TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka, tietokonetekniikka
Timo Lähteenmäki

Tutkintotyö

Timo Lähteenmäki

MONIPUOLINEN SÄÄASEMA

Työn valvoja: Yliopettaja Kai Poutanen
Tampere 2006

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Tietokonetekniikka
Lähteenmäki, Timo
Monipuolinen sääasema
Tutkintotyö 31 sivua + 10 liitesivua
Työn valvoja Yliopettaja Kai Poutanen
Lokakuu 2006
Hakusanat sääasema, mikrokontrolleri, anturi, Bluetooth

TIIVISTELMÄ

Erilaiset anturijärjestelmät ja pienikokoiset sulautetut järjestelmät monipuolisine tietoliikenneyhteyksineen kuuluvat nykyisin lähes jokaisen ihmisen arkeen. Usein nämä järjestelmät on suunniteltu niin varmoiksi ja yksinkertaisiksi, ettei niiden toimintaa tule edes ajateltua.

Anturijärjestelmän pääosat ovat mikrokontrolleripohjainen kytkentäalusta ohjelmistoineen ja useat erilaiset anturit. Antureista saadaan tärkeätä informaatiota monelle eri tekniikan alueelle tehdastekniikasta säätieteeseen. Valmistajilta on saatavissa varsin vähän tietoa ja asiasta kiinnostuneet harrastajat joutuvat usein suunnittelemaan alkeellisetkin kytkennät itse.

Tässä tutkintotyössä keskitytään AVR-mikrokontrolleripohjaisen mittausalustan ja sääasemassa käytettävien anturikytkentöjen suunnitteluun, toimintaan ja pääosin C-kieliseen ohjelmointiin. Sääasemalla antureilla saadaan tietoa esimerkiksi ilmanpaineen, lämpötilan ja ilmankosteuden vaihteluista. Näiden tietojen avulla voidaan seurata sääilmiöiden kehittymistä ja luoda pitkäaikaisiakin ennusteita.

Työssä edetään vaiheittain sääaseman anturijärjestelmän määrittelystä suunnitteluun ja toteutukseen sekä järjestelmätestaukseen. Suunnittelu muodostaa suurimman osan työstä ja sisältää tietoa erilaisista antureista, mikrokontrollereista ja elektroniikkasuunnittelun perusteista. Työssä pyritään antamaan hyvät perustelut tehdyille ohjelmisto- ja komponenttivalinnoille.

Osa työssä käytetyistä ohjelmistoista on saatavissa ilmaiseksi suoraan valmistajilta. Ohjelmiston kehitykseen käytetty ohjelmointiympäristö toimii sekä Unix/Linux- että Windows-käyttöjärjestelmissä. Osa tutkintotyöstä julkaistaan vapaasti käytettäväksi Internetissä ja järjestelmää kehitetään myös tulevaisuudessa.

TAMPERE POLYTECHNIC
Computer Systems Engineering
Computer Engineering
Lähteenmäki, Timo A Versatile Weather Station
Engineering Thesis 31 pages, 10 appendices
Thesis Supervisor Senior Lecturer Kai Poutanen
October 2006
Keywords weather station, microcontroller, sensor, Bluetooth

ABSTRACT

Different kinds of sensors and integrated- systems play an important part in many people's daily lives. This thesis explains the designing, functioning, and programming of a USB, and a Bluetooth-connection equipped AVR-microcontroller based weather station. In a weather station, the sensor information contains for example air pressure, air temperature, and air humidity fluctuations. This work begins from defining the sensor system, continues to the designing and building of the actual device, and ends with the testing of the whole system.
The component choices and software model decisions are explained by a good argumentation and most of the software used in this thesis is available for free personal use on Unix/Linux- and Windows-based operating systems. Part of this thesis is to be published on the Internet for free use, and the designed system is going to be under continuous development.

ALKUSANAT

Tämän tutkintotyön innoittajana on toiminut insinöörimäinen halu tutustua tarkasti mikrokontrolleripohjaisen elektroniikkalaitteen suunnitteluun ja ohjelmointiin sekä nykyisin yleiseen USB- ja Bluetooth-tiedonsiirtoon.

Ohjelmiston mahdollisen kaupallisen hyödyntämisen vuoksi tässä työssä ei esitetä laitteen täydellistä ohjelmistoa (yhteensä yli 1200 koodiriviä).

Haluan kiittää työni ohjaajaa Kai Poutasta ja muita opettajiani lukuisista mielenkiintoisista hetkistä tietokonetekniikan parissa opiskeluaikanani Tampereen ammattikorkeakoulussa.

Tampereella 30. lokakuuta 2006

Timo Lähteenmäki

SISÄLLYSLUETTELO

KÄYTETYT TERMIT JA LYHENTEET

A/D-muunnin	Analog-to-Digital-muunnin, muuntaa analogisen jännitteen digitaaliseen muotoon
ASCII	American Standard Code for Information Interchange, yleinen tietokoneiden merkistö, sisältää myös ohjauskoodeja
AVR	Atmelin 8-bittinen mikrokontrolleriperhe
ATmega	Atmelin AVR-tuoteperheen mikrokontrolleri
Bluetooth	Lyhyen kantaman langaton yhteys
C-kieli	Korkean tason ohjelmointikieli
CMOS	Complementary Metal-Oxide Semiconductor, pienen tehonkulutuksen omaava puolijohdetekniikka
EEPROM	Electrically Erasable Programmable Read-Only Memory, sähköisesti tyhjennettävä lukumuisti
Flash	Muisteissa käytettävä edullinen puolijohdetekniikka
Makefile	Käännöksen aputiedosto, joka sisältää tarvittavat tiedot kääntäjälle ja linkkerille
MIPS	Million Instructions Per Second, miljoonaa käskyä sekunnissa, eräs laskentatehon vertailuluku
I/O	Input/Output, mikrokontrollerin tulo- ja lähtöportit
ISP	In-System Programmer, mikrokontrollerin ohjelmointiväylä
RISC	Reduced Instruction Set Computer, rajoitetun käskykannan mikroprosessori
RS-232	Yleisesti käytetty sarjaliikenneväylästandardi
SRAM	Static Random Access Memory, staattinen ja nopea käyttömuisti
TTL	Transistor-Transistor Logic, +5 voltin käyttöjännitteellä toimiva mikropiiriperhe
U(S)ART	Universal (Synchronous) Asynchronous Receiver/Transmitter, synkroninen/asynkroninen sarjaliikennepiiri
USB	Universal Serial Bus, yleistynyt sarjaliikenneväylästandardi

1 JOHDANTO

Nykyisin yhä suurempi osa laitteistojen toimintaan vaadittavista tiedoista ja havainnoista saadaan erilaisten antureiden välityksellä. Anturit ovat korvanneet useissa prosesseissa tarvittavat ihmisen aisteihin perustuvat mittaustiedot miltei kokonaan. Anturilta saatava tieto on usein luotettavampaa, tarkempaa ja huomattavasti nopeampaa sekä edullisempaa kuin ihmiseltä saatavat havainnot. Anturimäärän lisääntyessä on siirrytty myös langattomaan tiedonsiirtoon suuren signaalijohtimien määrän vähentämiseksi.

Innostus tähän tutkintotyöhön lähti ajatuksesta suunnitella valinnaisille antureille sopiva, monipuolinen mikrokontrolleripohjainen mittausalusta, joka sisältää mahdollisesti myös langattoman tiedonsiirron. Pian käytettäviksi antureiksi valikoituivat lämpötilaa, ilmankosteutta ja ilmanpainetta mittaavat anturit, ja langattomaksi yhteystekniikaksi rajautui Bluetooth.

Tutkintotyössä suunnitellaan ja toteutetaan laite, joka pystyy halutuin väliajoin mittaamaan ja arkistoimaan nämä anturitiedot sekä siirtämään tiedot tarvittaessa tietokoneeseen USB-väylää tai Bluetooth-yhteyttä pitkin. Näitä tietoja voidaan käyttää myöhemmin esimerkiksi paikallisen sään kehityksen havainnointiin ja tiedot ovat siirrettävissä lähes reaaliaikaisesti Internet-sivuille muiden käyttäjien nähtäväksi.

Toisessa luvussa käsitellään laitteen elektroniikan ja ohjelmiston toiminnallinen määrittely. Koko järjestelmän suunnitteluprosessi käydään läpi kolmannessa luvussa ja neljäs luku sisältää järjestelmän testauksen.

2 TOIMINNALLINEN MÄÄRITTELY

2.1 Toimintaperiaate

Laitteen tulee kyetä mittaamaan ja arkistoimaan erilaisilta antureilta saatavaa tietoa ja lähettämään se käyttäjän tietokoneelle Bluetooth-yhteyden sekä USB-väylän yli.

2.2 Elektroniikka

Laitteen tulee perustua Atmelin valmistamaan 8-bittiseen AVR-mikrokontrolleriin. Laitteeseen valittavan mikrokontrollerin tulee täyttää toiminnallisessa määrittelyssä asetetut vaatimukset laskentatehon, muistiominaisuuksien ja ulkoisten liitäntöjen suhteen.

Järjestelmän ja käyttäjän päätelaitteen välinen yhteys toteutetaan USB-liitännällä ja langattomalla Bluetooth-yhteydellä ja hyödynnetään sen tarjoamaa sarjaporttiprofiilia (SPP). Järjestelmän tulee lähettää antureiden mittaustiedot käyttäjän päätelaitteelle vakiomuodossa, ASCII-koodattuna.

Laitteen elektroniikan toteutuksessa käytetään hyväksi valmistajien antamia esimerkkikytkentöjä ja Cadencen Orcad-tuoteperhettä. Suunnittelussa otetaan huomioon myös elektroniikkatuotteen suunnitteluun liittyvät suositukset ja määritellään lopulliseen tuotteeseen tulevat komponentit.

USB-yhteyden elektroniikka toteutetaan FTDI:n valmistamalla FT232BL-mikropiirillä ja tarvittavalla oheiskytkennällä. Bluetooth-yhteyden vaatima elektroniikka toteutetaan käyttäen Spark Fun Electronics:n BlueSMiRF Bluetooth -moduulia, joka toimii sarjaporttiprofiililla. Ilmanpaineanturi ja yhdistetty ilmankosteus- ja lämpötila-anturi liitetään kiinteäksi osaksi järjestelmää erillisiin kytkentäpisteisiin ja mahdollisille ylimääräisille antureille suunnitellaan omat liitäntänsä.

2.3 Ohjelmisto

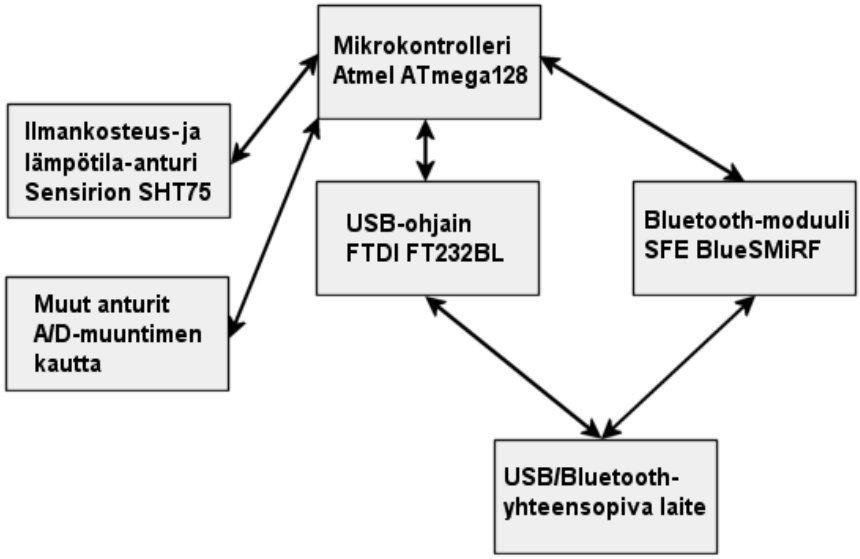
Ohjelmiston tulee toteuttaa toimintaperiaatteessa määritellyt toiminnat yhdessä aiemmin määritellyn elektroniikan kanssa. Ohjelmisto saa ohjaussyötteensä käyttäjältä ja tulostaa mittauksensa ASCII-koodattuna UART-pohjaisena sarjaväylää pitkin. Datan ylimääräisen analysoinnin ja jatkokäsittelyn tekee käyttäjän oma laitteisto.

Ohjelmointikielenä tulee käyttää C-kieltä ja tarvittaessa Assembler-kielisiä käskyjä. Ohjelmiston tulee olla selkeästi luettavaa ja modulaarista ja sen tulee olla käännettävissä ilmaisella WinAVR-ohjelmistolla. Järjestelmän ohjelmiston toiminta kuvataan kappaleessa 3.2 ja ohjelmiston toimintakaaviossa (liite 1).

3 SUUNNITTELU

3.1 Laitteisto

Laitteiston yksinkertaistettu lohkokaavio on esitettynä kuvassa 1.



Kuva 1 Laitteiston lohkokaavio

3.1.1 Mikrokontrollerin valinta

Atmelin AVR-mikrokontrollerit ovat CMOS-tekniikalla valmistettuja, 8-bittisellä RISC-ytimellä varustettuja mikrokontrollereita. AVR-mikrokontrollerit pystyvät suorittamaan yhden konekielisen käskyn kellojakson aikana ja ovat siten erittäin suorituskykyisiä useisiin muihin suorittimiin verrattuna. AVR-mikrokontrollerit sisältävät mikroprosessorin ydinosan lisäksi ohjelmoitavan SRAM-käyttömuistin, pysyvää flash-muistia, pysyvää EEPROM-muistia, erityyppisiä I/O-portteja, A/D-muuntimia ja muita lisäosia mallimerkinnän mukaan /4/.

Mikrokontrollerin valintakriteereinä ovat riittävä laskentateho ilman erillisiä komponentteja, ohjelmoitavan flash-muistin määrä, pieni tehonkulutus ja ulkoisten liitäntöjen määrä.

Mikrokontrollerin laskentateho
Laskentatehon minimivaatimuksen perusteena on tarvittava suurin tiedonsiirtonopeus mikrokontrollerilta vastaanottavalle laitteelle (>19200 bps) kun käytetään mikrokontrolleriin sisältyvää, ohjelmoitavaa UART-yksikköä. Varsinaisia mittauksia ei kannata tehdä kovin usein, koska käytettävillä antureilla on varsin suuri vasteaika muutoksille (SHT75:n maksimivasteaika 30s /1/).

Vaatimuksen perusteella mikrokontrollerin kellotaajuuden minimivaatimukseksi arvioidaan 8 MHz, jolloin päästään turvallisesti 38400 bps:n tiedonsiirtonopeuteen /2/. AVR-mikrokontrolleri pystyy suorittamaan yhden konekielisen käskyn jokaisella kellojaksolla, jolloin minimivaatimuksesta seuraa muihin laskutoimituksiin hyvin riittävä 8 MIPS:n laskentateho.

Mikrokontrollerin muistin määrä
Ohjelmiston vaatimus mikrokontrollerin vapaasti ohjelmoitavan muistin määrästä on vähintään neljä kilotavua, jotta koko ohjelmisto olisi mahdollista sijoittaa kontrolleriin. Atmelin AVR-mikrokontrollerit sisältävät mallin mukaan 32-8192 tavua SRAM-muistia, 64-4096 tavua EEPROM-muistia ja 1-256 kilotavua flash-muistia.

Tallennettaville anturitiedoille ja mahdollisille ohjelmistolaajennuksille tulee varata mahdollisimman paljon vapaata muistia, jolloin muistin kokonaistarpeeksi arvioidaan yli 32 kilotavua. 32 kilotavun muistivaatimus rajoittaa valinnan ATmega-sarjan mikrokontrollereihin, joiden muistikapasiteetti alkaa 32 kilotavusta /4/.

Mikrokontrollerin liitäntäportit

Mikrokontrollerin ja päätelaitteen välinen yhteys on toteutettu mikrokontrolleriin sisältyvällä ohjelmoitavalla UART-yksiköllä, jonka ulkoisia kytkentöjä varten tarvitaan kolme sitä varten varattua I/O-porttia (RxD, TxD ja SCK). USB-ohjain käyttää näitä portteja ja välittää tiedon eteenpäin vastaanottavalle laitteelle. Myös Bluetooth-moduuli välittää saman sarjamuotoisen tiedon radiolinkillä jolloin mikrokontrollerin ja Bluetooth-moduulin välille ei tarvita erillistä elektroniikkaa (USB ja Bluetooth eivät ole käytössä yhtäaikaisesti).

Mikrokontrollerin ja yhdistetyn lämpötila- ja ilmankosteusanturin välistä liityntää varten mikrokontrolleri tarvitsee kolme TTL-tasoista logiikkaporttia ja ilmanpaineanturia varten yhden kappaleen A/D-muunnintuloja sekä yhden TTL-tasoisen logiikkaportin. Lisäantureita varten tarvitaan lisäksi kolme kappaletta A/D-muunnintuloja ja kymmenen TTL-tasoista logiikkaporttia.

Riittävän mittaustarkkuuden saavuttamiseksi A/D-muuntimen tarkkuuden pitää olla vähintään kymmenen bittiä. Useissa edullisemmissa ja pienemmissä mikrokontrollereissa A/D-muunnin on toteutettu vertailukytkennällä, joka ei ole tähän sovellukseen riittävän tarkka. ATmega-sarjan mikrokontrollerit sisältävät vaihtelevan määrän peräkkäisapproksimaatioperiaatteella toimivia kahdeksan tai kymmenen bitin A/D-muuntimia /4/.

Mikrokontrollerin valinta

Vaatimuksien perusteella rajoittavimmat kriteerit ovat riittävä, ilman oheiskomponentteja saavutettava sisäinen kellotaajuus, vapaasti ohjelmoitavan flash-muistin määrä ja riittävän tarkan A/D-muuntimen saatavuus.

Näiden kriteerien perusteella valinta kohdistuu Atmelin ATmega-sarjan mikrokontrollereihin. Vaatimukset täyttävistä mikrokontrollereista pienin ja edullisin on ATmega-sarjan keskiluokkainen mikrokontrolleri, ATmega32. ATmega32 sisältää muun muassa ohjelmoitavan sarjaliikenneväylän, 32 kilotavua vapaasti ohjelmoitavaa flash-muistia ja 10-bittisen peräkkäisapproksimaation periaatteella toimivan muuntimen.

Tässä tutkintotyössä käytettäväksi mikrokontrolleriksi valittiin kuitenkin ATmega128 monipuolisuutensa, jatkokäyttömahdollisuuksiensa ja helpon saatavuutensa vuoksi.

3.1.2 Kosteusanturin ja lämpötila-anturin valinta

Kosteusanturin valinnassa tärkeimpiä kriteereitä olivat tarkkuus, hinta, käyttöjännite ja saatavuus. Sopivaa anturia etsittäessä havaittiin sarjaväyläinen ratkaisu perinteistä A/D-muuntimeen perustuvaa ratkaisua tarkemmaksi ja varmemmaksi.

Sopivin anturi löytyi sveitsiläiseltä Sensirion AG:ltä. Sen valmistama SHT75-anturi on tarkka, yhdistetty ilmankosteus- ja lämpötila-anturi, ja siinä on digitaalinen 2-johdin ulostulo /1/.

3.1.3 Ilmanpaineanturin valinta

Ilmanpaineanturia valitessa tärkeimmät kriteerit olivat sopiva käyttöalue, käyttöjännite, edullisuus ja hyvä saatavuus. Teollisuuskäyttöön tarkoitetut anturit on usein suunniteltu käyttöalueeltaan maanpinnan ilmanpaineesta poikkeaviksi, eivätkä ne siten sovellu sääasemakäyttöön. Tässä sovelluksessa tarvittava käyttöalue on maapallon ilmanpainealue, noin 800-1100 hehtopascalia (hPa) merenpinnan tasolla /7/.

Kun tiedetään anturin sijainnin korkeus merenpinnasta, voidaan anturista saadut lukemat sovittaa merenpinnan tasoa vastaavaksi lisäämällä 1 hPa jokaista 8 metriä kohden. Tällöin lukemat ovat vertailukelpoisia sijainnin korkeudesta riippumatta.

Freescale Semiconductor valmistaa lukuisia sopivia anturimalleja, joista tähän työhön sopivimmaksi havaittiin MPXAZ6115A-anturi, jossa on luontainen mittauselektroniikka ja joka kestää hyvin kosteutta ja erilaisia kemikaaleja. Anturin käyttöalue on 150-1150 hehtopascalia, joten se soveltuu hyvin sääasemakäyttöön /3/. MPXAZ6115A-anturia käytetäänkin useimmiten autojen moottorinohjauselektroniikassa ilmanpaineen valvontaan. Tässä yhteydessä anturi liitetään järjestelmän A/D-muuntimen sisääntuloon.

3.1.4 USB-ohjaimen valinta /6/

Universal Serial Bus (USB) on 1995 julkaistu sarjaväylästandardi, joka suunniteltiin alun perin lähinnä tietokonekäyttöön korvaamaan vanhaa RS232-pohjaista standardia, mutta joka on sittemmin levinnyt pääasialliseksi tiedonsiirtotavaksi myös erilaisiin elektronisiin laitteisiin kuten digitaalikameroihin, MP3-soittimiin, musiikki-instrumentteihin ja jopa televisioihin. USB-yhteydellä on mahdollista päästä nykyisin jopa 480 Mbitin siirtonopeuteen käyttämällä Hi-Speed-yhteensopivaa USB 2.0-laitteistoa.

USB toimii kolmessa seuraavassa nopeusluokassa:
- Low Speed, 1.5 Mbit/s, käytetään yleisimmin erilaisissa tietokoneen oheislaitteissa, kuten näppäimistöissä, hiirissä ja peliohjaimissa.
- Full Speed, 12 Mbit/s, nopein yhteystapa vanhemmilla USB-versioilla, liitetyt laitteet jakavat nopeuden keskenään.
- Hi-Speed, 480 Mbit/s, nopein nykyisen USB 2.0:n siirtonopeus.

Tässä työssä käytettävään tiedonsiirtoon riittää jo pelkän ensimmäisen nopeusluokan 1.5 Mbit/s:n siirtonopeus, mutta useimmat ohjaimet toimivat jopa USB 2.0:n Hi-Speed-nopeudella. USB-ohjainta valittaessa tärkeimmiksi kriteereiksi muodostuivat ohjaimen käyttöjännite, suora toimivuus AVR:n UART-lähdön kanssa, hyvä ajuri eri tietokoneiden käyttöjärjestelmille sekä edullisuus ja hyvä saatavuus.

Nämä kriteerit parhaiten täyttää FTDI:n (Future Technology Devices International) valmistama FT232BL-ohjain, joka toimii 5 voltin käyttöjännitteellä (voidaan ottaa suoraan isäntälaitteesta) ja toimii USB 1.1 ja USB 2.0-laitteistoissa Windows-, Mac OS-, ja Linux-ympäristöissä.

FT232BL on lyijytön EU:n RoHS-direktiivit täyttävä versio aiemmin valmistetusta FT232BM:stä ja on siten aiempaa ympäristöystävällisempi. Liittämällä ohjaimeen erillisen EEPROM-muistin voidaan laitteelle kirjoittaa omat tuotekuvaukset ja tunnistetiedot.

3.1.5 Bluetooth-moduulin valinta /8/, /9/

Bluetooth on vuonna 1999 määritelty langaton verkkoyhteystapa, joka on tarkoitettu lähinnä pienten kannettavien laitteiden väliseen viestintään. Bluetooth käyttää kansainvälisesti vapaata 2.4 – 2.485 GHz:n taajuusaluetta ja hyödyntää adaptiiviseen taajuushyppelyyn perustuvaa hajaspektritekniikkaa. Bluetooth-yhteyden välityksellä voidaan välittää sekä dataa että ääntä. Yhteyden maksimisiirtonopeus nykyisellä tekniikalla on 3 Mbps (versio 2.0).

Bluetooth-yhteys on tarkoitettu lyhyen kantaman (0-100 m) yhteydeksi ja sen toteuttavat moduulit ja päätelaitteet on jaettu kolmeen eri luokkaan lähetystehon ja toimintaetäisyyksien suhteen. Ensimmäisen luokan Bluetooth-laitteiden lähetystehon maksimiarvo on +16 dBm ja määritelty kantomatka 100 metriä vapaassa tilassa. Toisen luokan Bluetooth-laitteiden lähetystehon maksimiarvo on +4 dBm ja määritelty kantomatka 30 metriä vapaassa tilassa. Kolmannessa luokassa lähetystehon maksimi on vastaavasti 0 dBm ja kantavuus 1 metri vapaassa tilassa.

Bluetooth-laitteet muodostavat pikoverkon, jossa yksi laitteista toimii isäntälaitteena ja muut orjalaitteina. Isäntälaite on tyypillisesti suurempi laite, kuten käyttäjän mobiilipäätelaite tai tietokone.

Bluetooth-komponentteja on saatavilla erillisinä radio-osan ja ohjausprosessorin sisältävinä mikropiireinä ja käyttövalmiina moduuleina. Käyttövalmiit moduulit sisältävät radio-osan ja ohjausprosessorin lisäksi muun muassa liitäntäkomponentit isäntälaitteeseen, muistipiirit laiteohjelmistoa, lähetys- ja vastaanottopuskuria sekä parametrien tallentamista varten. Moduuleissa on useimmiten mukana myös tarvittava ohjelmisto.

Bluetooth-moduulien yleisin liitäntätapa isäntälaitteeseen on sarjaliikenneväylä. Bluetooth toimii useissa erilaisissa toimintatavan määrittelevissä profiileissa. Tietokonekäytössä yleisimmin käytetään sarjaporttiprofiilia (SPP), jonka avulla Bluetooth-yhteydellä voidaan korvata perinteiset langalliset sarjaliikenneväylät. SPP:lla toteutettu yhteys on muun järjestelmän kannalta täysin läpinäkyvä ja se toimii kuten perinteinen langallinen sarjaliikenneyhteys.

Tärkeimmät Bluetooth-moduulin valintaan vaikuttavat kriteerit ovat helppokäyttöisyys, liitäntätapa ja moduulin käyttämät profiilit. Saatavilla on useita edellä mainitut vaatimukset täyttäviä Bluetooth-moduuleja, joten valinnan ratkaisevat moduulin helppokäyttöisyys ja hinta.

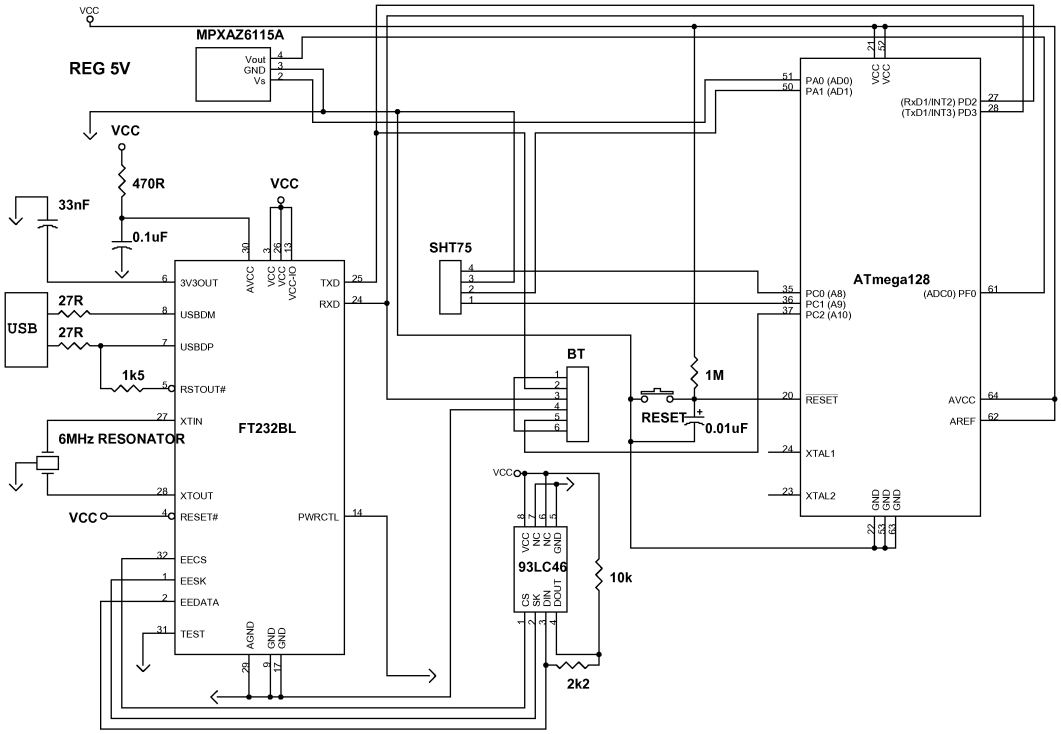
Käyttötarkoitukseen sopivimmaksi valitaan Spark Fun Electronicsin BlueSMiRF Bluetooth -moduuli, joka täyttää kaikki asetetut vaatimukset ja sisältää selkeän dokumentaation laitteen ominaisuuksista. BlueSMiRF on ensimmäisen luokan Bluetooth-moduuli, jolle luvataan moduulin piirilevylle rakennetulla antennilla jopa 100 metrin kantomatka. BlueSMiRF liitetään mikrokontrollerin ulkoiseen sarjaväylään ja se sisältää sarjaporttiprofiilin toteuttavan ohjelmiston.

3.1.6 Mikrokontrollerin kytkennät

Mikrokontrolleri tarvitsee perustoimintojansa varten käyttöjännitteen ja reset-signaalin kytkemisen. Tämän lisäksi mikrokontrollerin portit pitää kytkeä käytettäville antureille ja Bluetooth-moduulille.

Kaikkien antureiden ja USB-ohjaimen ulkoisen resonaattorin piirilevylle kytkemisessä tulee pyrkiä mahdollisimman lyhyisiin ja suoriin johdinvetoihin ulkoisten häiriöiden vähentämiseksi.

Järjestelmän yksinkertaistettu peruskytkentä on esitettynä kuvassa 2.



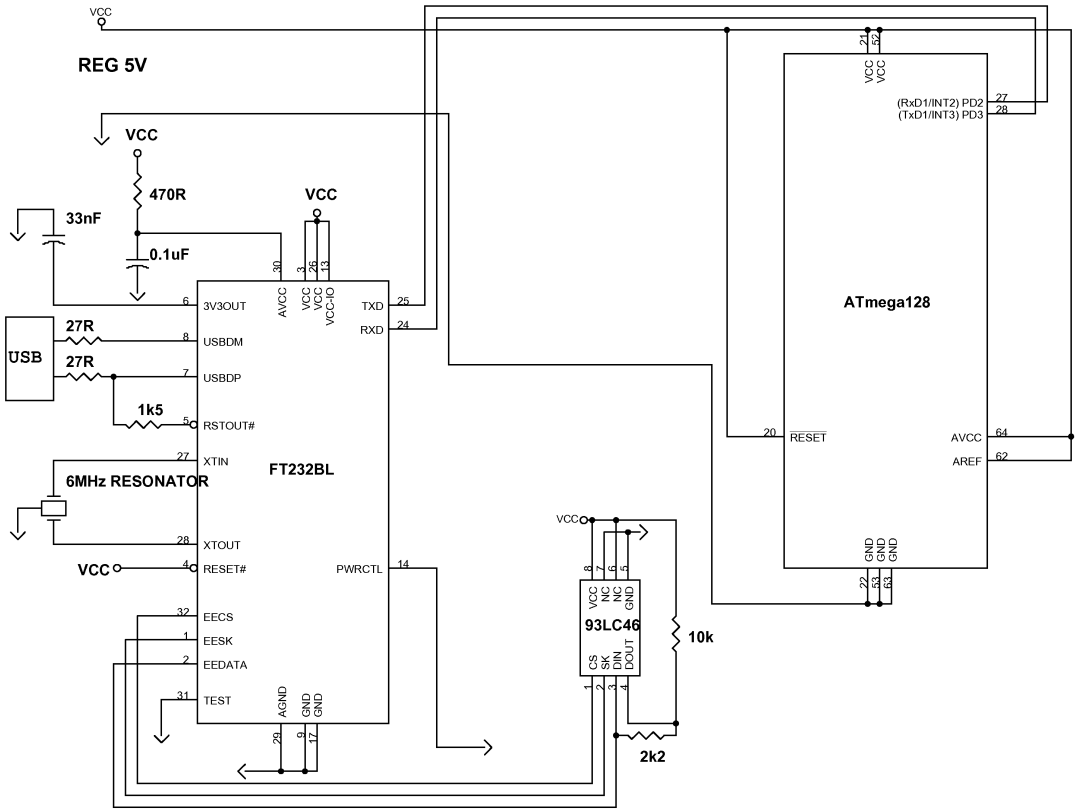
Kuva 2 Järjestelmän peruskytkentä

Kuvasta puuttuvat virtalähteen vaatimat komponentit ja mikrokontrollerin ISP-ohjelmointiliitin sekä valinnaiset anturilähdöt. Järjestelmän täydellinen kytkentäkuva on liitteessä 8.

Huomattavaa kytkennässä on, että mikrokontrollerista on jätetty ulkoisen kiteen XTAL-nastat kokonaan kytkemättä. ATmega128-mikrokontrollerissa on sisäinen, valinnainen kellopiiri, jonka avulla järjestelmäkellon taajuus voidaan asettaa toimimaan välillä 1-8 MHz.

3.1.7 USB-ohjaimen liittäminen mikrokontrolleriin

USB-ohjain liitetään mikrokontrollerin sarjaliikenneväylään, ohjaimen muodostavat FTDI:n FT232BL-mikropiiri ja sen tarvitsemat oheiskomponentit kuvan 3 mukaisesti. Kuvassa mukana myös ohjaimelle soveltuva EEPROM-muistipiiri 93LC46, johon voidaan ohjelmoida laitteen omat tunnistetiedot.



Kuva 3 USB-ohjaimen kytkentä

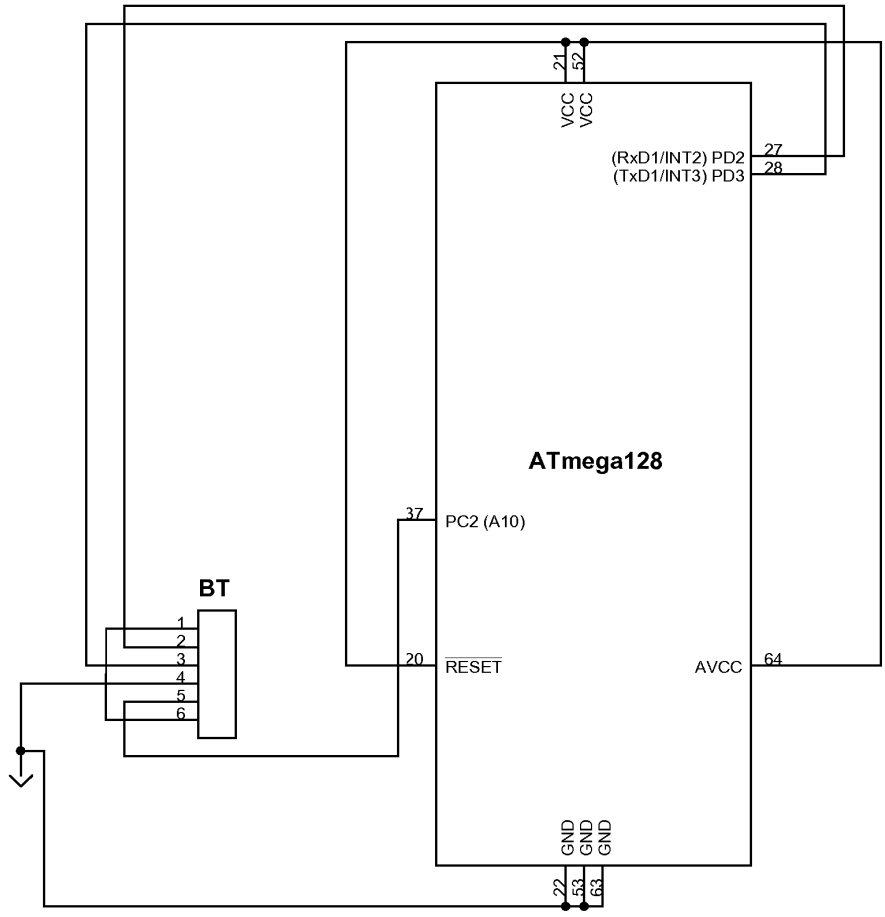
Kuvan VCC-nastat voidaan kytkeä joko järjestelmän yleiseen +5V:n käyttöjännitteeseen tai USB-liitännästä saatavaan noin +5V:n jännitteeseen.

3.1.8 Bluetooth-moduulin liittäminen mikrokontrolleriin

Bluetooth-moduuli liitetään mikrokontrollerin sarjaliikenneväylään. Moduulin UART-kontrolleri mahdollistaa tiedonsiirtonopeuden valinnan väliltä 2400 – 115200 bps. Oletuksena sarjaliikenneväylän nopeus on 9600 bittiä sekunnissa, mikä olisi riittävä tähän sovellukseen, jos laitteelta lähetettäisiin vain yksittäisiä tietoja.

Tässä yhteydessä tarkoituksena on kuitenkin lähettää suuria määriä mittaustietoja pitkiltäkin aikaväleiltä, jolloin nopeutta on hyvä lisätä tiedonsiirron nopeuttamiseksi ja virrankäytön vähentämiseksi. Siirtonopeuden voi vaihtaa lähettämällä moduulille AT-komentoja terminaaliohjelmalla. Lopulliseksi nopeudeksi valittiin 38400 muun oheislaitteiston asettamien rajoitusten vuoksi.

Moduuli liittyy mikrokontrolleriin TTL-tasoisena, joten erillisiä oheiskomponentteja ei tarvita. Bluetooth-moduulin liitäntä mikrokontrolleriin on esitetty kuvassa 4.



Kuva 4 Bluetooth-moduulin kytkentä mikrokontrolleriin

Kuten kuvasta 4 ilmenee, Bluetooth-moduuli on kytketty mikrokontrollerin ensimmäiseen sarjaliikenneväylään, UART0:aan.

Moduulin asetusten muokkausta varten moduuli vastaanottaa mikrokontrollerilta tai Bluetooth-isännältään Hayes-yhteensopivia AT-komentoja. AT-komennot ovat perinteinen, nykyisinkin käytettävä tapa ohjata erilaisia sarjaväylään liitettyjä laitteita, kuten puhelinmodeemia. AT-komennot alkavat aina "AT"-etuliitteellä ja päättyvät rivinvaihtoon. Yleensä vastaanottava laite vastaa syötettyyn komentoon joko "OK"- tai "ERROR"- viestillä. Joissakin erikoistapauksissa laite voi kuitenkin jättää vastaamatta komentoon, tai komennosta voi seurata jokin erikseen määrätty tulostus.

AT-komentojen vastaanottotilaan pääsemiseksi tulee Bluetooth-isännältä lähettää laitteelle ensin kolme (3) plus (+) merkkiä ja rivinvaihto. Tämän jälkeen lähetetyt merkit menevät suoraan moduulille eikä niitä välitetä eteenpäin mikrokontrolleriin. Tästä tilasta poistumiseksi tulee moduulille lähettää komento ATMD ja rivinvaihto, jonka jälkeen syötetyt merkit ohjataan jälleen mikrokontrollerin UART-liityntään. Mikrokontrollerista käsin operoitaessa ei edellisen kaltaista menettelyä tarvita.

3.1.9 Ilmanpaineanturin liittäminen mikrokontrolleriin

Ilmanpaineanturi liitetään mikrokontrollerin A/D-muunnoskanavaan ADC0(PF0). Lisäksi anturi tarvitsee +5 voltin käyttöjännitesyötön, joka toteutetaan ohjelmallisesti kytkettäväksi mikrokontrollerin I/O-nastasta PA0(AD0).
Anturin ja mikrokontrollerin välinen kytkentä on kuvattuna aiemmin kuvassa 2.

3.1.10 Ilmankosteus- ja lämpötila-anturin liittäminen mikrokontrolleriin

Yhdistetty ilmankosteus- ja lämpötila-anturi liitetään kahdella erillisellä I/O-väylällä mikrokontrolleriin. Näiden lisäksi anturi tarvitsee normaalin +5 voltin käyttöjännitesyötön ja käyttöjännitemaan. Tässä kytkennässä anturin jännitesyöttö on toteutettu mikrokontrollerin I/O-nastasta PA1(AD1).
Toteutuksesta johtuen anturin virransyöttö voidaan katkaista ohjelmallisesti aina niin halutessa. Anturin ja mikrokontrollerin välinen kytkentä on kuvattuna aiemmin kuvassa 2.

3.2 Ohjelmisto

Laitteen ohjelmisto on kirjoitettu ISO C99 -standardin mukaisella C-ohjelmointikielellä, WinAVR-ohjelmointiympäristön mukana tulevalle GNU GCC -kääntäjälle. Ohjelmisto sisältää myös C-koodiin sisällytettyjä Assembler-kielisiä laitteistoläheisiä käskyjä. Ohjelmoinnissa on pyritty mahdollisimman modulaariseen ja selkeään rakenteeseen, minkä ansiosta mahdolliset laajennukset ja muutokset ovat helppoja toteuttaa tulevaisuudessa. Koodin muotoilussa on noudatettu C-kielen yleissuositusten mukaisia merkintätapoja.

Suurin osa ohjelmoinnissa käytetyistä funktioista ja vakioista löytyy suoraan WinAVR:n mukana tulleesta avr-libc-kirjastosta. Tarvittavat mikrokontrollerikohtaiset vakiot löytyvät kirjaston io.h-tiedostosta, keskeytys- ja signaalifunktiot interrupt.h ja signal.h-tiedostoista, flash-muistin käyttöön vaadittavat funktiot pgmspace.h tiedostosta ja merkkijonon käsittelyfunktiot string.h-tiedostosta.

3.2.1 Ohjelmatiedostot

Pääohjelmatiedosto

Pääohjelmatiedosto sisältää pääohjelmasilmukan, yleisfunktiot ja osan ohjelmiston määrittelyistä. Pääohjelmatiedostoon sisällytetään kääntämisen aikana muut ohjelman vaatimat tiedostot. Pääohjelmatiedosto on nimeltään main.c (osittain liitteessä 2) ja määrittelytiedosto on nimeltään main.h (liite 3). Muita ohjelmatiedostoja on neljä, jotka sisältävät UART-funktiot, SHT75-anturin vaatimat funktiot, EEPROM- ja flash-muistin käsittelyfunktiot ja tarvittavat määrittelyt.

Yleisfunktiot

Yleisfunktiot sisältävät mikrokontrollerin alustusfunktiot, A/D-muunnosfunktiot, virransäästöfunktiot, viivefunktiot ja muunnosfunktiot. Yleisfunktiot sijaitsevat tiedostossa main.c.

UART-funktiot

UART-funktiot sisältävät mikrokontrollerin sisäisen UART:n alustusfunktion ja UART-liikenteen lähetys- ja vastaanottofunktiot. Funktiot sijaitsevat tiedostossa uart.c.

SHT75-funktiot

SHT75-funktiot sisältävät kaikki kyseisen anturin toimintaan tarvittavat alustus- ja ohjausfunktiot. Funktioiden tiedostot ovat nimeltään sht75.h ja sht75.c.

Muistinkäsittelyfunktiot

Muistinkäsittelyfunktiot sisältävät kaikki EEPROM- ja flash-muistin lukemiseen ja kirjoittamiseen liittyvät funktiot ja tarvittavan bootloader-osion. Funktiot ovat memory.c-tiedostossa.

Makefile

Makefileen sisältyy kaikki ohjelman kääntämisessä tarvittavat tiedot. Makefilessa on määritelty mm. käytettävä mikrokontrolleri, käännöksen kohde ja sen yhteydessä pääohjelmaan liitettävät kirjastot ja objektitiedostot. Tiedosto on nimeltään Makefile (liite 4).

3.2.2 Mikrokontrollerin alustus

Virran kytkemisen jälkeen ensimmäisenä alustetaan mikrokontrollerin sarjaliikenneväylä UART0 ja antureiden käyttämät I/O-portit. I/O-liitäntöjä käytettäessä tulee mikrokontrollerin rekistereihin kirjoittaa kyseisten porttien toimintasuunta ja alkuarvot. Toimintasuuntarekisterin arvo määrittelee käytetäänkö I/O-porttia tulo- vai lähtöporttina. Rekisteriin kirjoitettu arvo "0" asettaa portin tuloksi ja arvo"1" lähdöksi.

3.2.3 A/D-muunnoksen suorittaminen

A/D-muunnoksen suorittamista varten tulee mikrokontrollerin 8-bittisten ADMUX- ja ADCSR-rekisterien bitit asettaa oikein. Rekisterien rakenne on esitetty kuvassa 5.

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ADC Control and Status Register – ADCSR

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Kuva 5 ADMUX- ja ADCSR-rekisterien rakenne /2/

ADMUX-rekisteristä valitaan muunnettava kanava bittien 0-5 (MUX0-MUX5) yhdistelmänä, jättämällä bitit nollaksi valitaan ensimmäinen muunnoskanava. Bitillä 5 (ADLAR) voidaan asettaa tuleva 10-bittinen muunnostulos alkamaan 16-bittisen tulosrekisterin eniten merkitsevästä bitistä. Biteillä 6 ja 7 valitaan muunnoksessa käytettävä vertailujännite ja bitti 6 asettaa vertailujännitteeksi käyttöjännitteen.

ADCSR-rekisteristä valitaan sopiva näytteenottotaajuus biteillä 0-2 (ADPS0-ADPS2), näytteenottotaajuus toteutetaan mikrokontrollerin sisäisellä kellotaajuuden jakajalla. Tässä työssä tarvitaan A/D-muunnoksia vain harvoin, joten jakaja kannattaa asettaa mahdollisimman suureksi. Asettamalla bitit 0-2 ykkösiksi valitaan kellotaajuuden jakajaksi suurin mahdollinen luku eli 128. Rekisterin bitillä 3 (ADIE) voidaan asettaa A/D-muunnin aiheuttamaan keskeytys muunnostuloksen valmistuttua. Bitti 4 käynnistää A/D-muunnoksen kun Varsinainen A/D-muunnin käynnistyy asettamalla ADCSR-rekisterin ADEN-bitti ykköseksi. Atmelin ohjeistuksen mukaan ensimmäinen muunnostulos tulee hylätä ja vasta seuraava on totuudenmukainen /2/.

Aiemman selvityksen mukainen C-koodi:

```
#include <avr/io.h>
/*
*  io.h sisältää mm. portti- ja rekisterimäärityksiä sekä
*  apufunktioita. _BV (bit) muuntaa bit:lle määritetyn numeron
*  tavuksi jossa vain numeron osoittama bitti on asetettu.
*  Mnemoninen bitti voidaan täten kirjoittaa helposti haluttuun
*  rekisteriin. Rekisterin ABC bitti DEF asetetaan seuraavasti:
*  ABC |= _BV (DEF); (bittitason OR-operaatio, bitti DEF=1)
*  Rekisterin ABC bitti DEF nollataan seuraavasti:
*  ABC &= ~(_BV (DEF)); (bittitason AND-operaatio, bitti DEF=0)
*  Lisäksi koko rekisteri voidaan esimerkiksi nollata hyvin
*  helposti DEF-bittiä lukuunottamatta käskyllä ABC = _BV(DEF);
*/
ADMUX = _BV (REFS0); // käytetään sisäistä referenssijännitettä
ADCSR = _BV(ADEN)|_BV(ADSC)|_BV(ADFR); // muunnin aina päällä
ADCSR |= _BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0); // kellotaajuus/128
ADCSR |= _BV (ADIF); // aloitetaan hylättävä muunnos
loop_until_bit_is_set (ADCSR, ADIF); // odotetaan valmistumista
ADCSR |= _BV (ADIF); // aloitetaan varsinainen muunnos
loop_until_bit_is_set (ADCSR, ADIF); // odotetaan valmistumista
```

Kun A/D-muunnos on suoritettu, tallennetaan 10-bittinen tulos mikrokontrollerin 16-bittiseen ADC-tulosrekisteriin, josta se voidaan lukea helposti haluttuun muuttujaan. Tässä työssä tulokselle tehdään vielä seuraavia operaatioita:

```
int h=105; // paineanturin korkeus merenpinnasta
float pressure,rpres; // muuttujat ilmanpainearvoja varten
float cor=0.35; // anturiyksilöstä ja ympäristöstä johtuva korjausarvo
pressure=ADC; // tallennetaan tulos pressure-muuttujaan
/*
* paineanturin siirtofunktio /3/:
* Vout=Vs*(.009*P-.095)±Error , Vs=5V -> P≈(5*(40*Vout+19))/9 kPa
* ADC tuloksen muodostuminen /2/:
* ADC=(Vin*1024)/Vref , Vref=5V
* Lopullinen tulos=(5.0*(40.0*((pressure*5.0)/1024)+19.0))/9.0+cor;
* Vielä muunto hehtopascaleiksi, hPa = 10 kPa
*/
pressure = 10*((5.0*(40.0*((pressure*5.0)/1024)+19.0))/9.0+cor);
rpres = pressure + h/8.0; // ilmanpaine merenpinnan korkeudella
```

3.2.4 Muut toiminnot

Laitteen muihin toimintoihin kuuluu muunmuassa yksinkertainen asetusohjelma ja käyttövalikko, SHT75-anturin alustus ja lukeminen, UART-funktiot, EEPROM- ja flash-muistin käsittely sekä BlueSMiRF-moduulin tarkemmat asetukset.

3.2.5 Virrankulutuksen vähentäminen ja ajastus

Käyttämättömät portit tulee asettaa sisääntuloiksi ja asettaa loogiseen 1-asentoon, jolloin ne kytketään sisäisellä ylösvetovastuksella +5V:n jännitteeseen ja kuluttavat siten mahdollisimman vähän virtaa.

Myös D/A-muunnin kannattaa sammuttaa kun sitä ei käytetä:

```
ACSR |= _BV (ACD) | ~_BV (ACIE); // sammuttaa ACD:n ja estää sen
                                 // aiheuttamat keskeytykset
```

Koko kytkennän virrankulutus ilman Bluetooth-yhteyttä on mahdollista laskea 3,1 milliampeeriin käyttämällä mikrokontrollerin sisäistä kellotaajuuden jakajaa ja virransäästötilaa. Jos Bluetooth-yhteys halutaan säilyttää päällä, nousee laitteen virrankulutus noin 10 milliampeerilla. Seuraavalla komennolla laskee kellotaajuus vain 1/129 osaan entisestään, eli noin 62 kHz:iin:

```
XDIV = 0x80; // kellotaajuuden jakaja 128+1 /2/
```

Tätä pienemmän virrankulutuksen saavuttaminen on mahdollista käyttäen mikrokontrolleriin ulkoista herätettä, joka aiheuttaa syvemmässä virransäästötilassa keskeytyksen ja herättää mikrokontrollerin takaisin täyteen toimintaan. Käytettävässä kytkennässä ei ole huomioitu ulkoisen keskeytyksen tarvetta, joten kyseisiä virransäästötiloja ei voida käyttää.

Virransäästön takia kaikenlaiset pidemmät ajastukset kannattaa toteuttaa keskeytysten avulla, tässä työssä noin sekunnin viive toteutetaan seuraavan sivun koodilla.

Laitteen ajastuksessa ja virransäästössä käytettävä C-koodi:

```
set_sleep_mode(SLEEP_MODE_IDLE); // asetetaan virransäästötila
sei(); // sallitaan keskeytykset
TCNT1 = 0; // asetetaan Counter1 arvo nollaksi
TIFR |= _BV (TOV1); // asettaa lipun Counter1 ylivuodosta
TIMSK |= _BV (TOIE1); // keskeytys Counter1 ylivuotolipusta
TCCR1B = _BV (CS10); // laskurin kellotaajuus = järjestelmän
                     // kellotaajuus
sleep(); // asettaa mikrokontrollerin unitilaan
cli(); // estetään keskeytykset
```

Keskeytysohjelmassa käytettävä Counter1 on 16 bittinen laskuri, jonka ylivuoto tapahtuu kun laskuri ylittää luvun 65535. Käytettäessä peruskellotaajuden jakoa (XDIV) luvulla 129, aiheutuu laskurin ylivuoto seuraavin väliajoin:

$$\frac{65536\,\text{lisäystä}}{62015\,\text{Hz}} \approx 1{,}06\,\text{sekuntia}$$

Mikrokontrolleri herää tällöin unitilastaan suorittamaan asetetun keskeytysohjelman, ja jatkaa sen jälkeen varsinaisen ohjelman vuorossa olevien käskyjen suoritusta. Jotta keskeytys toimisi oikein, tulee käytettävän kääntäjän ja kirjastojen kanssa kirjoittaa pääohjelmatiedoston loppuun tyhjä keskeytysohjelma, vaikka sille ei olisikaan muuta käyttöä:

```
INTERRUPT (SIG_OVERFLOW1)
{
}
```

Edellä oleva funktio suoritetaan vain kun Counter1-laskurissa tapahtuu ylivuoto, jokaiselle keskeytystyypille on olemassa oma signaalinimityksensä.

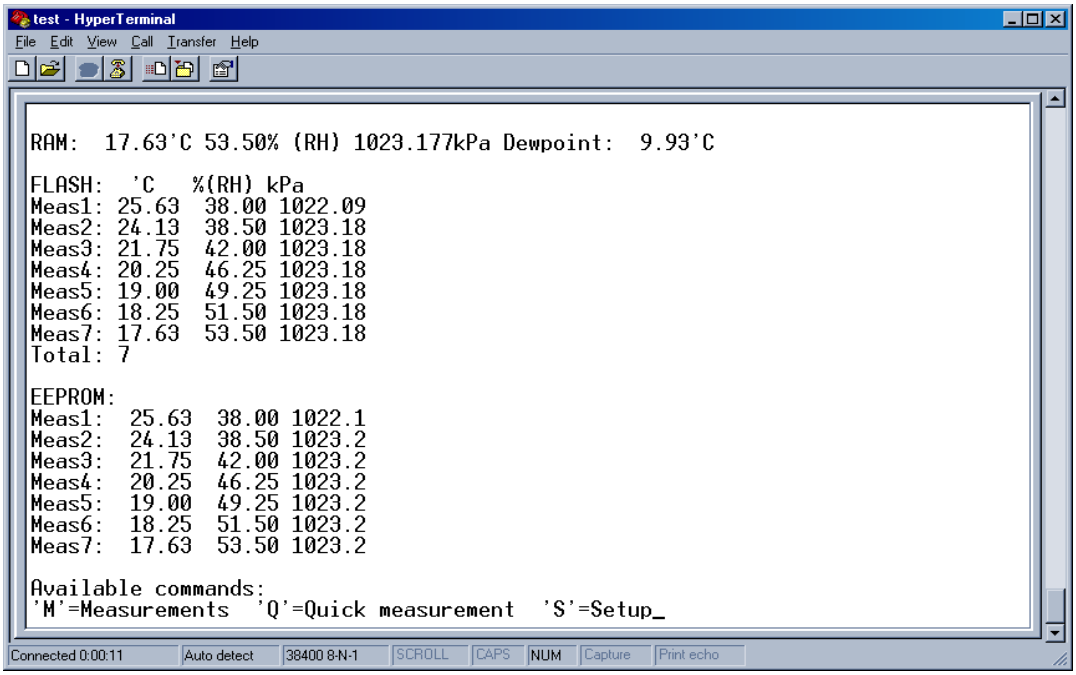
4 TESTAUS

4.1 Elektroniikan testaus

Elektroniikan testaamiseen ei ollut erityistä testaussuunnitelmaa. Antureiden testauksessa ilmanpaineanturin toiminta tarkistettiin ennen kytkemistä yleismittarilla. Ilmankosteus- ja lämpötila-anturi on täysin digitaalinen, joten sen testaaminen ei onnistu ilman oheiskytkentää ja logiikka-analysaattoria. Antureiden piirilevyyn kytkemisen jälkeen niiden toiminta testattiin myös ohjelmallisesti.

4.2 Ohjelmiston testaus

Ohjelmistonkaan testaamiseksi ei ollut erillistä testaussuunnitelmaa. Ohjelmiston testauksessa käytiin läpi käyttäjän mahdolliset syötteet, syötteistä aiheutuva toiminta ja mahdolliset tulostukset. Erityisen testauksen alaisena oli SHT75-anturin, A/D-muuntimen, USB-sarjaliikenneväylän ja Bluetooth-yhteyden toimivuus. Viimeisessä vaiheessa kiinnitettiin huomiota mittausten toistettavuuteen ja virransäästötilojen toimivuuteen. Virransäästön testaamisessa käytettiin apuna yleismittaria. Perustoiminnassa laitteen läpi kulki noin 23 mA:n sähkövirta. Virransäästötilassa laitteen kuluttama virtamäärä väheni tyypillisesti 20 mA:lla jolloin laitteen kokonaiskulutus oli enää noin 3 milliampeeria.



Kuva 6 Anturitietojen analysointi ja lukeminen terminaaliohjelmasta

4.3 Testauksen luotettavuus ja tulokset

Käytettävät anturit toimivat erittäin luotettavasti laajalla käyttöalueella, yhdistetty ilmankosteus- ja lämpötila-anturi sisältää tehtaalla asetetut kalibrointitiedot ja myös ilmanpaineanturi antaa ulostuloonsa valmiiksi lämpötilakorjatun tuloksen.

Lämpötila- ja kosteusanturin luotettavuuden testaamiseksi voidaan antureiden tietoja verrata pitkällä ajalla, muuttuvissa olosuhteissa, useisiin luotettavaksi todettuihin ja/tai kalibroituihin lämpötila- ja kosteusmittareihin.

Ilmanpaineanturin luotettavuuden testaaminen ja täsmällinen kalibrointi on hankalampaa, koska yleisesti saatavilla olevat mittarit eivät ole kovin tarkkoja ja niiden tulisi sijaita täsmälleen samassa tilassa testattavan anturin kanssa. Tässä työssä käytettiin ilmanpaineanturin kalibroimiseksi Ilmatieteen laitoksen Internet-sivuilta /7/ löytyvää tarkkaa ilmanpainetietoa lähimmältä, eli Hatanpään sääasemalta, ja lisättiin anturin lukemaan laitteen sijaintipaikan korkeudesta aiheutuva muutos ilmanpaineeseen (1 hPa jokaista 8 metriä kohden).

Testauksen aikana laitteen ja ohjelmiston toiminnasta ei löydetty vakavia puutteita.

5 YHTEENVETO

Tutkintotyön tavoitteena oli suunnitella ja toteuttaa laitteisto, jolla voidaan luotettavasti mitata erilaisia ilmastoon liittyviä suureita, ja joka kykenee välittämään mittaustiedon käyttäjän laitteelle USB-väylää ja langatonta Bluetooth-yhteyttä pitkin. Lisäksi tavoitteena oli tutkia ATmega-mikrokontrollerin ja erilaisten antureiden toimintaa ja ohjelmointia.

Työ osoittautui haasteelliseksi koska AVR-sarjan mikrokontrollereista ei ollut aiempaa kokemusta, eikä vastaavanlaisesta laitteistosta löytynyt valmiita esimerkkejä. Laitteiston kytkentäkaavion ja piirilevyn suunnittelemiseksi oli valmistajilta onneksi saatavissa riittävästi tietoa. Haasteellisimmaksi osoittautui toimivan ja monipuolisen ohjelmiston suunnittelu.

Laitteisto ohjelmistoineen osoittautui varsin toimivaksi kokonaisuudeksi, joskin virransäästöominaisuuksia ja keskeytysten hallintaa voisi parantaa entisestään lisäämällä laitteeseen muutaman ulkoisen komponentin.

Valmiilla järjestelmällä on kaupallista potentiaalia, koska laitteelle sopivia käyttökohteita löytyy runsaasti eri teollisuuden aloilta ja logistiikasta. Laitetta voidaan käyttää muunmuassa valvomaan ja mittaamaan herkän materiaalin kuljetusympäristöä ja estämään näin esimerkiksi elintarvikkeiden ennenaikainen pilaantuminen. Laite on myös varsin edullinen valmistaa ja käyttää.

Jatkossa laitteen toiminta on helposti laajennettavissa ylimääräisten liitäntöjen, ohjelmiston modulaarisuuden ja runsaan vapaan ohjelmamuistin vuoksi. Myös olemassa olevien antureiden käyttöä voidaan kehittää ja tietoliikenneyhteyksien nopeutta säätää tarpeen mukaan. Lisätietoja laitteesta ja ohjelmistosta saa suoraan tekijältä ohjelmiston lähdekoodin yhteystietojen kautta.

LÄHTEET

1. SHT75-datalehti. http://www.sensirion.com/images/getFile?id=25.
Sensirion Inc. [26.10.2006]

2. ATmega128-datalehti.
http://atmel.com/dyn/resources/prod_documents/doc2467.pdf.
Atmel Corporation. [26.10.2006]

3. MPXAZ6115A-datalehti.
http://www.freescale.com/files/sensors/doc/data_sheet/MPXAZ6115A.pdf.
Freescale Inc. [26.10.2006]

4. AVR-tuoteperhe. http://www.atmel.com/products/AVR/.
Atmel Corporation. [26.10.2006]

5. All About USB. http://www.datapro.net/techinfo/usb_info.html.
DataPro International Inc. [26.10.2006]

6. FT232BL-datalehti.
http://www.ftdichip.com/Documents/DataSheets/ds232bl18.pdf.
Future Technology Devices International Ltd. [26.10.2006]

7. Sää ja ilmasto. http://www.fmi.fi/saa/ Ilmatieteenlaitos. [26.10.2006]

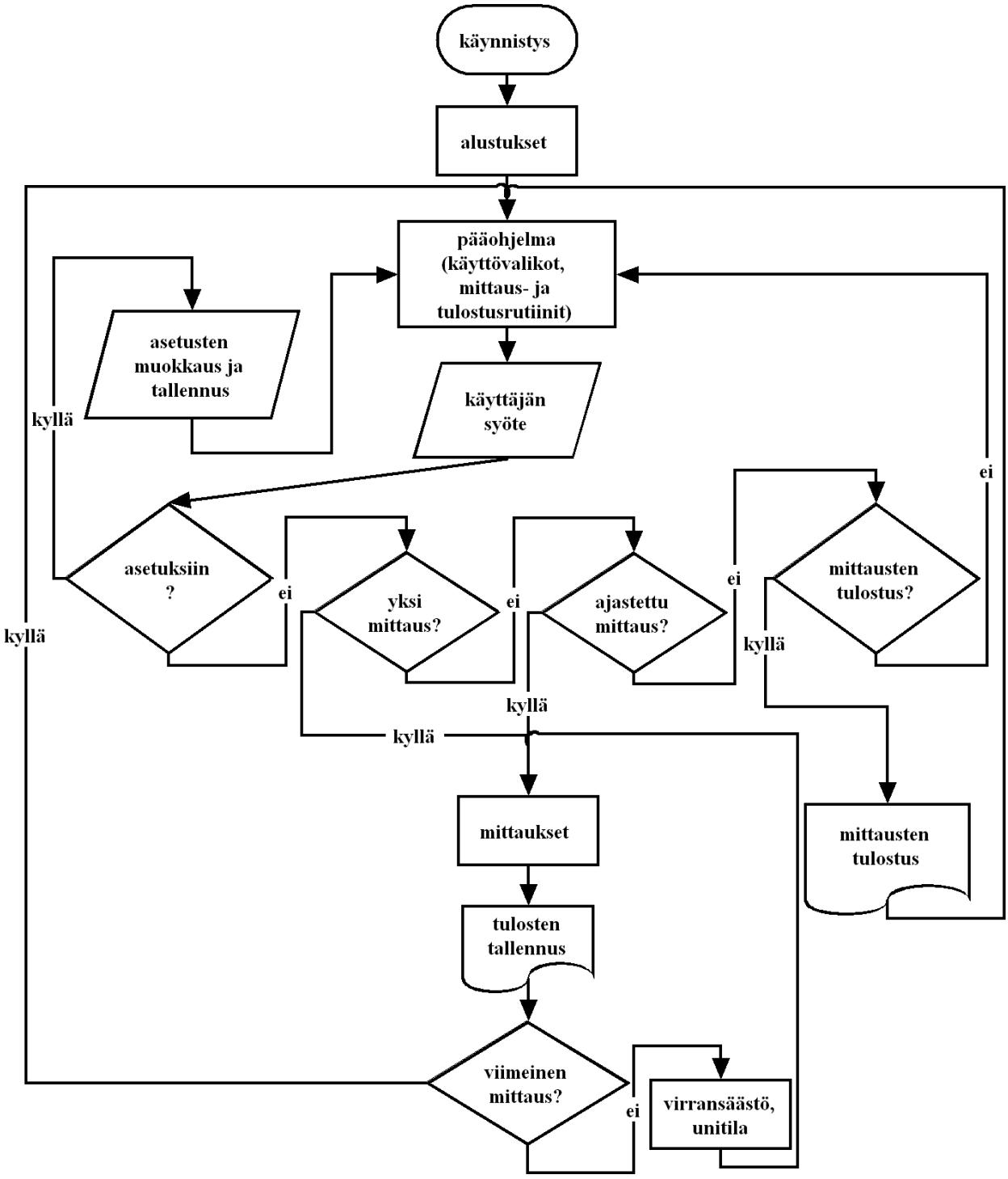
8. Bluetooth - The Official Bluetooth Wireless Info Site.
http://www.bluetooth.com/Bluetooth/Learn/Technology/Specifications/.
Bluetooth SIG. [26.10.2006]

9. BlueSMiRF-datalehti.
http://www.sparkfun.com/datasheets/RF/BlueSMiRF_v1.pdf.
Spark Fun Electronics. [26.10.2006]

LIITTEET

1. Ohjelmiston toimintakaavio
2. Osittainen lähdekoodi main.c
3. Lähdekoodi main.h
4. Lähdekoodi Makefile
5. Laitteen täydellinen kytkentäkaavio, taitettu A3
6. Laitteen piirilevykuva

Liite 1. Ohjelmiston toimintakaavio



Liite 2. Osittainen lähdekoodi main.c

```
/*****************************************************************
*                Product:   Weather Station WS1                  *
*      Author, Copyright:   Timo Lähteenmäki                     *
*                  Email:   timo@tlarc.com                       *
*                                                                *
*  ASK PERMISSION BEFORE USING ANY PART OF THE FOLLOWING CODE    *
*****************************************************************/

#include "main.h" // yleiset määrittelyt
#include "sht75.h" // SHT75-anturin alustus- ja käyttöfunktiot

// Globaalit muuttujat
float EEMEM tempe[5000];  // EEPROM-taulukko lämpötila-arvoille
float EEMEM rhum[5000];   // EEPROM-taulukko kosteusarvoille
float EEMEM rpres[5000];  // EEPROM-taulukko ilmanpainearvoille
int valueindex=0;         // Taulukon indeksin osoitin

/*****************************************************************
*   Function name : main
*   Returns :       None
*   Parameters :    None
*   Purpose :       Main loop
*****************************************************************/
int main (void)
{
      // Muuttujien alustuksia
      float pressure;
      ...

      // Ajastimien ja unitilan alustus
      TCNT1 = 0; // asetetaan Counter1 arvo nollaksi
      TIFR |= _BV (TOV1); // asettaa lipun Counter1 ylivuodosta
      TIMSK |= _BV (TOIE1); // keskeytys Counter1 ylivuotolipusta
      TCCR1B = _BV (CS10); // laskurin kellotaajuus = järjestelmän kellotaajuus
      ACSR |= _BV (ACD) | ~_BV (ACIE); // sammutetaan ACD, estetään keskeytys
      set_sleep_mode(SLEEP_MODE_IDLE); // asetetaan virransäästötila

      // Alustetaan oheislaitteet ja portit
      ...
```

```
	// Alustetaan UART
	UART_Init(((xtal/16)/baud)-1);

	// Tulostus UART:n kautta
	print("\r\n\n\n\n\n\n\n\n\n\n\n\n");
	print("\r\n                        Weather Station 1 \r\n\n");

	// Päättymätön pääohjelma
	while(1)
	{
		// Valikoiden ja kontrollien tulostus
		...

		// Yksittäinen SHT75:n lämpötilamittaus
		Temp = TemperatureSHT75Func();

		// Yksittäinen A/D-muunnos
		ADMUX = _BV (REFS0); // käytetään sisäistä referenssijännitettä
		ADCSR = _BV(ADEN)|_BV(ADSC)|_BV(ADFR); // muunnin aina päällä
		ADCSR |= _BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0); // kellotaajuus/128
		ADCSR |= _BV (ADIF); // aloitetaan muunnos
		loop_until_bit_is_set (ADCSR, ADIF); // odotetaan muunnoksen
                                             // valmistumista
		pressure = ADC; // sijoitetaan tulos pressure-muuttujaan

		// Odotetaan 125 ms
		Delay(125);

		// Muunnetaan tulokset BCD-muotoon ja tehdään tarvittavat
		// laskutoimitukset
		Float2BCD(Temp,TempBCD,2);
		...

		// Tallennetaan tulokset
		Store_Value(rpres,pressure);
		...
		valueindex++; // kasvatetaan taulukon indeksin osoitinta

		// Unitila 2 sekunniksi
		Sleepxs(2);
	}
}
```

```
/************************************************************
*   Function name : Float2BCD
*   Returns :       BCD
*   Parameters :    float f, char* BCD,int D
*   Purpose :       Converts float to BCD
************************************************************/
void Float2BCD(float f, char* BCD,int D)
{
       dtostrf(f,6,D,BCD); // muuntaa luvun f (maksimi 5 numeroa + desimaalierotin)
                           // BCD-muotoon D määrällä desimaaleja ja tallentaa tuloksen
                           // annettuun merkkijonoon
}

/************************************************************
*   Function name : Delay
*   Returns :       None
*   Parameters :    unsigned int millisec
*   Purpose :       Simple inaccurate delay-loop
************************************************************/
void Delay(unsigned int millisec) // aiheuttaa n. 1 ms viiveen @ 8MHz
{
    unsigned int i;

    while (millisec--)
        for (i=0; i<1400; i++)   // >5 kellojaksoa/kierros
         asm volatile ("nop"::); // assembler-käsky, ei tehdä mitään
}

/************************************************************
*   INTERRUPT AND SIGNAL HANDLING FUNCTIONS
*   Purpose : Interrupt and signal handling
************************************************************/
INTERRUPT (SIG_OVERFLOW1) // keskeytysfunktio Counter1 ylivuotoa varten
{
}
SIGNAL (SIG_UART0_RECV) // Signaalifunktio UART0:n vastaanotosta
{
}
```

Liite 3. Lähdekoodi main.h

```
/*****************************************************************
*                   Product:   Weather Station WS1                *
*         Author, Copyright:   Timo Lähteenmäki                   *
*                      Email:   timo@tlarc.com                    *
*                                                                 *
* ASK PERMISSION BEFORE USING ANY PART OF THE FOLLOWING CODE      *
*****************************************************************/

#include <avr/io.h>              // tärkeimmät mikrokontrollerikohtaiset määrittelyt
#include <stdlib.h>              // C-kielen standardikirjasto
#include <avr/interrupt.h>       // keskeytyskäsittelyt
#include <avr/signal.h>          // signaalikäsittelyt
#include <avr/pgmspace.h>        // funktioita flash-muistin käsittelyyn
#include <string.h>              // merkkijonofunktiot
#include <avr/sleep.h>           // unitilan määrittely- ja apufunktiot

#define F_CPU 8000000UL          // mikrokontrollerin kellotaajuus [Hz]
#define xtal F_CPU               // kiteen kellotaajuus [Hz]
#define baud 38400               // UART:n baudinopeus

// Helpompi tapa yksittäisten porttien kääntelyyn
#define sbi(port, bit) port |= (1<<bit)  // asetetaan bit:n osoittama bitti portista
#define cbi(port, bit) port &= ~(1<<bit) // nollataan bit:n osoittama bitti portista

// Määritellään sleep()-funktio: sallitaan unitila, ajetaan unitila, estetään unitila
// Unitila voimassa kunnes tapahtuu keskeytys
#define sleep()
do { MCUCR |= _BV(SE); asm volatile ("sleep"::); MCUCR &= ~_BV(SE);} while(0)

// Määritellään EEMEM sijaitsevaksi mikrokontrollerin eeprom-osiossa
#define EEMEM __attribute__ ((section(".eeprom")))

// Flash-muistin määritykset
#define FLASH_DECL(decl) decl PROGMEM            // flash-muisti ohjelmamuistissa
#define FLASH_READ_BYTE(addr) pgm_read_byte(addr) // luku pgm_read_byte funktiolla
typedef unsigned int MyAddressType;              // osoitetyypin määritys
typedef unsigned char FLASH_DECL(*MyFlashCharPointer);     // osoitin flash-muistiin
unsigned char ReadFlashByte(MyAddressType flashStartAdr);  // yhden tavun luku
void write_flash(unsigned int code_page,uint8_t page[]);   // kirjoitusfunktio
void boot_program_page (uint32_t page, uint8_t *buf); // yhden flash-sivun kirjoitus,
                                                      //  boot-osion avulla
```

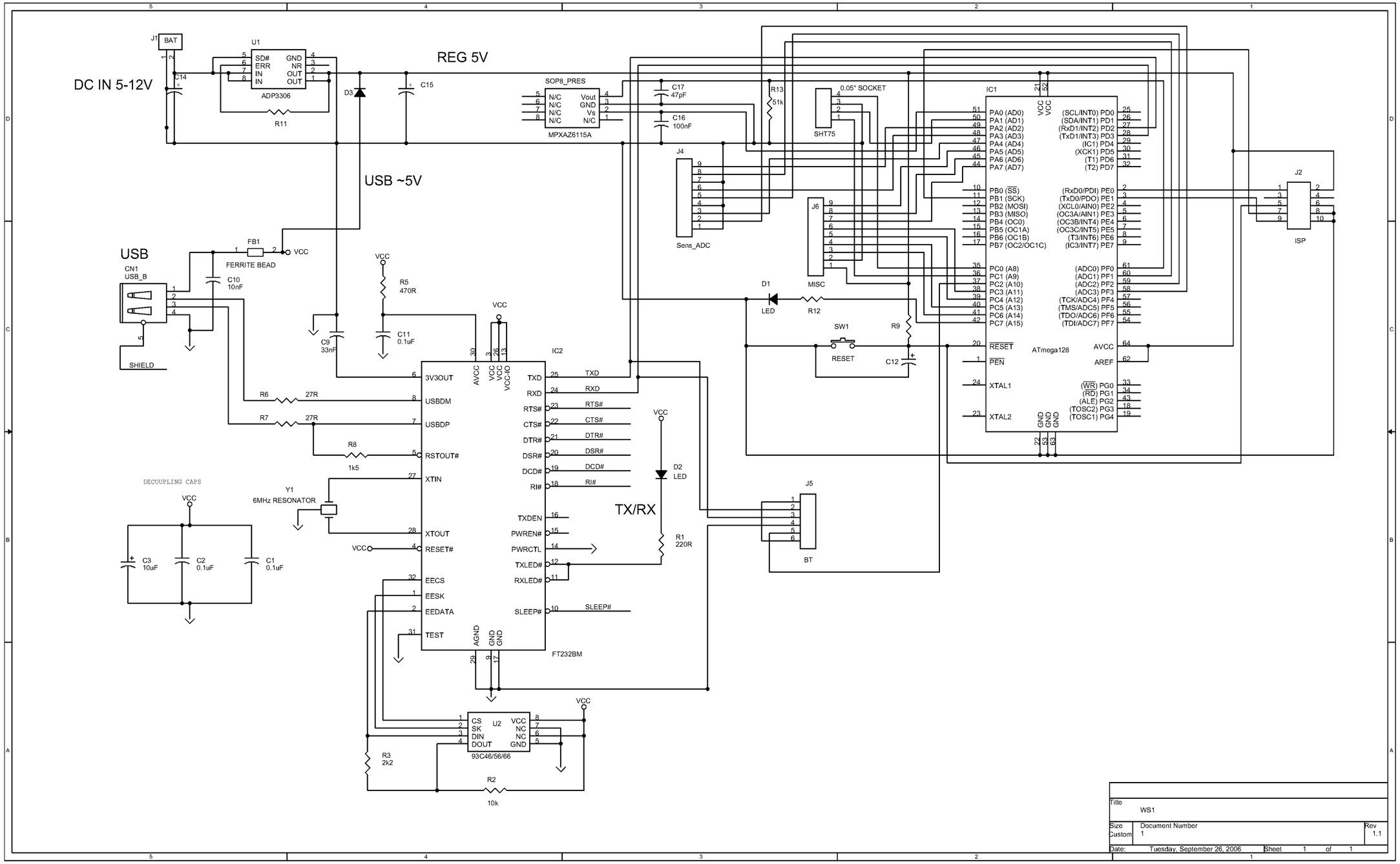
```
// UART-funktiot
void UART_Init(unsigned int br);              // alustus baudinopeudelle br
unsigned char UART_Receive(void);             // vastaanottaa yhden merkin
void UART_Transmit(unsigned char data);       // lähettää yhden merkin
void UARTTextIn (char* str, int length);      // vastaanottaa length-pituisen
                                              // merkkijonon ja tallentaa sen str-
                                              // muuttujaan
void print(char *text);   // tulostaa merkkijonon

// Muunnos- ja tallennusfunktiot
void Float2BCD(float f, char* BCD, int D);    // muuntaa float-muuttujan sisällön BCD-
                                              // koodatuksi merkkijonoksi, BCD-
                                              // muuttujaan, D desimaalilla
void Store_Value(float values[], float value); // tallentaa annetun arvon taulukkoon

// Ajastus- ja virransäästöfunktiot
void Delay(unsigned int millisec);            // viivefunktio, lyhyitä viiveitä varten
void Sleepxs(unsigned int sec);               // unitilafunktio, sekuntien viiveitä ja
                                              // virransäästöä varten
void PWRSAVE(void);                           // toistaiseksi voimassa oleva unitila
```

Liite 4. Lähdekoodi Makefile

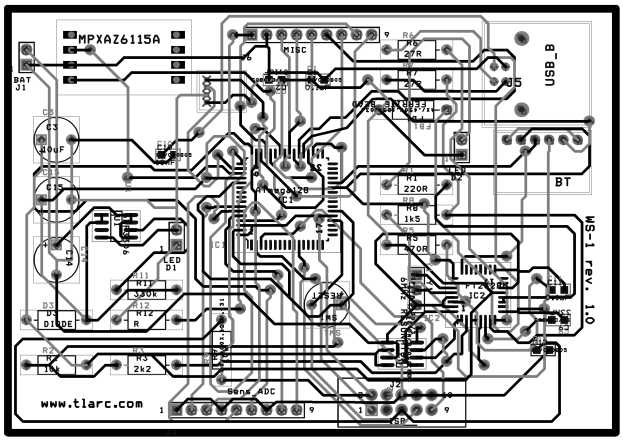
```
# WinAVR Makefile for TLARC WS1 v1
# make all = Make software.
# make clean = Clean out built project files.

# Specify mcu, output format, target name, optimization and bootloader location
MCU = atmega128
FORMAT = ihex
TARGET = main
OPT = s
BOOTLOAD = 0x1E000

# Sources
SRC = $(TARGET).c sht75.c uart.c memory.c
EXTRAINCDIRS = d:\other\avr\include\avr

# Compiler flags
CFLAGS = -g -O$(OPT) -funsigned-char -funsigned-bitfields -fpack-struct -fshort-enums\
-Wall -Wstrict-prototypes -Wa,-adhlns=$(<:.c=.lst) $(patsubst %,-I%,$(EXTRAINCDIRS))
CFLAGS += -std=gnu99

# Linker flags and libraries
LDFLAGS = -Wl,-Map=$(TARGET).map,--cref,--section-start=.bootloader=$(BOOTLOAD),-lm

# Shell commands
DIRAVR := $(shell pwd | sed 's@\(.*\)/apps.*$$@\1@' )
DIRAVRBIN = $(DIRAVR)/bin
DIRAVRUTILS = $(DIRAVR)/utils/bin
DIRINC = .
DIRLIB = $(DIRAVR)/avr/lib
SHELL = sh
CC = avr-gcc
OBJCOPY = avr-objcopy
OBJDUMP = avr-objdump
SIZE = avr-size
REMOVE = rm -f
COPY = cp

HEXSIZE = $(SIZE) --target=$(FORMAT) $(TARGET).hex
ELFSIZE = $(SIZE) -x -A $(TARGET).elf

MSG_FLASH = Creating load file for Flash:
MSG_EEPROM = Creating load file for EEPROM:
MSG_EXTENDED_LISTING = Creating Extended Listing:
MSG_SYMBOL_TABLE = Creating Symbol Table:
MSG_LINKING = Linking:
MSG_COMPILING = Compiling:
MSG_ASSEMBLING = Assembling:
MSG_CLEANING = Cleaning project:

OBJ = $(SRC:.c=.o) $(ASRC:.S=.o)
LST = $(ASRC:.S=.lst) $(SRC:.c=.lst)
ALL_CFLAGS = -mmcu=$(MCU) -I. $(CFLAGS)
ALL_ASFLAGS = -mmcu=$(MCU) -I. -x assembler-with-cpp $(ASFLAGS)

all: $(TARGET).elf $(TARGET).hex $(TARGET).eep $(TARGET).lss $(TARGET).sym sizeafter


sizeafter:
	@if [ -f $(TARGET).elf ]; then echo; echo $(MSG_SIZE_AFTER); $(ELFSIZE); fi

%.hex: %.elf
	@echo
	@echo $(MSG_FLASH) $@
	$(OBJCOPY) -O $(FORMAT) -R .eeprom $< $@
```

```
%.eep: %.elf
	@echo
	@echo $(MSG_EEPROM) $@
	-$(OBJCOPY) -j .eeprom --set-section-flags=.eeprom="alloc,load" \
	--change-section-lma .eeprom=0 -O $(FORMAT) $< $@
%.lss: %.elf
	@echo
	@echo $(MSG_EXTENDED_LISTING) $@
	$(OBJDUMP) -h -S $< > $@
%.sym: %.elf
	@echo
	@echo $(MSG_SYMBOL_TABLE) $@
	avr-nm -n $< > $@
.SECONDARY : $(TARGET).elf
.PRECIOUS : $(OBJ)
%.elf: $(OBJ)
	@echo
	@echo $(MSG_LINKING) $@
	$(CC) $(ALL_CFLAGS) $(OBJ) --output $@ $(LDFLAGS)
%.o : %.c
	@echo
	@echo $(MSG_COMPILING) $<
	$(CC) -c $(ALL_CFLAGS) $< -o $@
%.s : %.c
	$(CC) -S $(ALL_CFLAGS) $< -o $@
%.o : %.S
	@echo
	@echo $(MSG_ASSEMBLING) $<
	$(CC) -c $(ALL_ASFLAGS) $< -o $@

clean: clean_list
clean_list :
	@echo
	@echo $(MSG_CLEANING)
	$(REMOVE) $(TARGET).hex
	$(REMOVE) $(TARGET).eep
	$(REMOVE) $(TARGET).elf
	$(REMOVE) $(TARGET).map
	$(REMOVE) $(TARGET).sym
	$(REMOVE) $(TARGET).lss
	$(REMOVE) $(OBJ)
	$(REMOVE) $(LST)
	$(REMOVE) $(SRC:.c=.s)
	$(REMOVE) $(SRC:.c=.d)

# Automatically generate C source code dependencies
%.d: %.c
	set -e; $(CC) -MM $(ALL_CFLAGS) $< | sed 's,\(.*\)\.o[ :]*,\1.o \1.d : ,g' > $@;
	\[ -s $@ ] || rm -f $@
```

Liite 5. Laitteen täydellinen kytkentäkaavio



Liite 6. Laitteen piirilevykuva



DRILL CHART				
SYM	DIAM	TOL	QTY	NOTE
×	0.019		4	
+	0.022		4	
△	0.028		47	
◇	0.031		30	
⊠	0.034		5	
⊞	0.037		24	
⧉	0.038		8	
○	0.042		10	
TOTAL			132	