MIC4427CN hilan jännitteen 0 V:iin, jolloin mosfet katkaisee virtapiirin. Jos mikrokontrollerin jännite putoaa, muuttuu mosfet automaattisesti johtavaksi. Tämä mahdollistaa palautumisen, jos akun ja aurinkopaneelin jännite laskee niin, että koko laite sammuu.

4.2.4 Sähkömagneettinen yhteensopivuus

Laitteen valmistamisessa on sähkömagneettisen yhteensopivuuden kannalta otettu huomioon lähinnä suojaaminen kipinäpurkauksia ja ylijännitteitä vastaan. Kipinäpurkauksia vastaan suojautuminen on toteutettu etuvastuksella, joka rajoittaa sisään pääsevän vir-

ran 150 mA:in. Kytkentäilmiötä ja sen suuria jännitteitä vastaan on suojauduttu diodisuojauspiirillä. LM7805 ja L7809 on suojattu sisäisesti omilla kipinäpurkaussuojauksilla ja ylijännitesuojauksella. LM705 ja LM7809 suojaavat myös muita arkoja piirejä levyllä. Mikäli kipinäpurkaus aiheuttaa laitteen uudelleenkäynnistymisen, osaa se jatkaa toimintaansa ilman käyttäjän väliintuloa.

Laite ei lähetä eikä myöskään vastaanota mitään, minkä vuoksi oletan myös, että se ei häiriinny ulkoisesta säteilystä eikä myöskään lähetä häiritsevää säteilyä. Olettamusta myös tukee se, että laitteessa kulkee vain tasajännite, jolloin sen ei tulisi aiheuttaa minkäänlaista resonanssia. Laitteen johtimet ovat myös pituudeltaan sellaiset, että ne eivät osu yleisesti käytössä olevien taajuuksien puoli- tai neljännesaallonpituuksille.

4.2.5 Lämpöanalyysi

Lämpöanalyysi suoritettiin niiden komponenttien osalta, joilla analyysi on datalehden tietojen mukaan mahdollista suorittaa. Lämpöanalyysin tarkoituksena on selvittää, voivatko laitteen yksittäiset komponentit tuhoutua oman lämpenemisensä seurauksena. Taulukkoon 4 on merkitty eri komponenttien lämpövastukset.

Taulukko 4. Eri komponenttien lämpöarvoja [7, s. 20; 8, s. 4; 9, s. 2; 10, s. 1].

Komponentti	**Lämpöresistanssi $R_{\Theta JC}$**	**Maksimaalinen liitinlämpötila T_{JMAX}**
CY8C27443	95 °C/W	125 °C
IRF9530	1,67 °C/W	150 °C
LM780x	5 °C/W	125 °C
MIC4427CN	42 °C/W	85 °C

Komponentin suurin sietämä tehonkulutus lasketaan seuraavalla kaavalla: [19, s. 17.]

$$Q_{MAX} = \frac{T_{JMAX} - T_{AMB}}{R_{\Theta JC}}$$

$$Q_{MAXLM780x} = \frac{125\ °\mathrm{C} - 40\ °\mathrm{C}}{5\ °\mathrm{C/W}}$$

$$Q_{MAXLM780x} = \frac{125\ °\mathrm{C} - 40\ °\mathrm{C}}{5\ °\mathrm{C/W}}$$

$$Q_{MAXLM780x} = 17\ W$$

$$Q_{MAX\mathrm{MIC4427CN}} = \frac{85\ °\mathrm{C} - 40\ °\mathrm{C}}{42\ °\mathrm{C/W}}$$

$$Q_{MAX\mathrm{MIC4427CN}} = 1{,}07\ W$$

$$Q_{MAX\mathrm{CY8C27443}} = \frac{125\ °\mathrm{C} - 40\ °\mathrm{C}}{95\ °\mathrm{C/W}}$$

$$Q_{MAX\mathrm{CY8C27443}} = 0{,}89\ W$$

$$Q_{MAX\mathrm{IRF9530}} = \frac{150\ °\mathrm{C} - 40\ °\mathrm{C}}{1{,}67\ °\mathrm{C/W}}$$

$$Q_{MAX\mathrm{IRF9530}} = 65{,}86\ W$$

Missä T_{AMB} = 40 °C

Lasketaan kunkin komponentin tehonkulutus, jotta saadaan selville ylittääkö niiden tehonkulutus niiden laskennallisia maksimiarvoja:

$$P = U \cdot I$$

$$P_{LM780x} = U_{LM7805} \cdot I_{LM7805MAX}$$

$$P_{LM780x} = 14\ V \cdot 8\ mA$$

$$P_{LM780x} = 0{,}112\ W$$

LM7805:n ja LM7809:n kohdalla tehonkulutus pysyy selkeästi asetetuissa rajoissa.

$$P_{\mathrm{MIC4427CN}} = U_{\mathrm{MIC4427CN}} \cdot I_{LM7805MAX}$$

$$P_{\mathrm{MIC4427CN}} = 14\ V \cdot 4\ mA$$

$$P_{\mathrm{MIC4427CN}} = 0{,}056\ W$$

MIC4427CN:n kohdalla tehonkulutus pysyy myös asetetuissa rajoissa.

$$P_{\mathrm{CY8C27443PXI}} = U_{\mathrm{CY8C27443PXI}} \cdot I_{\mathrm{CY8C27443PXI}}$$

$$P_{\mathrm{CY8C27443PXI}} = 5\,V \cdot 8\,mA$$

$$P_{\mathrm{CY8C27443PXI}} = 0{,}04\,W$$

Myöskään PSoC ei ylitä tehonkulutuksellaan maksimirajoja. Seuraavaksi lasketaan IRF9530:ssä lähteestä hilaan kulkevan virran aiheuttama lämpeneminen.

$$P_{\mathrm{IRF9530}} = U_{\mathrm{RF9530}} \cdot I_{\mathrm{RF9530}}$$

$$P_{\mathrm{IRF9530}} = 14\,V \cdot 150\,mA$$

$$P_{\mathrm{IRF9530}} = 2{,}1\,W$$

Siitä huolimatta, että IRF9530:n läpi kulkee laitteen mittakaavassa suuri virta, ei lämmitysteho riitä ylittämään maksimiarvoja.

Lämpöanalyysin perusteella laite ei normaalikäytössä tuota itse lämpöä niin paljoa, että se estäisi laitteen käyttämisen korkeimmissa odotettavissa olevissa lämpötiloissa.

4.3 Ohjelmistosuunnittelu

Laitteen ohjelmisto noudattaa vuokaaviota, joka on liitteenä 4. Ohjelmointi on tehty C-kielellä, joskin keskeytyksiä varten on täytynyt lisätä kaksi riviä Assemblyä. Laitteen ohjelmisto koostuu kahdesta päätilasta: kalibrointitilasta ja mittaustilasta. Ohjelmakoodi itsessään on liitteenä 5.

4.3.1 Kalibrointitila

Kalibrointitilassa laite odottaa joko käyttäjän antamaa syötettä tai 60 sekuntin kulumista. Käyttäjä voi antaa omana syötteenään säiliön korkeuden 60—200 cm:n välille 10 cm:n porrastuksin. Kalibrointi tehdään kolmella mikrokytkimellä, jotka kukin aiheuttavat keskeytyksen [20]. Keskeytysrutiinissa ohjelmisto tarkistaa, mitä painiketta on painettu ja suorittaa kunkin painikkeen edellyttämät toimenpiteet. Ylös-painikkeen painaminen kasvattaa kalibrointiarvoa heksadesimaalilla 0x0A, eli 10 cm, ja alas-painikkeen painaminen taas pienentää kalibrointiarvoa vastaavasti. Valintapainike valitsee kyseisen arvon, minkä jälkeen laite voi jatkaa mittaustilaan.

Laite siirtyy mittaustilaan myös siinä tapauksessa, jos on kulunut 60 sekuntia. Ohjelmistossa on ajastinmoduuli, joka antaa keskeytyksen millisekuntin välein. Keskeytysrutiinissa ohjelmisto laskee kolmessa eri muuttujassa tapahtuneita keskeytyksiä eli millisekunteja. Näiden avulla lasketaan 60 sekuntia, minkä jälkeen merkitään 60 sekuntia kuluneeksi ja ohjelmisto siirtyy mittaustilaan.

4.3.2 Mittaustila

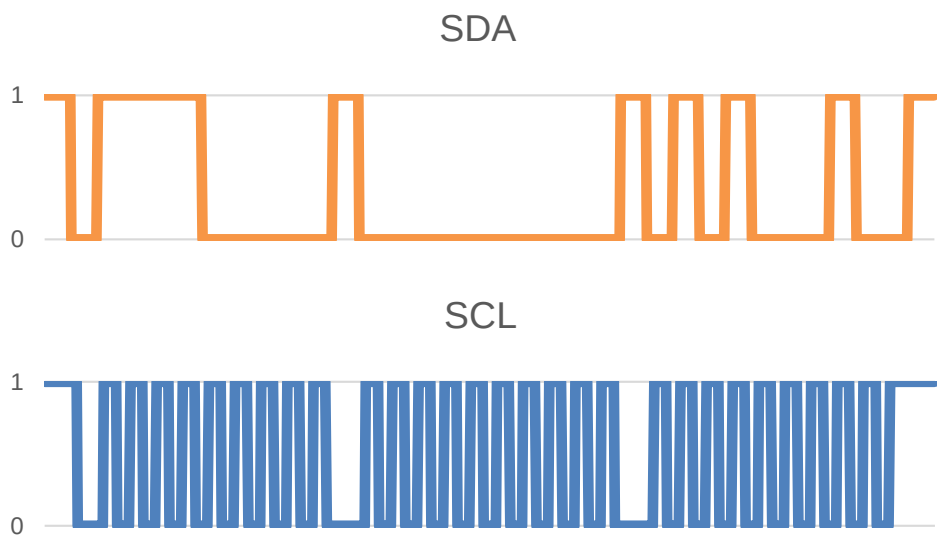
Laite on mittaustilassa niin kauan kuin virtaa riittää. Kalibrointitilaan palataan vain uudelleenkäynnistyksen kautta. Mittaustilassa on neljä eri vaihetta: jännitteenseuranta-, mittaus-, tiedonkäsittely- ja virheenkäsittelytila.

Jännitteenseurantatilassa laite tarkkailee AD-muunnoksen avulla sekä aurinkopaneelin että akun jännitettä. Alhaiseksi jännitteeksi määritettiin 4 V, sillä PSoC toimii vielä 3,0 V jännitteellä [7, s. 1]. Alhaisen jännitteen arvoksi luettiin PSoC:n AD-muunnosmoduulilla 0x8A. Jotta raja-alueella tapahtuva jännitteenvaihtelu ei aiheuttaisi korkeataajuista muuttumista alhaisen ja riittävän jännitteen välillä, laskee ohjelma peräkkäisiä alhaisia tiloja. Mikäli sekä aurinkopaneelilla että akulla on ollut peräkkäisiä alhaisia jännitteitä, siirtyy laite hätäkäyttötilaan. Hätäkäyttötilassa laite pitää molemmat mosfetit johtavassa tilassa, jotta kaikki saatavilla oleva virta saadaan käyttöön.

Mikäli jännite on alhainen aurinkopaneelilla, mutta riittävä akulla, pysyy laite normaalitilassa, mutta tällöin virta ohjataan kulkemaan akulta. Mikäli jännite puolestaan on alhainen vain akulla, siirtyy laite lataamaan akkua. Akkua ladataan niin kauan, kunnes akun jännite on yli 9 V, eli kun AD-muunnin saa arvon 0xF8. Lataaminen kuitenkin keskeytyy, mikäli akkua on ladattu kahdeksan tuntia.

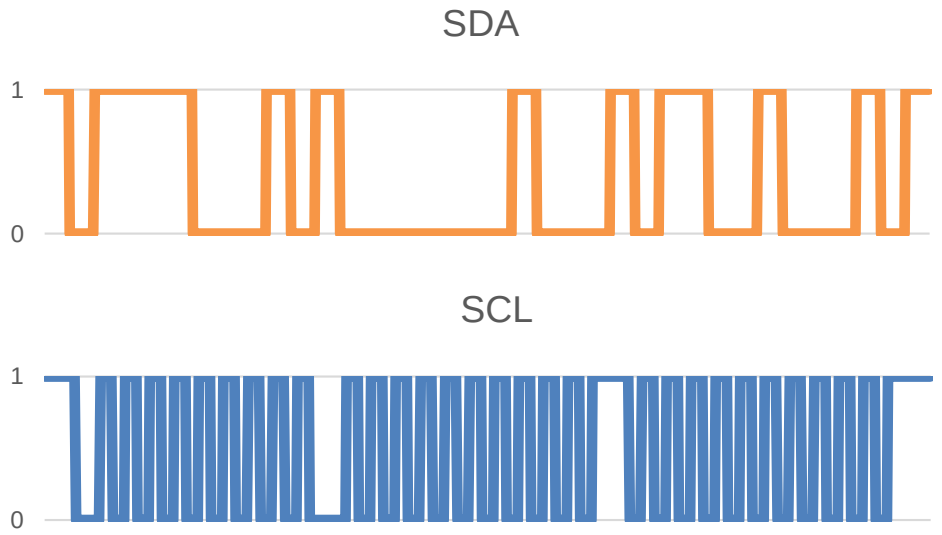
Jos akulla ja aurinkopaneelilla on riittävä jännite, pysyy laite normaalitilassa eli käyttää aurinkopaneelin virtaa.

Mittaustilassa laite suorittaa etäisyydenmittauksen. Etäisyys mitataan vesisäiliön katon ja vedenpinnan välillä. Mittausta varten PSoC kirjoittaa I^2C-väylään: aloitusbitin 0x00, osoitteen 0xF1, eli osoite 0xF0 kirjoitusbitillä, komentorekisterin osoitteen 0x00, senttimetrien mittauspyynnön 0x51 ja pysäytysbitin 0x01. Tämän jälkeen ohjelma odottaa kaksi sekuntia mittauksen valmistumista varten. Kuviossa 2 on havainnollistettu laitteiden välinen viestintä I^2C-väylässä mittauskäskyn aikana.



Kuvio 2. SDA- ja SCL-signaalit mittauskäskyä annettaessa.

Lukeminen tapahtuu kirjoittamalla I^2C-väylään: aloitusbitti 0x00, osoite 0xF1, eli 0xF0 kirjoitusbitillä, lukurekisterin osoite 0x02, toistetun aloituksen bitti 0x00, osoite 0xF0 lukubitillä. Tämän jälkeen ohjelmisto lukee kolme ensimmäistä bittiä ja lähettää niiden kuittaukset. Lopuksi luetaan neljäs bitti ja lähetetään NAK osoituksena lukemisen lopettamisesta. Kuviossa 3 on havainnollistettu isäntä- ja orjalaitteiden viestintä I^2C-väylässä lukupyynnön ja -vastauksen aikana. [11; 21.]



Kuvio 3. SDA ja SCL-signaalit lähetettäessä lukupyyntöä ja luettaessa.

Tiedonkäsittelytilassa ohjelmisto vertaa kalibrointiarvoa mitattuun etäisyyteen ja osoitetaan saadun tuloksen mukaan LED-valoilla vesimäärää. Jos ohjelma saa jostain syystä suurempia etäisyyksiä kuin kalibrointiarvo, se kasvattaa häiriölaskuria.

Häiriönkäsittelytila tarkistaa, ovatko jännitearvot aiheuttaneet häiriöitä ja onko mittaustuloksissa tapahtunut häiriöitä. Jos jompikumpi jännitteistä laskee alhaiseksi tai jos mittaustuloksen häiriöitä on esiintynyt neljä kertaa peräkkäin, syttyy häiriön merkkivalo.

4.4 Aurinkopaneelin ja akuston mitoitus

4.4.1 Sisilian ilmasto

Sisiliassa on aurinkoista keskimäärin 2680 tuntia vuoden 8760 tunnista. Tämä aika vaihtelee siten, että joulukuussa päivää kohden on auringonpaistetta 4,5 tuntia ja heinäkuussa 10,5 tuntia. [22.]

Insolaatioon vaikuttavat päivämäärä, auringon deklinaatio, etäisyys auringosta, leveyspiiri, vuorokaudenaika, auringon zeniitti ja sääolosuhteet [23]. Joulukuussa Sisiliassa Palermon mittauspisteellä insolaatio on:

$$E_{PALERMO} = 1910 \frac{Wh/m^2}{d}$$

Joulukuussa päivän pituus on noin 10 tuntia. Säteilytysvoimakkuudeksi saadaan 191 W/m^2 [24].

4.4.2 Virrankulutus

Laitteen teoreettinen virrankulutus on mikrokontrollerin käyttämän virran ($I_{\mu cMAX}$ = 8 mA) [7, s. 20], SRF02-ultraäänianturin käyttämän virran ($I_{SRF02TYP}$ = 4 mA) [11], MIC4426-ajurin virta ($I_{MIC4426}$ = 4 mA) [8, s. 1], LM780xCT:n virta ($I_{LM7805CTMAX}$ = 8 mA) [10, s. 3, 6] ja LED-valojen käyttämän virran summa. MOSFET:ien virrankulutus on häviävän pieni, eikä sitä oteta huomioon.

Normaalitilassa palaa samanaikaisesti korkeintaan viisi keltaista ja kaksi vihreää LED-valoa. Keltaisen LED-valon kynnysjännite on 2,1 V ja vihreän 2,2 V. Syöttöjännite on 5 V ja tavoiteltava virrankulutus 5 mA, jolloin saadaan etuvastuksien arvoiksi:

$$R_{KELT} = \frac{U_{SYÖTTÖ} - U_{KELT}}{I}$$

$$R_{KELT} = \frac{2{,}9\ V}{5\ mA}$$

$$R_{KELT} = 580\ \Omega$$

$$R_{KELT} = \frac{U_{SYÖTTÖ} - U_{KELT}}{I}$$

$$R_{KELT} = \frac{2{,}9\ V}{5\ mA}$$

$$R_{KELT} = 580\ \Omega$$

Yhteensä seitsemän LED-valon virrankulutus on 35 mA.

Tästä saadaan kokonaisjännitteeksi:

$$I_{kok} = I_{SRF02TYP} + I_{\mu cMAX} + I_{MIC4426} + I_{LM780XMAX} + I_{LED} = 59mA.$$

Laitteen kalibrointivaiheessa palaa korkeimmillaan neljä ylimääräistä LED-valoa, joista kaksi on keltaisia ja kaksi punaisia, jolloin laitteen virrankulutus on 79 mA [25].

Laitteen virrankulutukseksi kalibrointivaiheessa mitattiin 69mA ja korkeimmaksi mittaustilan virrankulutukseksi 55 mA.

4.4.3 Aurinkopaneelin ja akuston valinta

Aluksi käytettäväksi aurinkopaneeliksi valittiin Sanyon AM-5913 -aurinkopaneeli. Paneelin datalehden epäselvyydet aiheuttivat epävarmuutta paneelin tehon riittävyydestä ja päädyin vaihtamaan SP-Elektroniikan myymään 12 V:n aurinkopaneeliin. Tästä paneelista ei kuitenkaan ollut mahdollista saada minkäänlaista datalehteä, joten sen riittävyys perustui yrityksen tekemiin mittauksiin. Suoritin myös itse mittauksia paneelille sen riittävyyden selvittämiseksi.

Taulukko 5. Maaliskuussa mitattuja jännite- ja virta-arvoja.

Päivämäärä	**Jännite**	**Virta**
4.3.2013	14,25 V	32 mA
7.3.2013	14,70 V	49,5 mA
9.3.2013	14,25 V	50 mA

Käyttöön valittu aurinkopaneeli on fyysisiltä mitoiltaan 70mm x 80mm. Näin ollen aurinkopaneelin pinta-ala on:

$$A = 80\,mm \cdot 70\,mm$$
$$A = 56\,mm^2$$

Paneeli on testattu Oulun leveyspiireillä kirkkaassa kevätsäässä, jolloin siitä on saatu 80mA ja 12V ulos [26]. Oulussa huhtikuussa insolaatio on:

$$E_{OULU} = 2090\,\frac{Wh/m^2}{d}$$

Samaan aikaan huhtikuussa Oulussa päivän pituus on 15 tuntia. Säteilytysvoimakkuudeksi on saatu 206 W/m^2 [27]. Testatulle aurinkopaneelille on siis tullut 1,15 W:n teho. Testatun aurinkopaneelin teho on ollut:

$$P_{\mathrm{OULU}} = E_{OULU} \cdot A$$
$$P_{OULU} = 206\,\frac{W}{m^2} \cdot 0{,}0056\,m^2$$
$$P_{OULU} = 1{,}15\,W$$

Mitattu teho on ollut:

$$P_{\mathrm{M\_OULU}} = U_{M\_OULU} \cdot I_{M\_OULU}$$
$$P_{M\_OULU} = 12\,V \cdot 80\,mA$$
$$P_{M\_OULU} = 0{,}96\,W$$

Näin saadaan suuntaa-antavaksi hyötysuhteeksi:

$$\eta = \frac{0{,}96\ W}{1{,}15\ W}$$
$$\eta = 0{,}83$$

Tästä voidaan laskea aurinkopaneelin riittävyys Sisiliassa vuoden aurinkotehottomimpaan aikaan eli joulukuussa.

$$P_{\mathrm{PALERMO}} = E_{PALERMO} \cdot A \cdot \eta$$
$$P_{PALERMO} = 191\frac{W}{m^2} \cdot 0{,}83$$
$$P_{PALERMO} = 0{,}88\ W$$

Normaalitilassa tehonkulutus on:

$$P_{\mathrm{NORM}} = U \cdot I_{NORM}$$
$$P_{NORM} = 12V \cdot 59mA$$
$$P_{NORM} = 0{,}71W$$

Koska tehonkulutus ei ylitä joulukuussa tapahtuvaa tuottoa, vaikuttaa aurinkopaneeli olevan riittävä. Laitteen on kuitenkin oltava äärimmäisen toimintavarma, eikä se saisi sammua. Tästä syystä laitteen on kyettävä toimimaan ainakin 14 tunnin ajanjakso ilman ulkoista virtaa. Näin voidaan mitoittaa akusto seuraavasti:

$$I_{\mathrm{BAT}} = 14h \cdot 59mA$$
$$I_{BAT} = 826mAh$$

Vaikka akun kapasiteetin on oltava 826 mAh, jotta saadaan 14 tunnin yhtäjaksoinen toiminta-aika ilman ulkoista virtalähdettä, valitsin käytettäväksi kolme 250 mAh / 9 V akkua. Akut kytketään rinnakkain, jotta saadaan 750 mAh kapasiteetti. Laite ei todennäköisesti ole koko aikaa enimmäiskulutuksella, jolloin todellinen toiminta-aika muodostuu vaadittua pidemmäksi. Akun elektrodimateriaaliksi valikoitui nikkeli-metallihydridi. Nikkeli-metallihydridi ei aiheuta vastaavaa palovaaraa kuin litiumpolymeeri- tai nikkeli-kadmium-akut.

Nikkeli-metallihydridi-akun lataus on akun käyttöiän kannalta turvallisinta 10 %–20 % latausvirralla akun virtakapasiteetista. Akun virtakapasiteetti on 750mAh, jolloin latausvirran tulee olla 75 mA–150 mA. [28.] Heinäkuussa Palermossa auringon intensiteetti on:

$$E_{PALERMO} = 7790\ \frac{Wh/m^2}{d}$$

Heinäkuussa päivän pituus on 14 tuntia 25 minuuttia, jolloin säteilytysvoimakkuudeksi saadaan 540 W/m^2 [24].

$$P_{PALERMO} = I_{PALERMO} \cdot A \cdot \eta$$
$$P_{PALERMO} = 540\ W/m^2 \cdot 0{,}0056\ m^2 \cdot 0{,}83$$
$$P_{PALERMO} = 2{,}51\ W$$

Heinäkuussa latausvirta olisi siis:

$$I_{HEINÄKUU} = \frac{P_{PALERMO}}{U}$$
$$I_{HEINÄKUU} = \frac{2{,}51\ W}{12\ V}$$
$$I_{HEINÄKUU} = 209\ mA$$

Jotta akkua ei pilattaisi liian korkealla virralla, on aurinkopaneelin virtaliittimen jälkeen sijoitettava etuvastus rajoittamaan virran kulkua.

$$R_{ETU} = \frac{U}{I}$$
$$R_{ETU} = \frac{12\ V}{150\ mA}$$
$$R_{ETU} = 80\ \Omega$$

Koska nikkeli-metalli-hydridi-akkua ei saisi ladata yli kahdeksaa tuntia, keskeytetään akun lataaminen ohjelmistolla, mikäli akun lataaminen on kestänyt yli kahdeksan tuntia. Tämän jälkeen lataaminen aloitetaan kuitenkin uudelleen, mikäli jännite on alhainen.

5 Kokoonpano, testaus ja käyttöönotto

Laitteen kokoonpano oli projektin haastavin osa-alue. Ensimmäiset piirilevyt jyrsin Metropolia Ammattikorkeakoulun Leppävaaran toimipisteen LPKF ProtoMat S63 -piirilevyjyrsimellä. Lopullisen piirilevyn jyrsin Albertinkadun toimipisteen LPKF ProtoMat S43 -piirilevyjyrsimellä.

Komponenttien juottaminen ahtaalle piirilevylle vaati tarkkaa suunnittelua, sillä komponentit olivat korkeuksiltaan varsin erilaisia. Koska juotettavia komponenttijalkoja oli lähes 200, oli virheen mahdollisuus hyvin todennäköinen. Lopullisiin testeihin päätynyt piirilevy säästyi virheiltä lähes täysin, ja syntyneetkin oli mahdollista korjata. Erityisen vaikeaa oli juottaa 12 millsin leveydellä jyrsittyihin johtimiin, koska johtimien kupari irtosi todella helposti piirilevyn lasikuitumateriaalista.

Laitteen lopullisia testejä suoritin monin tavoin. Laitteen suorituskykyä selvittääkseni käytin laitetta huhtikuun alussa pelkästään aurinkopaneelin avulla. Laite toimi aurinkoisena huhtikuun päivänä Suomessa noin neljän tunnin ajan päivän aurinkoisimpaan aikaan. Pilvisenä päivänä laite onnistui sytyttämään virtatilaa osoittavan vihreän LED-valon, mutta ei muuta.

Laitteen toiminnan yleisen vakauden testaamista varten kytkin laitteen tehonlähteeseen, jonka kilpiarvoina oli 12 V ja 1 A. Luotettavassa tehonlähteessä ollessaan laite toimi moitteettomasti keskeytyksettä viikon pituisen jakson. Tuona aikana laitteen häiriötä osoittava punainen LED-valo syttyi muutaman kerran, mutta häiriö poistui muutaman sekunnin jälkeen. Oletan häiriöiden syynä olleen hetkellisen mittaushäiriön. Mittalaite oli tuettu osoittamaan seinälle, jolloin ohitse kulkevat henkilöt aiheuttivat mittaustuloksiin vaihtelevuutta.

Virrankäyttöä testasin myös akkujen kanssa, mutta akkukäytöllä laite toimi vain neljä tuntia. Ero on merkittävä suunniteltuun 13 tuntiin, joten otin asian tutkiakseni. Ero johtuu todella suuresta jännitehäviöstä, mikä piirilevyllä aiheutuu akun ja PSoC:n välillä. Akun napojen välisen jännitteen ollessa 9,4 V, on PSoC:in käyttöjännitepinnin ja maapotentiaalin välillä vain 3,5 V. PSoC toimii vielä 3,0 V:in asti, jolloin neljän tunnin aikana tapahtuva jännitteenlasku sammuttaa PSoC:in ja estää näin laitteen toiminnan.

Samassa yhteydessä havaitsin myös suunnitteluvirheen: vihreä virtatilaa osoittava LED-valo on kytketty LM7805:n ulostulona olevaan 5 V:n jänniteverkkoon. Tästä johtuen vihreä LED-valo palaa niin kauan, kun akulta tai aurinkopaneelilta tulee pienikin jännite ja vähänkin virtaa. Vihreä LED-valo palaa myös siinä tapauksessa, kun aurinkopaneelilta tai akulta riittää virtaa, mutta säiliö on tyhjä. Tästä johtuen on mahdotonta tietää, onko laite vielä toiminnassa vai onko säiliö vain tyhjentynyt. Tämän vuoksi lisäsin alkuperäiseen ohjelmakoodiin tyhjää säiliötä osoittamaan vilkkuvan häiriötilaa osoittavan punaisen LED-valon.

6 Johtopäätökset

6.1 Laitteen asentaminen

Aurinkopaneeli on sijoitettava mahdollisimman kohtisuorasti keskipäivän aurinkoa kohden. Suuntaamista varten on laitteessa sarana, joka on mahdollista kiristää tiettyyn asentoon.

Mittalaitteen asennuspaikka on valittava vesisäiliön päältä mahdollisimman keskeltä, mutta toisaalta mahdollisimman kaukaa veden syöttöaukosta. Laitteen on kuitenkin oltava noin 40 cm:n etäisyydellä säiliön laidoista, jotta keila heijastuisi suoraan veden pinnasta eikä säiliön kyljistä.

LED-paneeli on mahdollista sijoittaa käytännössä minne vain. CAT-5-verkkokaapelissa tapahtuva jännitehäviö on olematon kaapelin pienen vastuksen takia. Paneelia ei tosin kannata sijoittaa paikkaan, missä se on suorassa auringonvalossa, sillä LED-merkkivalojen lukeminen on tällöin pienen virrankäytön vuoksi hankalaa.

6.2 Laitteen toiminta

Testeissä havaitsin laitteen toiminnan olevan vakaata. Varmassa virtalähteessä laite toimi virheettömästi viikon kestävän keskeytyksettömän testijakson ajan. Häiriötilasta kertova punainen LED-valo syttyi harvakseltaan, mutta katosi yleensä muutamassa se-

kunnissa. Laite ei kuitenkaan ollut ympärivuorokautisessa valvonnassa, joten se oli saattanut tehdä jotain odottamatonta, mutta siitä ei saatu viitettä. Mikäli virheitä on sattunut, niin virheistä palautuminen on toiminut tällöin erinomaisesti.

Laitteen virrankulutus oli myös odotetusti todella vähäinen, minkä ansiosta sen käyttäminen on mahdollista suhteellisen pienellä aurinkokennolla. Suomen pituuspiireillä tehdyt testit antoivat jo positiivista suuntaa laitteen käytöstä.

Akkukäytön heikko kestävyys oli takaisku, mutta laite toimii kuitenkin erinomaisesti silloin, kun sitä eniten tarvitaan eli päivällä. Takuuvarmaksi laite ei tämän työn puitteissa muodostunut, mutta olettaakseni pienellä kehitystyöllä akkukäyttö olisi mahdollista saada toimimaan.

6.3 Laitteen kehittäminen

Laitteen kehittämiseen tuli insinöörityön aikana paljon ajatuksia. Ensisijainen kehityskohde olisi selvittää ja korjata akkukäyttöä haittaava jännitehäviö. Myös laitteen virtatilaa osoittava vihreä LED-valo tulisi kytkeä ohjautumaan PSoC:n pinneistä, koska näin laite kertoisi itse olevansa toimintakykyinen.

Seuraava kehityssuunta olisi irtautua SRF02-ultraäänianturin käyttämisestä, sillä oman ultraäänianturin valmistaminen suoraan piirilevylle olisi todennäköisesti pitkällä aikavälillä kustannustehokkaampi ratkaisu, kuin valmiin anturikomponentin käyttäminen. Anturikomponentin yhteyteen voisi lisätä lämpötila-anturin, jotta kylmän ja kuuman sään aiheuttamat mittausvirheet tulisivat huomioitua.

Myös tiedon esittämistä pelkkien LED-valojen sijaan olisi mahdollista jalostaa. Sijoittamalla infopaneeliin toinen PSoC saataisiin esimerkiksi huomioääniä tai LCD-näyttö käyttöön. Infopaneelia voisi olla myös mahdollista käyttää kalibrointivaiheessa tiedon esittämiseen selkeämmin. Tämä mahdollistaisi myös useamman parametrin helpomman muutoksen, jolloin esimerkiksi kyljellään olevan sylinterin mallisen säiliön käyttö mahdollistuisi. Tällöin paneelin ja mittauslaitteen välinen kommunikointi voitaisiin toteuttaa rinnakkaisliikenteen sijaan sarjaliikenteenä, esimerkiksi I^2C-väylällä. Tämä kehittyneempi paneeli olisi mahdollista myös kytkeä USB-väylän ylitse PC-tietokoneeseen ja eteenpäin esimerkiksi www-palvelimelle. Tietoa olisi myös mahdollista käsitellä LabViewn avulla

tietokoneella. Koska CY8C27443PXI ei tue USB-väylän käyttöä, olisi laitteessa siirryttävä käyttämään toista mallia. Jotta laitteesta saataisiin kehitettyä todellinen tuote, olisi sille myös tehtävä CE-merkinnän edellyttävät testaukset.

Pohdin myös laitteelle muita käyttökohteita ja sainkin siihen hyviä ehdotuksia. Suomen oloissa laitetta olisi mahdollista käyttää esimerkiksi järvien vedenpinnan korkeuden seurantaan. Myös esimerkiksi kaivoveden varassa olevilla kesämökeillä uskoisin löytyvän jossain määrin kiinnostusta veden määrää kohtaan, etenkin jos vesi tulee pumpun avulla mökkiin. Sain myös tiedusteluja, olisiko laite mahdollista asettaa ohjaamaan esimerkiksi pumppuja tai muita laitteita, kun tietty raja-arvo on saavutettu. Uskon, että laitetta olisi mahdollista jalostaa hyvinkin monenlaisiin tarkoituksiin.

Lähteet

1. Airaksinen, Harri. 2012.

2. Ultrasonic Basics. 2013. Verkkodokumentti. Banner Engineering. <http://www.bannerengineering.com/training/faq.php?faqID=34&div=1>. Luettu 9.3.2013.

3. Infrared vs. ultrasonic - What you should know. 2013. Verkkodokumentti Narobo, Society of Robots. <http://www.societyofrobots.com/member_tutorials/node/71>. Luettu 7.3.2013.

4. IP Classification Codes. 2004. Verkkodokumentti. Bourns, INC. <http://www.newenglandlaser.com/attachments/File/pdf/IP_codes.pdf>. Luettu 5.3.2013.

5. Liitäntäjohto MSO. 2013. Verkkodokumentti. Clas Ohlson. <http://www.clasohlson.com/fi/Liit%C3%A4nt%C3%A4johto-MSO/Pr490191050>. Luettu 11.3.2013.

6. DC-Liitin 2,1 / 5,5mm. 2008. Verkkodokumentti. Partco OY. <http://www.partco.biz/verkkokauppa/product_info.php?products_id=8668>. Luettu 4.9.2013.

7. CY827443 Datasheet. 2012. Verkkodokumentti. Cypress Semiconducters. <http://www.cypress.com/?docID=35973>. Luettu 24.10.2012.

8. MIC3326/4427/4428 Datasheet. 2008. Verkkodokumentti. Micrel. <http://www.micrel.com/_PDF/mic4426.pdf>. Luettu 2.4. 2013.

9. IRF9530 Datasheet. 2002. Verkkodokumentti. Fairchild semiconductors. <http://www.datasheetcatalog.org/datasheet/fairchild/IRF9530.pdf>. Luettu 20.1.2013.

10. LM780x Datasheet. 2012. Verkkodokumentti. Fairchild Semiconductors. <http://www.fairchildsemi.com/ds/LM/LM7805.pdf>. Luettu 20.1.2013.

11. SRF02 Datasheet. 2012. Verkkodokumentti. Devantech LTD. <http://www.robot-electronics.co.uk/htm/srf02tech.htm>. Luettu 20.11.2012.

12. Overview for Applying Ultrasonic Technology. 2013. Verkkodokumentti. Airmar Technology Corporation. <http://airmartechnology.com/uploads/AirPDF/Intro_Overview.pdf>. Luettu 20.3.2013.

13. Devantech SRF02 Sensor. 2013. Verkodokumentti. Manu Systems. <http://en.manu-systems.com/SRF02.shtml>. Luettu 9.3.2013.

14. HyperPhysics. 2013. Verkkodokumentti. Georgia State University. <http://hyperphysics.phy-astr.gsu.edu/%E2%80%8Chbase/sound/souspe.html>. Luettu 9.3.2013.

15. Air Pressure Calculator. 2013. Verkkodokumentti. Sengpielaudio. <http://www.sengpielaudio.com/calculator-airpressure.htm>. Luettu 5.4. 2013

16. I2C-Bus: What's that? 2013. Verkkodokumentti. IIC Bus. <http://www.i2c-bus.org/>. Luettu 10.3.2013.

17. The I2C-bus specification. 2001. Verkkodokumentti. Philips Semiconductors. <http://www.nxp.com/documents/other/39340011.pdf>. Luettu 10.3.2013.

18. Forum. 2013. Verkkodokumentti. Devantech LTD. <http://www.robot-celectronics.co.uk/forum/viewtopic.php?f=2&t=630>. Luettu 9.1.2013.

19. Thermal Resistance Theory and Practice. 2000. Verkkodokumentti. Infineon Technologies AG. <http://www.infineon.com/dgdl/smdpack.PDF?folderId=db3a304412b407950112b417b3e623f4&fileId=db3a304412b407950112b417b42923f5>. Luettu 10.4.2013.

20. PSoC 1 GPIO Demystified. 2009. Verkkodokumentti. Cypress Semiconductors. <http://www.cypress.com/?rID=39496>. Luettu 20.12.2012.

21. Using the I2C Bus. 1013. Verkkodokumentti. Devantech LTD. <http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html>. Luettu 2.9.2013.

22. Tabella CLINO. 2012. Verkkodokumentti. Aeronautica Miltare Servizio Meterlologico. <http://clima.meteoam.it/viewClino.php?type=File&station=429&name_station=Trapani%20Birgi>. Luettu 10.11.2012.

23. Insolation. 2012. Verkkodokumentti. University of California Department of Geography. <http://www.geog.ucsb.edu/ideas/Insolation.html>. Luettu 5.12.2012.

24. Gaisma - Palermo. 2012. Verkkodokumentti. Tukiainen, Matti. <http://www.gaisma.com/en/location/palermo.html>. Luettu 20.12.2012.

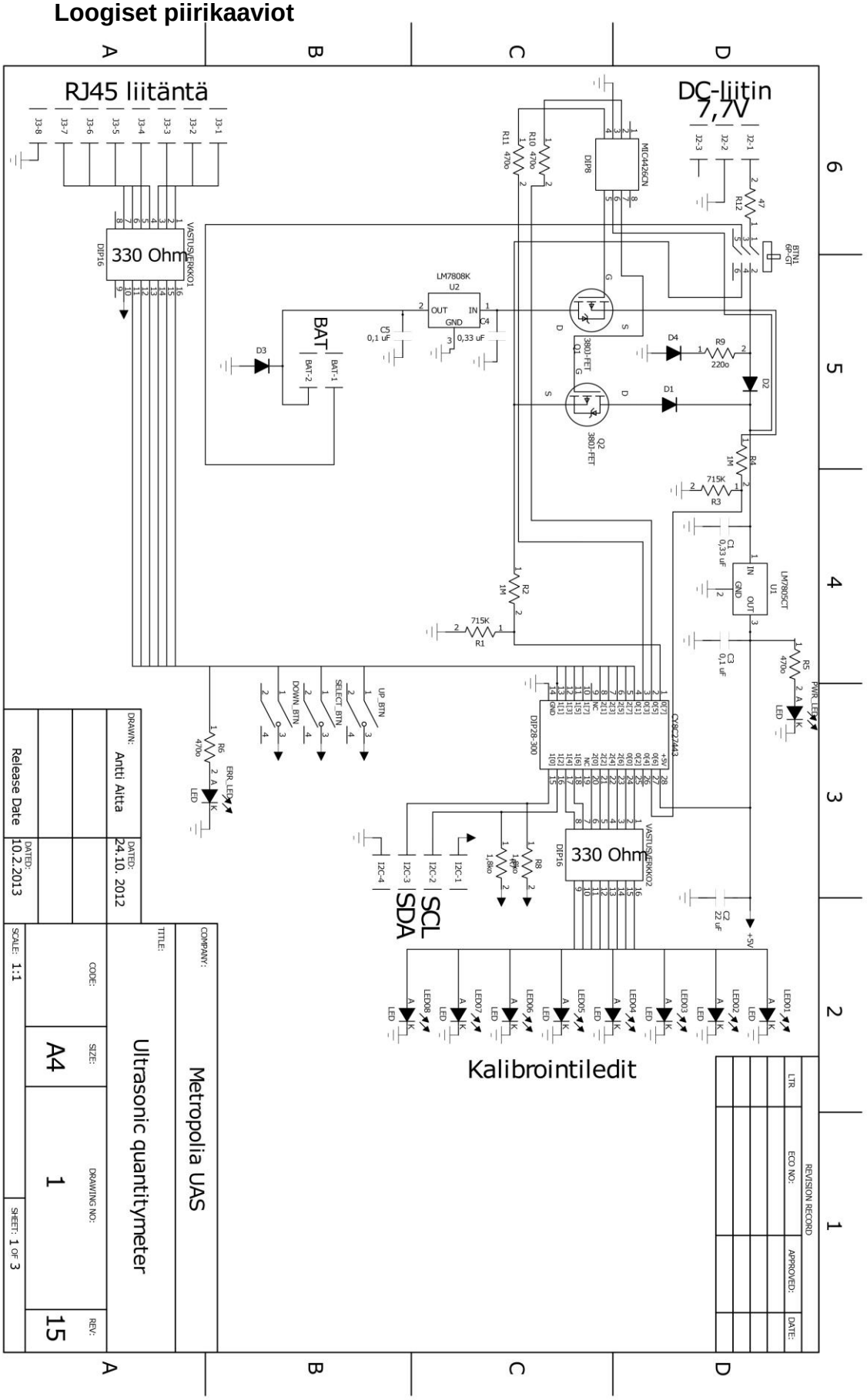
25. Harraste Elektroniikka. 2007. Verkkodokumentti. Hutamo, Kari. <http://koti.mbnet.fi/~huhtama/ele/index.php?si=ml28.sis>. Luettu 12.12.2012.

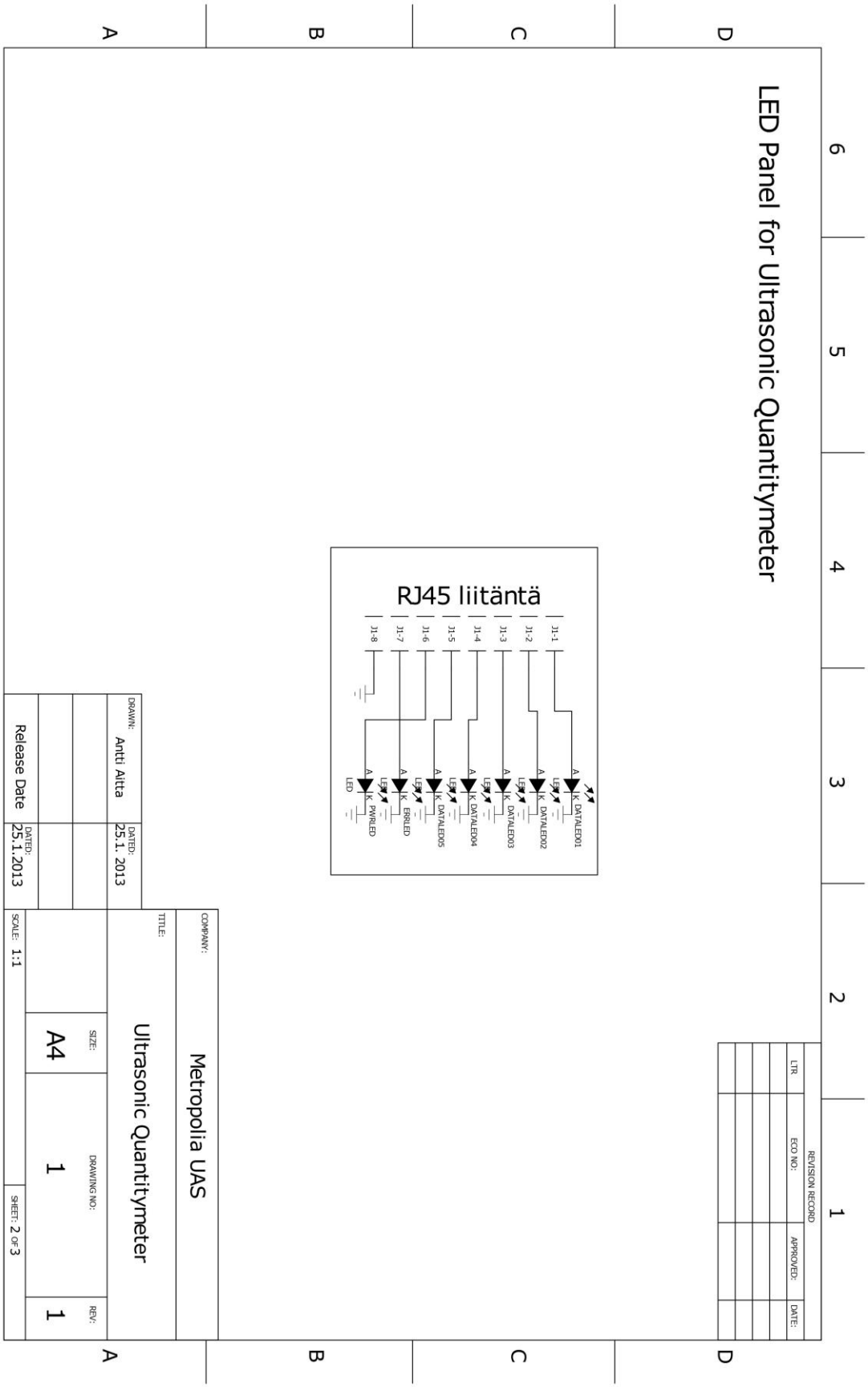
26. Aurinkokenno 12V/0.5W. 2013. Verkkodokumentti. SP-Elektroniikka. <http://www.spelektroniikka.fi/tuotteet/elektroniikka-verkkolaitteet-aurinkopaneelit-/aurinkokenno-12v05w-100374>. Luettu 10.2.2013.

27. Gaisma - Oulu. 2012. Verkkodokumentti. Tukiainen, Matti. <http://www.gaisma.com/en/location/oulu.html>. Luettu 20.12.2012.

28. Charging Nickel-Metal-Hydride. 2011. Verkkodokumentti. Battery University. <http://batteryuniversity.com/learn/article/charging_nickel_metal_hydride>. Luettu 21.2.2013.

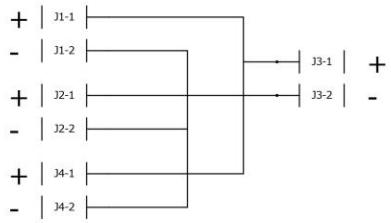
Loogiset piirikaaviot




LED Panel for Ultrasonic Quantitymeter
RJ45 liitäntä
J1-1
J1-2
J1-3
J1-4
J1-5
J1-6
J1-7
J1-8
DATALED01
DATALED02
DATALED03
DATALED04
DATALED05
ERRLED
PWRLED
LED
REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
COMPANY:
Metropolia UAS
TITLE:
Ultrasonic Quantitymeter
DRAWN:
Antti Aitta
DATED:
25.1. 2013
Release Date
DATED:
25.1.2013
SIZE:
A4
DRAWING NO:
1
REV:
1
SCALE: 1:1
SHEET: 2 OF 3

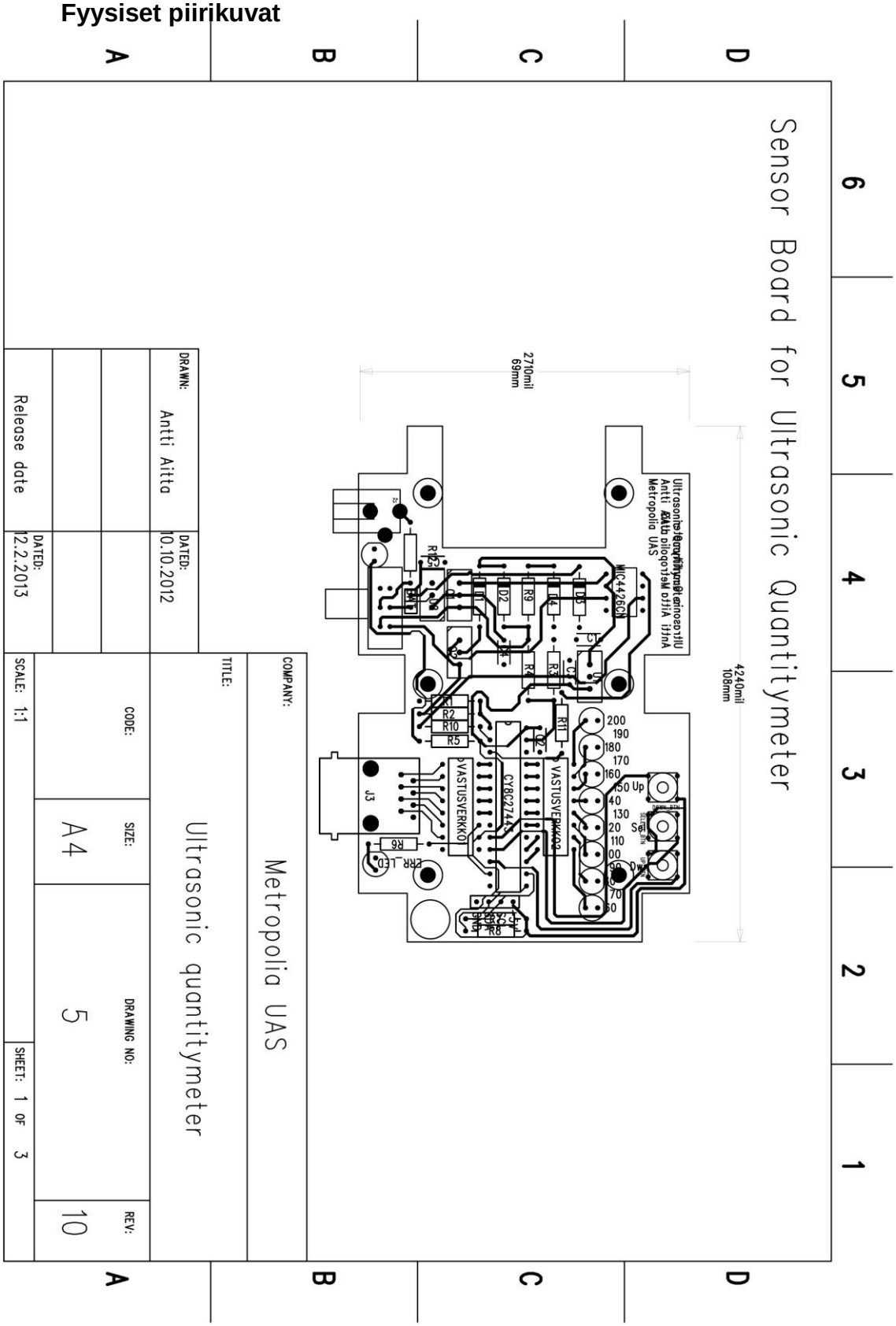
Batteryholder for Ultrasonic Quantitymeter

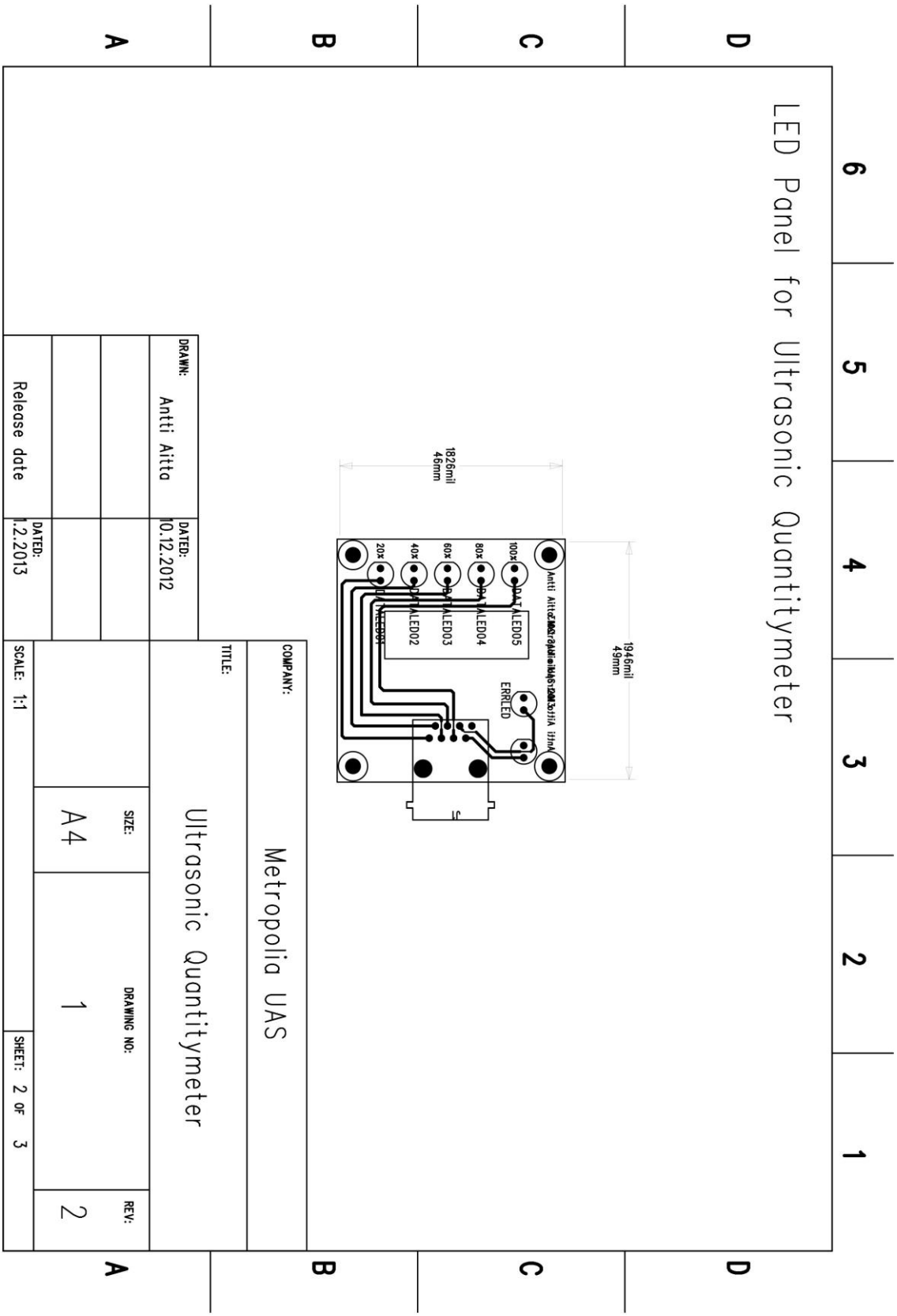
REVISION RECORD			
LTR	ECO NO:	APPROVED:	DATE:



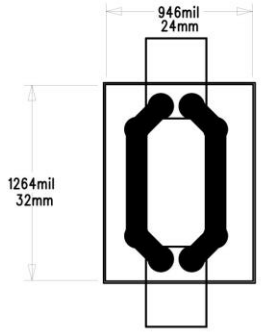
DRAWN: Antti Aitta	DATED: 8.2.2013	COMPANY: Metropolia UAS	
		TITLE: Ultrasonic quanitymeter	
		SIZE: A4	DRAWING NO: 1 / REV: 1
Release Date	DATED: 8.2.2013	SCALE: 1:1	SHEET: 3 OF 3

Fyysiset piirikuvat



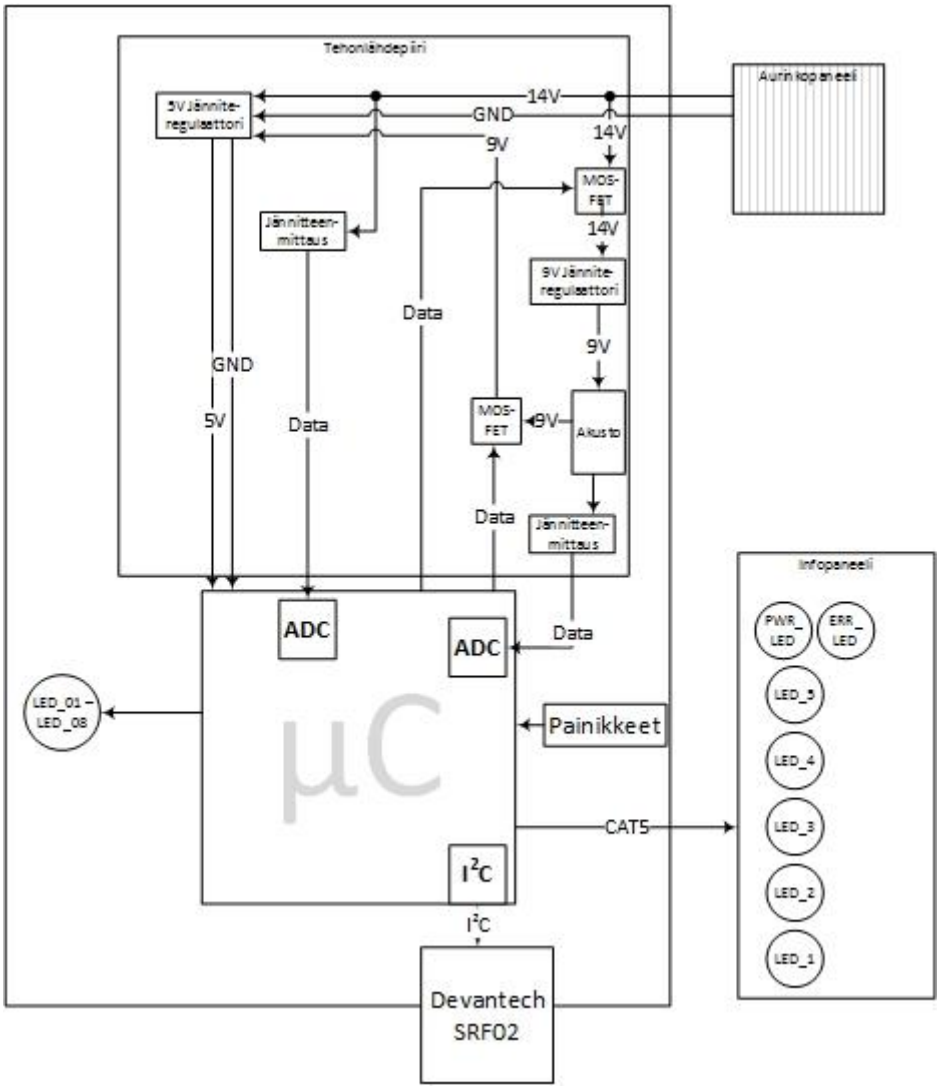

LED Panel for Ultrasonic Quantitymeter
1826mil
46mm
1946mil
49mm
100x
80x
60x
40x
20x
DATALED05
DATALED04
DATALED03
DATALED02
ERRLED
DRAWN:
Antti Aitta
DATED:
10.12.2012
Release date
DATED:
1.2.2013
COMPANY:
Metropolia UAS
TITLE:
Ultrasonic Quantitymeter
SIZE:
A4
DRAWING NO:
1
REV:
2
SCALE: 1:1
SHEET: 2 OF 3

Batteryholder for Ultrasonic Quantitymeter

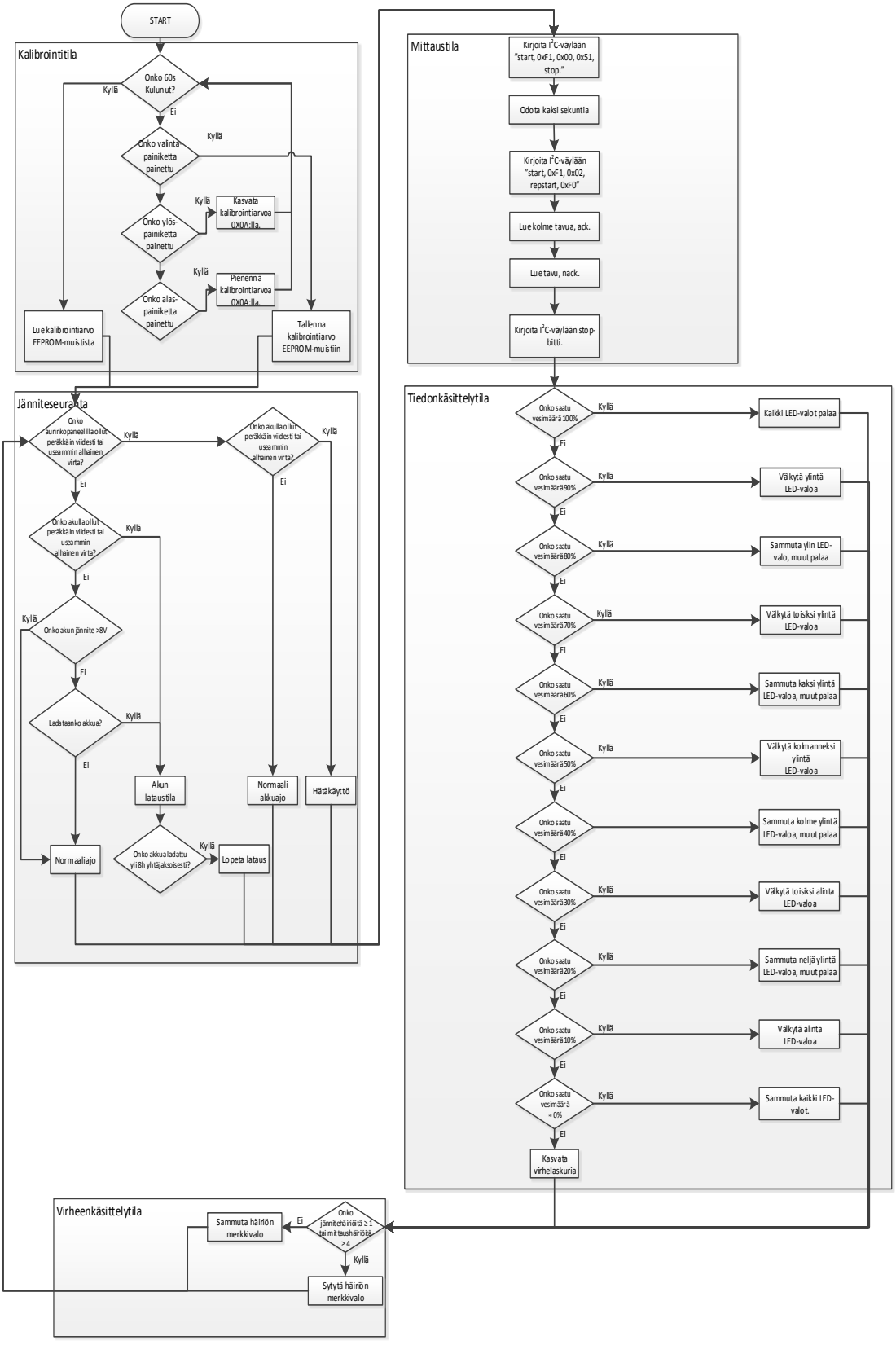


DRAWN: Antti Aitta	DATED: 10.2.2013	COMPANY: Metropolia UAS			
		TITLE: Ultrasonic Quantitymeter			
			SIZE: A4	DRAWING NO: 1	REV: 1
Realease Date	DATED: 10.2.2013	SCALE: 1:1		SHEET: 3 OF 3	

Laitteen lohkokaavio



Ohjelmakoodin vuokaavio



Ohjelmakoodi

```
//-------------------------------------------------------------------------------
// Solar powered ultrasonic water quantitymeter for water tanks
// PROGRAMMER:   Antti Aitta Metropolia UAS
// VERSION:                      1.40 FINAL
// DATE:                         11.4. 2013
//-------------------------------------------------------------------------------

#include <m8c.h>                                                    // part specific constants
#include <PSoCAPI.h>                                                // PSoC API definitions
#include <PSOCGPIOINT.h>
#include <stdio.h>
#include <stdlib.h>
#pragma interrupt_handler btnISR timeISR

/* Prototypes */
void Calibration(void);
void StartModules(void);
void VoltageControl(void);
void BatteryCheck(void);
void SolarCheck(void);
void Charging(void);
void BatteryUse(void);
void SolarUse(void);
void EmergencyUse(void);
void Measure(void);
void btnISR(void);
void timeISR(void);
void DataHandler(void);
void ErrorHandler(void);

/* Global variables */
unsigned char batstate = 0x00;
unsigned char solstate = 0x00;
unsigned char SolarVoltage = 0x00;
unsigned char BatteryVoltage = 0x00;
unsigned char ChargingFlag = 0x00;
unsigned char ChargingCounter = 0x00;
unsigned char SelectFlag = 0x00;
unsigned char ContinueFlag = 0x00;
unsigned char EmergencyFlag = 0x00;
unsigned char VoltageError = 0x00;
unsigned char MeasurementError = 0x00;
unsigned char i = 4;                                                // 4 bytes to be read
unsigned char height = 0x06;
unsigned char ErrorCounter = 0x00;
unsigned char flash;
unsigned char hh = 0x00;
unsigned char mm = 0x00;
unsigned char ss = 0x00;
unsigned char ms = 0x00;
BYTE X[4] = {0x00, 0x00, 0x00, 0x00};

/* Main starts here */
void main(void) {
  StartModules();
  M8C_EnableGInt;
  M8C_EnableIntMask(INT_MSK0 , INT_MSK0_GPIO);
  Timer8_EnableInt();
  SleepTimer_EnableInt();
  SleepTimer_SetInterval(SleepTimer_64_HZ);
  BATVPGA_SetGain(BATVPGA_G1_00);
  BATVPGA_Start(BATVPGA_MEDPOWER);
  BATVADC_Start(BATVADC_HIGHPOWER);                                 // Turn on Analog section
  BATVADC_SetResolution(8);                                         // Set resolution to 8 Bits
  BATVADC_GetSamples(0);                                            // Start ADC to read continuously
  SOLVPGA_SetGain(SOLVPGA_G1_00);
  SOLVPGA_Start(SOLVPGA_MEDPOWER);
  SOLVADC_Start(SOLVADC_HIGHPOWER);                                 // Turn on Analog section
  SOLVADC_SetResolution(8);                                         // Set resolution to 8 Bits
  SOLVADC_GetSamples(0);                                            // Start ADC to read continuously
  Calibration();                                                    // Calibrate the tank height
  while(1){                                                         // Main loop starts here
  VoltageControl();                                                 // Check for voltages
  Measure();                                                        // Measure distance
  DataHandler();                                                    // Handle data acquired and show it
  ErrorHandler();                                                   // Handle errors
  }                                                                 // Main loop ends here
}                                                                   // Main ends here
```

```
void Calibration(void){                                         // Calibrate the tank height
  ErrLED_Switch(0);                                             // Turn on Error LED
  while(SelectFlag == 0x00 && ContinueFlag == 0x00){
    switch(height){

      case 0x14:                                                // Height at 200cm (0x14)
        LED01_Switch(0);                                        // Start LED01
        LED02_Switch(1);
        LED03_Switch(1);
        LED04_Switch(1);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(1);
      break;

      case 0x13:                                                // Height at 190cm (0x13)
        LED01_Switch(0);                                        // Start LED02 and LED01
        LED02_Switch(0);
        LED03_Switch(1);
        LED04_Switch(1);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(1);
      break;

      case 0x12:                                                // Height at 180cm (0x12)
        LED01_Switch(1);                                        // Start LED02
        LED02_Switch(0);
        LED03_Switch(1);
        LED04_Switch(1);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(1);
      break;

      case 0x11:                                                // Height at 170cm (0x11)
        LED01_Switch(1);                                        // Start LED03 and LED02
        LED02_Switch(0);
        LED03_Switch(0);
        LED04_Switch(1);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(1);
      break;

      case 0x10:                                                // Height at 160cm (0x10)
        LED01_Switch(1);                                        // Start LED03
        LED02_Switch(1);
        LED03_Switch(0);
        LED04_Switch(1);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(1);
      break;

      case 0x0F:                                                // Height at 150cm (0x0F)
        LED01_Switch(1);                                        // Start LED04 and LED03
        LED02_Switch(1);
        LED03_Switch(0);
        LED04_Switch(0);
        LED05_Switch(1);
        LED06_Switch(1);
         LED07_Switch(1);
         LED08_Switch(1);
       break;

       case 0x0E:                                               // Height at 140cm (0x0E)
        LED01_Switch(1);                                        // Start LED04
        LED02_Switch(1);
        LED03_Switch(1);
        LED04_Switch(0);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(1);
       break;
```

```
case 0x0D:                                          // Height at 130cm (0x0D)
  LED01_Switch(1);                                  // Start LED05 and LED04
  LED02_Switch(1);
  LED03_Switch(1);
  LED04_Switch(0);
  LED05_Switch(0);
  LED06_Switch(1);
  LED07_Switch(1);
  LED08_Switch(1);
break;

case 0x0C:                                          // Height at maximum 120cm (0x0C)
  LED01_Switch(1);                                  // Start LED05
  LED02_Switch(1);
  LED03_Switch(1);
  LED04_Switch(1);
  LED05_Switch(0);
  LED06_Switch(1);
  LED07_Switch(1);
  LED08_Switch(1);
break;

case 0x0B:                                          // Height at 110cm (0x0B)
  LED01_Switch(1);                                  // Start LED06 and LED05
  LED02_Switch(1);
  LED03_Switch(1);
  LED04_Switch(1);
  LED05_Switch(0);
  LED06_Switch(0);
  LED07_Switch(1);
  LED08_Switch(1);
break;

case 0x0A:                                          // Height at 100cm (0x0A)
  LED01_Switch(1);                                  // Start LED06
  LED02_Switch(1);
  LED03_Switch(1);
  LED04_Switch(1);
  LED05_Switch(1);
  LED06_Switch(0);
  LED07_Switch(1);
  LED08_Switch(1);
break;

case 0x09:                                          // Height at 90cm (0x09)
   LED01_Switch(1);                                 // Start LED07 and LED06
   LED02_Switch(1);
   LED03_Switch(1);
   LED04_Switch(1);
   LED05_Switch(1);
   LED06_Switch(0);
   LED07_Switch(0);
   LED08_Switch(1);
 break;
 case 0x08:                                         // Height at 80cm (0x08)
   LED01_Switch(1);                                 // Start LED07
   LED02_Switch(1);
   LED03_Switch(1);
   LED04_Switch(1);
   LED05_Switch(1);
   LED06_Switch(1);
   LED07_Switch(0);
   LED08_Switch(1);
 break;

 case 0x07:                                         // Height at 70cm (0x07)
   LED01_Switch(1);                                 // Start LED08 and LED07
   LED02_Switch(1);
   LED03_Switch(1);
   LED04_Switch(1);
   LED05_Switch(1);
   LED06_Switch(1);
   LED07_Switch(0);
   LED08_Switch(0);
 break;

 case 0x06:                                         // Height at minimum 60cm (0x06)
   LED01_Switch(1);                                 // Start LED08
   LED02_Switch(1);
   LED03_Switch(1);
```

```
        LED04_Switch(1);
        LED05_Switch(1);
        LED06_Switch(1);
        LED07_Switch(1);
        LED08_Switch(0);
      break;

      default:
        ErrLED_Switch(0);                                   // Switch to error state
    }
  }

  if(SelectFlag == 0x00){
    E2PROM_E2Read( 0x0000, &height, 1 );
  }

  LED01_Switch(1);                                          // Turn off LEDs and  DATALEDs
  LED02_Switch(1);
  LED03_Switch(1);
  LED04_Switch(1);
  LED05_Switch(1);
  LED06_Switch(1);
  LED07_Switch(1);
  LED08_Switch(1);
  DATALED01_Switch(1);
  DATALED02_Switch(1);
  DATALED03_Switch(1);
  DATALED04_Switch(1);
  DATALED05_Switch(1);
  Timer8_Stop();
  E2PROM_bE2Write(0x0000, &height, 1, 25);
}

void VoltageControl(void){                                  // Do battery and solar voltage
  BatteryCheck();                                           // check
  SolarCheck();


  if(SolarVoltage < 0x8A) {                                 // Counting following states where
    solstate++;                                             // solar panel voltage is low
  } else {
    solstate = 0x00;
  }

  if(BatteryVoltage < 0xF7) {                               // Counting following states where
  batstate++;                                               // battery voltage is low
  } else {
    batstate = 0x00;
  }

  if(solstate >= 0x01){                                     // Does solpanel have low voltage
    if(batstate >= 0x01){                                   // Does battery have low voltage
      EmergencyUse();                                       // Battery voltage low, emergency!
    } else {
      BatteryUse();                                         // Battery voltage normal,
    }                                                       // Normal battery use!
  }

  else if(batstate >= 0x01) {
    Charging();                                             // SolV normal, BatV low, recharge
  }
  else if(BatteryVoltage > 0xF0)                            // SolV normal, BatV normal,
    SolarUse();                                             // Normal battery use!
  }

  else if(ChargingFlag == 0x01){                            // Should we be recharging?
    Charging();
  } else {
    SolarUse();                                             // Normal solar use
  }
}

void BatteryCheck(void){
  if (mm == 0x3B && ChargingFlag == 0x01 && EmergencyFlag == 0x00) {     // Hourly BatteryCheck
    MOSFET1_Switch(1);                                      // Turn off MOSFET1
    MOSFET2_Switch(1);                                      // Turn off MOSFET2
    while(BATVADC_fIsDataAvailable() == 0);
      BatteryVoltage = BATVADC_iGetData();                  // Get Data
    BATVADC_ClearFlag();                                    // Clear data ready flag
  } else if (ChargingFlag == 0x00 && EmergencyFlag == 0x00){            // BtCheck if !Charging & !Emerg
    while(BATVADC_fIsDataAvailable() == 0);
```

```
      BatteryVoltage = BATVADC_iGetData();                    // Get Data
    BATVADC_ClearFlag();
  }

  if(ChargingFlag == 0x01 && hh > 0x07){                      // Prevent overcharging
    ChargingFlag = 0x00;
    MOSFET1_Switch(1);                                        // Turn off MOSFET1
    MOSFET2_Switch(1);                                        // Turn off MOSFET2
  }
}

void SolarCheck(void){
  while(SOLVADC_fIsDataAvailable() == 0);
  SolarVoltage = SOLVADC_iGetData();                          // Get Solar panel Data
  SOLVADC_ClearFlag();                                        // Clear data ready flag

}


void Charging(void){                                          // Normal recharging mode
  MOSFET1_Switch(0);                                          // Turn on MOSFET1
  MOSFET2_Switch(1);                                          // Turn off MOSFET2
  VoltageError = 0x00;
  ChargingFlag = 0x01;
}

void BatteryUse(void){                                        // Normal battery usage
  MOSFET1_Switch(1);                                          // Turn off MOSFET1
  MOSFET2_Switch(0);                                          // Turn on MOSFET2
  VoltageError = 0x00;                                        // Switch to normal state
  ChargingFlag = 0x00;
  EmergencyFlag = 0x00;
}

void SolarUse(void){                                          // Normal solar panel usage
  MOSFET1_Switch(1);                                          // Turn off MOSFET1
  MOSFET2_Switch(1);                                          // Turn off MOSFET2
  VoltageError = 0x00;                                        // Switch to normal state
  ChargingFlag = 0x00;
  EmergencyFlag = 0x00;
}

void EmergencyUse(void){                                      // Emergency use
  MOSFET1_Switch(1);                                          // Turn off MOSFET1
  MOSFET2_Switch(0);                                          // Turn on MOSFET2
  VoltageError = 0x01;                                        // Switch to error state
  ChargingFlag = 0x00;
  EmergencyFlag = 0x01;
}

void StartModules(void){                                      // Start needed modules
  SleepTimer_Start();
  I2Cm_Start();
  MOSFET1_Start();
  MOSFET2_Start();
  ErrLED_Start();
  Timer8_Start();

  LED01_Start();
  LED02_Start();
  LED03_Start();
  LED04_Start();
  LED05_Start();
  LED06_Start();
  LED07_Start();
  LED08_Start();
  DATALED01_Start();
  DATALED02_Start();
  DATALED03_Start();
  DATALED04_Start();
  DATALED05_Start();
}

void Measure(void){
  I2Cm_fSendStart(0xF0,I2Cm_WRITE);                           // Write start 0xF0 (0xE0), WR
  I2Cm_fWrite(0x00);                                          // Write  0x00, command register
  I2Cm_fWrite(0x51);                                          // Write  0x51, measure cm
  I2Cm_SendStop();                                            // Write stop

  SleepTimer_SyncWait(32, SleepTimer_WAIT_RELOAD);
  I2Cm_fSendStart(0xF0,I2Cm_WRITE);                           // Write start 0xF0 (0xE0)
```

```
  I2Cm_fWrite(0x02);                                          // Write 0x02, range high byte

  I2Cm_fSendRepeatStart(0xF0,I2Cm_READ);                      // Write repstart 0xF0 (0xE0), R for(i
= 1; i < 4; i++){
    X[i] = I2Cm_bRead(I2Cm_ACKslave);                         // Get one byte master ack
  }
  X[4] = I2Cm_bRead(I2Cm_NAKslave);                           // Get one byte master nack
  I2Cm_SendStop();                                            // Write stop
  SleepTimer_SyncWait(32, SleepTimer_WAIT_RELOAD);
}

void DataHandler(void){

  if (X[2] <= height + 0x05){                                 // 100%
    DATALED01_Switch(0);                                      // Turn on every DATALED
    DATALED02_Switch(0);
    DATALED03_Switch(0);
    DATALED04_Switch(0);
    DATALED05_Switch(0);
    MeasurementError = 0;
  }

  else if (X[2] > height && X[2] < height * 0x02) {
                                                              // 90% (height > X > height * 2)
    DATALED01_Switch(1);                                      // Blink DATALED05
    DATALED02_Switch(0);                                      // Turn on others
    DATALED03_Switch(0);
    DATALED04_Switch(0);
    DATALED05_Switch(0);

    SleepTimer_SyncWait(16, SleepTimer_WAIT_RELOAD);

    DATALED01_Switch(0);
    DATALED02_Switch(0);
    DATALED03_Switch(0);
    DATALED04_Switch(0);
    DATALED05_Switch(0);
    MeasurementError = 0;
  }

  else if (X[2] >= height * 0x02 && X[2] < (height * 0x03)) {  // 80%
    DATALED01_Switch(1);                                      // Turn off DATALED01
    DATALED02_Switch(0);                                      // Turn on others
    DATALED03_Switch(0);
    DATALED04_Switch(0);
    DATALED05_Switch(0);
    MeasurementError = 0;
  }

  else if (X[2] >= (height * 0x03) && X[2] < (height * 0x04)) {  // 70%
    DATALED01_Switch(1);                                      // Blink DATALED02
    DATALED02_Switch(1);                                      // Turn off DATALED01
    DATALED03_Switch(0);                                      // Turn on others
    DATALED04_Switch(0);
    DATALED05_Switch(0);

    SleepTimer_SyncWait(16, SleepTimer_WAIT_RELOAD);

    DATALED01_Switch(1);
    DATALED02_Switch(0);
    DATALED03_Switch(0);
    DATALED04_Switch(0);
    DATALED05_Switch(0);
    MeasurementError = 0;
  }

  else if (X[2] >= (height * 0x04) && X[2] < (height * 0x05)) {  // 60%
    DATALED01_Switch(1);                                      // Turn off DATALED01 and DATALED02
    DATALED02_Switch(1);                                      // Turn on others
    DATALED03_Switch(0);
    DATALED04_Switch(0);
    DATALED05_Switch(0);
    MeasurementError = 0;
  }

  else if (X[2] >= (height * 0x05) && X[2] < (height * 0x06)) {  // 50%
    DATALED01_Switch(1);                                      // Blink DATALED03
    DATALED02_Switch(1);                                      // Turn off DATALED01 and DATALED02
    DATALED03_Switch(1);                                      // Turn on others
    DATALED04_Switch(0);
    DATALED05_Switch(0);
```

```
  SleepTimer_SyncWait(16, SleepTimer_WAIT_RELOAD);

  DATALED01_Switch(1);
  DATALED02_Switch(1);
  DATALED03_Switch(0);
  DATALED04_Switch(0);
  DATALED05_Switch(0);
  MeasurementError = 0;
}

else if (X[2] >= (height * 0x06) && X[2] < (height * 0x07)) {         // 40%
  DATALED01_Switch(1);                                                // Turn off DATALED01,02,03
  DATALED02_Switch(1);                                                // Turn on others
  DATALED03_Switch(1);
  DATALED04_Switch(0);
  DATALED05_Switch(0);
  MeasurementError = 0;
}

else if (X[2] >= (height * 0x07) && X[2] < (height * 0x08)) {         // 30%
  DATALED01_Switch(1);                                                // Blink DATALED04
  DATALED02_Switch(1);                                                // Turn off DATALED01,02,03
  DATALED03_Switch(1);                                                // Turn on others
  DATALED04_Switch(1);
  DATALED05_Switch(0);

  SleepTimer_SyncWait(16, SleepTimer_WAIT_RELOAD);

  DATALED01_Switch(1);
  DATALED02_Switch(1);
  DATALED03_Switch(1);
  DATALED04_Switch(0);
  DATALED05_Switch(0);
  MeasurementError = 0;
}

else if (X[2] >= (height * 0x08) && X[2] < (height * 0x09)) {         // 20%
  DATALED01_Switch(1);                                                // Turn on DATALED05
  DATALED02_Switch(1);                                                // Turn off others
  DATALED03_Switch(1);
  DATALED04_Switch(0);
  DATALED05_Switch(0);
  MeasurementError = 0;
}

else if (X[2] >= (height * 0x09) && X[2] < (height * 0x0A - 0x05)) {  // 10%
  DATALED01_Switch(1);                                                // Blink DATALED05
  DATALED02_Switch(1);                                                // Turn off others
  DATALED03_Switch(1);
  DATALED04_Switch(1);
  DATALED05_Switch(1);

  SleepTimer_SyncWait(16, SleepTimer_WAIT_RELOAD);

  DATALED01_Switch(1);
  DATALED02_Switch(1);
  DATALED03_Switch(1);
  DATALED04_Switch(1);
  DATALED05_Switch(0);
  MeasurementError = 0;
}

else if (X[2] >= (height * 0x0A - 0x05) && X[2] < (height * 0x0B) ) { // 0%
  DATALED01_Switch(1);                                                // Turn off every DATALED
  DATALED02_Switch(1);                                                // Blink errorled
  DATALED03_Switch(1);
  DATALED04_Switch(1);
  DATALED05_Switch(1);
  MeasurementError = 0;
  ErrLED_Switch(1);
  SleepTimer_SyncWait(16, SleepTimer_WAIT_RELOAD);

  ErrLED_Switch(0);

}else {                                                               // Error occured!
  MeasurementError++;                                                 // Increase Error counter
}

if (X[2] >= (height * 0x0B)){
  MeasurementError++;                                                 // Increase Error counter
```

```
  }
}

void ErrorHandler(void) {                                   // Handle Errors
  if (MeasurementError >= 0x04 || VoltageError >= 0x01) {
    ErrLED_Switch(0);                                       // Errors have occured, ErrLED on
  } else{
    ErrLED_Switch(1);                                       // No errors occured, ErrLED off
  }
}

void btnISR(void) {
  if(SelectFlag == 0x00){
    if((UpBtn_Data_ADDR & UpBtn_MASK)){                     // Up button function
      height++;                                             // Increase by 10 cm (0x01)
      if(height >= 0x14){
        height = 0x14;
      }
    }
    else if((DownBtn_Data_ADDR & DownBtn_MASK)){            // Down button function
      height--;                                             // Decrease by 10 cm (0x01)

      if(height <= 0x06){
        height = 0x06;
      }
    }

    else if((SelectBtn_Data_ADDR & SelectBtn_MASK)){        // Select button function
      SelectFlag = 0x01;
    }
  }
}

void timeISR(void){                                         // Timer interrupt
  ms++;                                                     // Milliseconfs increasement
    if(ms > 0x09){                                          // Tenths full, reset
      ms = 0x00;
      ss++;                                                 // Seconds increasement
      if(ss > 0x3B){                                        // Seconds full, reset
        ss = 0x00;
        mm++;                                               // Minutes increasement
        ContinueFlag = 0x01;                                // Minutes full, --> Measuring
        if(mm > 0x3B){                                      // Minutes full, reset
          mm = 0x00;
          hh++;                                             // Hours increasement
          if(hh > 0x17){                                    // Hours full, reset
            hh = 0x00;
        }
      }
    }
  }
}
```