LÄMPÖMITTARI

Matti Rasinen

Opinnäytetyö
Kesäkuu 2011
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikkan suuntautumisvaihtoehto

RASINEN, MATTI : Lämpömittari

Opinnäytetyö 30 s., liitteet 13s.
Kesäkuu 2011

Työn tarkoituksena oli tehdä tietokoneella tai matkapuhelimella etäluettava lämpömittari, jolla pystytään mittaamaan lämpötilaa sisätiloissa. Lämpömittarista lähetetään mitattu lämpötila sarjaportin avulla tietokoneelle, ja tietokoneelta lämpötiladata välitetään langattomasti Bluetooth-tekniikkaa käyttäen matkapuhelimelle. Työhön sisältyi lämpömittarin prototyypin suunnittelu ja toteuttaminen sekä tietokoneella ja matkapuhelimella käytettävien lämpödatan lukusovellusten ohjelmoiminen.

Lämpömittarin sulautetun ohjelmiston kehitys tapahtui Olimexin ATmega 128 -alustalla ja Atmel AVR Studio 4 -kehitysympäristössä. Tietokoneella lämpömittarin tuottama data vastaanotetaan varta vasten sitä varten kehitetyllä ohjelmalla, joka on ohjelmoitu käyttäen Qt-kehitysympäristöä. Qt on graafisten käyttöliittymien kehitysympäristö. Matkapuhelimen lämpötiladatan vastaanottosovellus ohjelmoitiin Eclipse-kehitysympäristössä ohjelmointikielen ollessa Java.

Avainsanat: Lämpötilamittaus, AVR, Qt, Android.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer Systems Engineering
Option of Embedded Systems and Electronics

RASINEN, MATTI : Thermometer

Bachelor’s thesis 30 pages., appendices 13 pages.
June 2011

The purpose of this thesis work was to design thermometer that can be read remotely by using a computer or a mobile phone. Thermometer can be used in indoor conditions. The measured temperature is sent via serial connection to a computer, and from the computer it is possible to pass the measured data wirelessly to a mobile phone using Bluetooth technology. This thesis work included designing and assembling a prototype of the thermometer, and designing and programming the temperature reading applications both for a computer and a mobile phone.

Thermometer's embedded software was developed using Olimex ATmega 128 development board and Atmel AVR Studio 4 -design environment. On a computer the measured temperature data is received with an application developed for this purpose. The application was developed using Qt, which is graphical user interface development software. The temperature reading application for a mobile phone was designed in Eclipse environment, and the programming language was Java.

Keywords: Thermometer, AVR, Qt, Android.

ESIPUHE

Työn tarkoituksena oli rakentaa lämpömittari, jolla voi mitata huoneen lämpötilaa. Työssä käytetyt Qt- ja Eclipse-kehitysympäristöt olivat minulle täysin tuntemattomia entuudestaan, joten työn opettavaisin osuus tuli kahden eri kehitysympäristön ja Java-ohjelmointikielen opettelusta. Työn hyöty oli itselleni suuri, sillä enää ei tarvitse kävellä paria metriä lämpömittarille, vaan sisälämpötilan voi katsoa suoraan tietokoneen ja matkapuhelimen ruudulta.

Tampereella kesäkuussa 2011

Matti Rasinen

SISÄLLYSLUETTELO

LYHENTEIDEN JA TERMIEN LUETTELO

Android	Googlen mobiilikäyttöjärjestelmä
ATmega	Atmel AVR-tuoteperheen tuotesarja
Bluetooth	Langattoman tiedonsiirron teknologia
IDE	Integrated development environment. Tekstieditori ja kääntäjä samassa paketissa
JTAG	Joint Test Action Group. IEEE 1149.1 -standartin mukainen yleisesti käytettävä mikropiirien testaus- ja kehitysapuväline.
USART	Universal Synchronous/Asynchronous Receiver-Transmitter, sarjaliikenteen lähetys- ja vastaanottopiiri.
RFCOMM	Radio Frequency Communication, Bluetoothin protokolla, joka emuloi RS232-sarjaportteja.
SPP	Serial Port Profile. Bluetoothin profiili, joka pohjautuu ETSI 07, 10 ja RFCOMM –protokolliin.

1 JOHDANTO

Opinnäytetyönä suunniteltiin ja rakennettiin lämpömittari, joka lähettää lämpötilatiedon sarjaväylää käyttäen tietokoneelle ja tietokone puolestaan lähettää tiedon edelleen Bluetoothin avulla matkapuhelimelle. Idea laitteen suunnittelusta lähti mielenkiinnosta mikrokontrollerien ohjelmointiin ja kuinka sillä voisi toteuttaa lämpötilan-mittauksen.

Tietokoneen sarjaportille tulevan lämpötilatiedon voisi lukea Windowsin Terminaali-ohjelmalla, mutta tämän työn tarkoituksena on sulautetun piirin ohjelmoimisen ohella oppia graafisen käyttöliittymän suunnittelua ja ohjelmointia Qt-kehitysympäristössä. Tämän vuoksi lämpötilan lukemiselle tehdään oma sovellus.

Tällä hetkellä markkinoiden nopeimmin kasvavana mobiilijärjestelmänä Linux-pohjainen Android on mielenkiintoinen alusta toteuttaa lämpötilan lukeminen matkapuhelimella. Android-sovelluksien kehittäminen on tällä hetkellä rajoittunut Java-ohjelmointikieleen, ja yleisesti kehitysympäristönä käytetään Eclipse-kehitysympäristöä

2 KÄYTETYT IDE-OHJELMISTOT

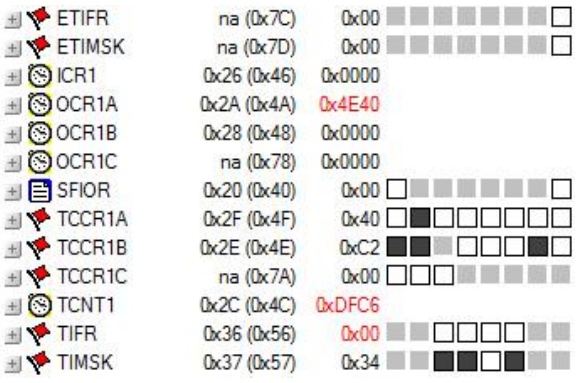
Tämä luku käsittelee työssä käytettyjä kehitysympäristöjä (IDE, Integrated Development Environment). Mikrokontrollerin ohjelmointiin käytettiin Atmelin AVR Studiota. Loppukäyttäjälle näkyvät tietokone- ja matkapuhelinsovellukset ohjelmoitiin Qt- ja Eclipse-kehitysympäristöllä.

2.1 Atmel AVR Studio

Atmel AVR Studio on Atmelin ilmainen kehitysympäristö AVR-mikrokontrollereille. AVR-sovellusten kehittäminen tapahtuu Windows-ympäristössä.

AVR Studion ominaisuuksiin kuuluu ulkopuolisten C-kääntäjien tuki, ja melko usein siinä käytetäänkin WinAVR-kääntäjää. AVR Studion ominaisuuksia ovat myös simulaattori sekä tuki monelle eri AVR-kehitystyökaluille, joita ovat esimerkiksi JTAG ICE ja STK500.

Simulaattorilla voidaan simuloida ohjelman toimintaa askel kerrallaan. Simulaattorilla voidaan myös tutkia, mitä eri rekisterit pitävät sisällään, tutkia eri porttien tiloja, ja tutkia, mitä eri arvoja eri muistipaikoista löytyy. Kuvassa 1 näytetään esimerkkinä Timer-Counter1-rekisterien sisältö.



Kuva 1. Esimerkki Timer-Counter1-simuloinnista

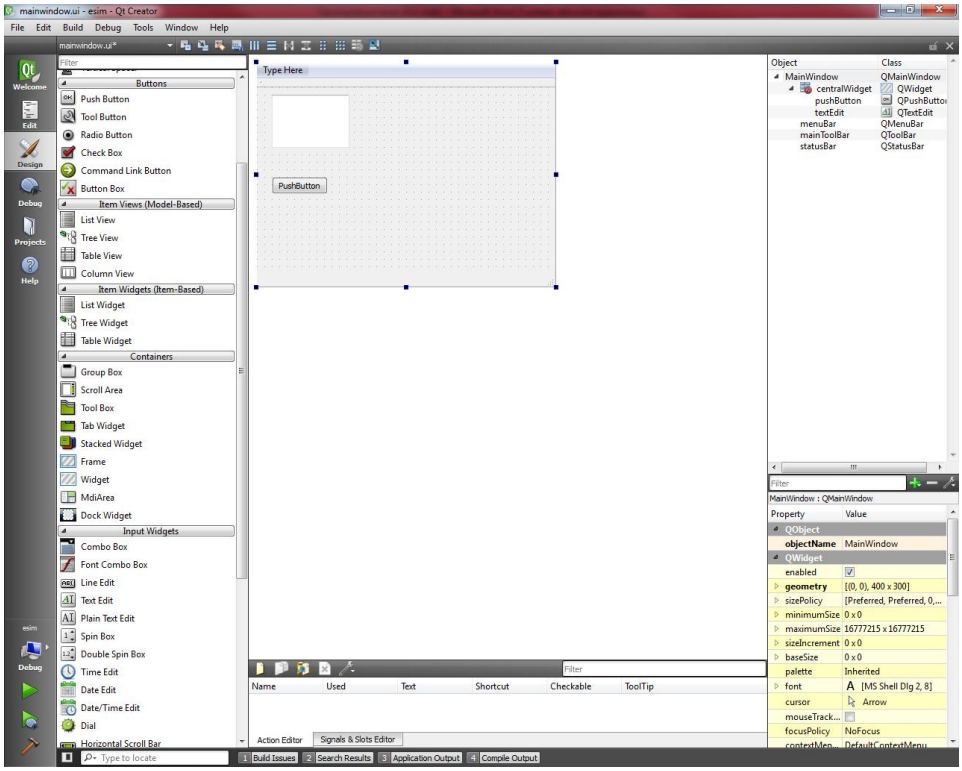
AVR Studion parhaimpia ominaisuuksia on mahdollisuus testata kontrolleriin ladattua ohjelmaa JTAG-debuggerilla. Käytännössä tämä tarkoittaa, että ohjelmaa voidaan testata suoraan kontrollerilla, sekä tutkia rekistereiden arvoja samalla tavalla kuin simulaat-

torissa. Erona simulaattoriin on mahdollisuus testata ulkoisen elektroniikan vaikutus ohjelmaan, joka tarkoittaa esimerkiksi, reagoiko ohjelma oikein ulkoisiin keskeytyksiin, syttyykö ledi palamaan tai saadaanko oikea A/D-muunnostulos.

2.2 Qt Creator

Qt Creator on alustariippumaton graafisten käyttöliittymien kehitystyökalu, jolla voi ohjelmoida graafisia käyttöliittymiä Windowsille, Linuxille, Mac OS X- ja eri Unix-pohjaisille käyttöjärjestelmille. Qt Creatorin ominaisuuksiin kuuluvat kehittynyt teks-tieditori, versionhallinta, käyttöliittymien suunnittelutyökalut, simulaattori ja ohjelman helppo sovittaminen mobiililaitteisiin.

Käyttöliittymäsuunnittelutyökalun avulla voidaan helposti tehdä monipuolisia käyttöliit-tymiä. Käyttöliittymiä voidaan tehdä joko käyttämällä valmiita muokattavissa olevia widgettejä ja dialogeja tai voidaan luoda oma komponentti käyttöliittymään. Kuvassa 2 on esimerkki Qt Creatorin käyttöliittymien suunnittelutyökalusta.

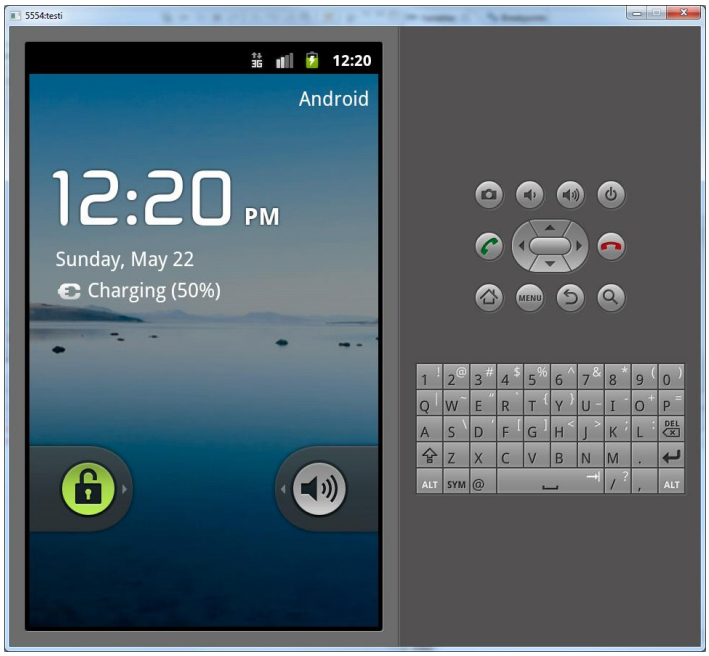


Kuva 2. Käyttöliittymien suunnittelutyökalu

Qt Creatorin mukana tulevalla simulaattorilla voidaan simuloida muun muassa Nokian matkapuhelimille tehtyä sovellusta. Simulaattorin avulla ohjelmaa ei tarvitse aina siirtää matkapuhelimelle testausta varten, joten sen avulla säästyy aikaa. /1/

2.3 Eclipse

Eclipse on avoimen lähdekoodin ohjelmointityökalu usealle eri käyttöjärjestelmälle, ja se tukee laajennuksia kolmansilta osapuolilta. Tässä työssä Eclipsen kanssa käytetään Googlen julkaisemia laajennuksia, joiden avulla voidaan ohjelmoida Eclipsellä Android-ille tarkoitettuja sovelluksia. Laajennuksien mukana tulee simulaattori, jolla voidaan testata sovelluksien toimintaa. Simulaattorin heikkona puolena on, ettei sillä voi testata Bluetooth-sovelluksia. Kuvassa 3 esitellään Android-simulaattorin käyttöliittymä.



Kuva 3. Android-simulaattori

Laajennuksien mukana tulee myös mahdollisuus debugata simulaattorissa ja matkapuhelimessa tapahtuvaa toimintaa. Tällä tavoin voidaan helposti paikallistaa ohjelmistossa olevia toiminnallisia virheitä.

3 LAITTEISTON KEHITYS

Tässä luvussa kerrotaan lämpömittarin laitteiston kehityksestä, kehityksessä käytetyistä komponenteista ja kehitysalustasta.

3.1 Lämpöanturi

Lämpötila-anturina käytetään Smartecin SMT160-30-TO92-lämpötila-anturia. Kyseinen lämpötila-anturi valittiin, koska lämpötilatieto saadaan siitä pulssisuhteena. Pulssisuhteen muuttaminen asteiksi on ohjelmallisesti helppoa.

SMT160 on TTL-tasoinen, eli sen voi yhdistää suoraan mikrokontrollerille. Lämpötilatieto saadaan anturilta laskemalla *t* kaavasta

missä *D.C* on pulssisuhde ja *t* on lämpötila. Seuraavassa laskuesimerkkinä tilanne, jossa pulssisuhteeksi on mitattu 32.2 %. Tällöin lämpötilaksi saadaan

Anturin lämpöalue on -45 °C… +130 °C. Tarkkuus tällä alueella on ± 2 astetta. Anturin resoluutio on <0.005 °C, eli anturi pystyy havaitsemaan pienetkin lämpötilavaihtelut. Lopullinen tarkkuus tietysti riippuu siitä, kuinka hyvin on pulssisuhteen mittaus on toteutettu mikrokontrollerilla. /3/

3.2 Kehitysalusta

Kehitysalustana käytetään Olimexin AVR-MT-128-kehitysalustaa. Kehitysalusta käyttää Atmelin ATmega 128 –mikrokontrolleria, joka toimii 16 MHz taajuudella. Ohjelmointiliittiminä toimivat JTAG ja STKxxx-yhteensopiva ICSP 5x2.

Kehitysalustan tärkeimmät ominaisuudet ovat

- 2x16-rivinen LCD-näyttö
- RS232-liitäntä
- TTL-tasoinen asynkroninen sarjaliitäntä
- Ulkoinen kellokide
- Viisi painonappia
- Piikkirimaliittimet käyttämättömille porteille

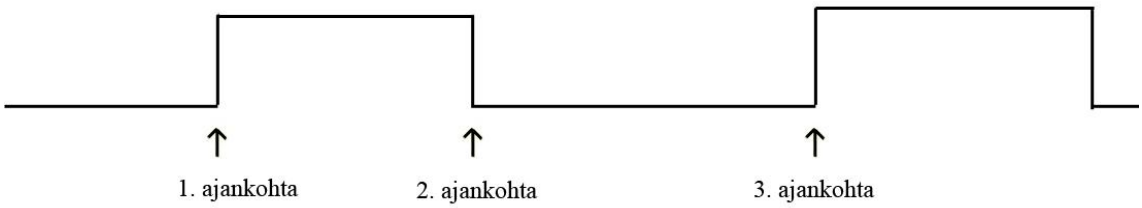
Suurin syy, minkä takia tässä työssä päädyttiin käyttämään tätä kehitysalustaa, oli, että on JTAG-debuggerimahdollisuus.

3.3 Mikrokontrollerin ohjelmisto

Seuraavissa aliluvuissa käsitellään mikrokontrolleriin ladattavan ohjelmiston eri osa-alueita ja testaamista.

3.3.1 Pulssisuhteen mittaaminen

Mikrokontrollerille sovellusta tehtäessä tarvitsee ensin miettiä, kuinka pulssisuhteenmittaus toteutetaan. Pulssisuhteen laskemiseksi tarvitsee tallentaa kuvan 4 mukaisista mukaiset ajankohdat.



Kuva 4. Pulssisuhdemittauksien ajankohdat

Pulssisuhde saadaan kaavasta

$$D.C = \frac{T_{high}}{T}$$

Kaavassa T on jaksonaika, T_{high} on pulssin aktiivisenaoloaika ja D.C on pulssisuhde. Pulssisuhde saadaan vähentämällä toinen ajankohta ensimmäisestä ajankohdasta, jolloin saadaan $T_{high.}$ Jaksonaika saadaan vähentämällä kolmannesta ajankohdasta ensimmäisestä ajankohta, jolloin voidaan laskea pulssisuhde pääohjelmassa. /5/

Pulssista tallennetaan muistiin laskevien ja nousevien reunojen ajankohdat Timer1_capt_vect-keskeytysohjelmalla:

```
ISR(TIMER1_CAPT_vect)
{
            b=ICR1;
            if(suunta==0) {leveys=((unsigned long)ovf<<16)+(unsigned long)b;
            TCCR1B=0b01000010;  suunta=1; TIFR^=0b00100000; hightime=(leveys-
            eka_r)*8*0.0000000625;  }
            else {toka_r=eka_r; eka_r=((unsigned long)ovf<<16) +(unsigned
            long)b ;TCCR1B=0b00000010; suunta=0;TIFR^=0b00100000;}
            kesklippu++;
}
```

Keskeytysohjelma käynnistyy joko nousevalla tai laskevalla reunalla. Mikäli suuntana on laskeva reuna, kopioidaan b-muuttujan sisältö *leveys*-muuttujaan, muutetaan *TCCR1B*-rekisterissä seuraava keskeytys tapahtumaan nousevasta reunasta ja tyhjennetään *TIFR*-rekisterissä *input capture*-lippu. *Input capture*-lippu on tyhjennettävä, koska tallennettavan reunan muutos aiheuttaa *input capture:n* luulemaan, että uusi keskeytys olisi jo tapahtunut. Lopuksi lasketaan myös pulssin aktiivisenaoloaika *hightime*-muuttujaan. Tulos kerrotaan kahdeksalla, koska Timer-Counter1 on asetettu toimimaan kellotaajuuden kahdeksasosanopeudella - eli oikea aika on kahdeksan kertaa isompi kuin *input capture:ssa* luetut arvot. Tulos kerrotaan vielä 62.5 nanosekunnilla, koska tällöin saadaan oikeata aikaa vastaava tulos. Tämä kertolasku suoritetaan, koska mikrokontrollerissa yksi askel vastaa kontrollerissa käytetyn kellon jaksonaikaa ja kellotaajuuden ollessa 16 MHz jaksonaika on 62.5 ns. Keskeytyksen tapahduttua nousevalla reunalla tallennetaan edellisen keskeytyksen ajankohta *toka_r*-muuttujaan ja keskeytyshetken ajankohta *eka_r*-muuttujaan. Lopuksi vaihdetaan keskeytys tapahtumaan laskevalta reunalta ja nollataan *input capture*-keskeytyslippu.

3.3.2 LCD-näyttö

LCD-näyttö alustetaan käyttöön seuraavasti:

```
void lcdinit()
{
            PORTC=0b00110000;
            e_puls();
            _delay_ms(6);
            PORTC=0b00110000;
            e_puls();
            _delay_ms(6);
            PORTC=0b00110000;
            e_puls();
            _delay_ms(6);
            PORTC=0b00100000;
            e_puls();
            sendcmd(0x28); _delay_ms(2);
            sendcmd(0x08); _delay_ms(2);
            sendcmd(0x01); _delay_ms(3);
            sendcmd(0x06); _delay_ms(2);
            sendcmd(0x0C); _delay_ms(2);
```

```
            sendcmd(0x80); _delay_ms(1);
}
```

Olimexin LCD-näyttö alustetaan alussa Hitachin 44780 normin mukaisesti luvuilla 0b00110000. Tämän jälkeen voidaan alustaa näyttö käyttökuntoon asettamalla esimerkiksi näyttö käyttämään 4-bittisiä komentoja, tyhjentämällä näyttö ja kykemällä näyttö päälle. /4/

LCD-näytölle kirjoittamista varten sille tehtiin oma funktio, jossa lähetetään kirjain kerrallaan näytölle.

```
void lcdkirj()
{
            int a
            sendcmd(128); _delay_ms(2);
            sendcmd(2); _delay_ms(2);
            for(a=0;a<=31;a++)
            {
                        sendchar(lcdbuf[a]); _delay_us(40);
                        if(a==15){ sendcmd(0b11000000); _delay_us(40);}
            }
}
```

Lähetys tapahtuu for-silmukassa, jossa globaalin *lcdbuf[]*-taulukon arvot lähetetään näytölle käyttäen *sendchar*-funktiota. Taulukon 0-paikka vastaa näytöllä ensimmäistä ja 31 viimeistä paikkaa. Kun for-silmukka pääsee LCD-näytön ensimmäisen rivin loppuun eli paikkaan 15, lähetetään LCD-näytön kontrollerille komento, joka vaihtaa LCD:n käyttämään toista riviä. Aina, kun *lcdkirj()*-funktiota kutsutaan, asetetaan aluksi LCD-näyttö käyttämään ensimmäistä riviä ja palautetaan kursori rivin alkuun.

3.3.3 Sarjaportti

Sarjaportti asetetaan käyttämään 9600 bit/s -tiedonsiirtonopeutta asettamalla *UBRR1*-rekisteriin arvo 103.

```
unsigned int BaudRate =103;
UBRR1H = (unsigned char) (BaudRate>>8);
UBRR1L = (unsigned char) BaudRate;
```

UCSR1B-rekisterissä sallitaan sarjaportin lähetys, ja *UCSR1C*-rekisterissä asetetaan yhden merkin pituudeksi 8-bittiä. Tämä tapahtuu seuraavasti:

```
UCSR1B = UCSR1B | 0b00001000;
UCSR1C = UCSR1C | 0b10000110;
```

Tiedonsiirto mikrokontrollerilta PC:lle tapahtuu Timer1_compa_vect-keskeytyspalvelussa.

```
ISR(TIMER1_COMPA_vect)
{
            uint8_t merkki;
            OCR1A +=20000;
            scount++;
            if(scount==100)
            {
                        if(lcdbuf[7]!='0'){ UDR1=lcdbuf[7]; while(!(UCSR1A &
                        (1<<UDRE1)));}
                        if(lcdbuf[8]!='0'){ UDR1=lcdbuf[8]; while(!(UCSR1A &
                        (1<<UDRE1)));}
                        UDR1=lcdbuf[9]; while(!(UCSR1A & (1<<UDRE1)));
                        UDR1='.'; while(!(UCSR1A & (1<<UDRE1)));
                        UDR1=lcdbuf[11];
                        scount==0;
            }
}
```

Ajastinkeskeytyspalveluun mennään 10 millisekunnin välein, mutta itse datan lähetys tapahtuu sadan keskeytyksen eli sekunnin välein. Lähetysvaiheessa tarkistetaan, onko lämpötila satoja, kymmeniä vai alle kymmeniä asteita. Tarkistus tehdään, jottei lähetettäisi turhia nollia sarjaväylään. While-silmukkaa käytetään varmistuskeinona, ettei lähetetä dataa ennen kuin lähetysrekisteri on tyhjä.

3.3.4 Pääohjelmasilmukka

Ohjelman tärkeimmät toimenpiteet ja laskennat tehdään pääohjelmasilmukassa, koska keskeytysohjelmat pitää olla mahdollisimman lyhyitä. Tämä siksi, että mahdolliset tulevat keskeytykset eivät viivästyisi. Jaksonaika lasketaan tuoreimman ja edellisen nousevan reunan vähennyslaskuna. Sen jälkeisillä kertolaskuilla jaksonaika muunnetaan sekunneiksi:

```
jakso=(eka_r-toka_r)*8*0.0000000625;
```

Lämpötila lasketaan käyttämällä luvussa 3.1 esiteltyä kaavaa. Kertomalla tulos kymmenellä vältetään float-muuttujan käyttämistä ja saadaan silti tulos yhden desimaalin tarkkuudella.

```
temp=((cycle-0.320)/0.00470)*10;
```

Pääohjelmasilmukan loput toimenpiteet toteutetaan seuraavasti:

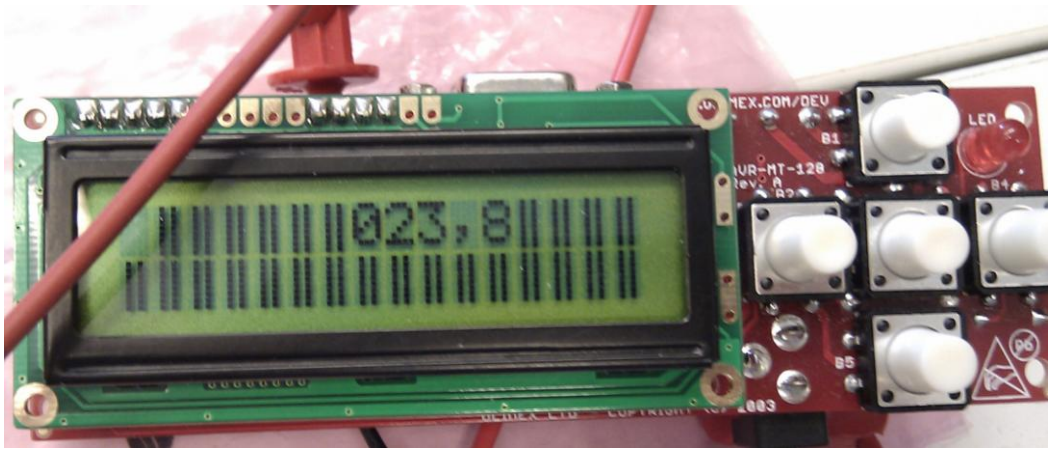
```
if(!(temp-temp2>=50) || !(temp-temp2<=-50 )|| tosi==false)
{
            maara++;
            temp3+=temp;
            if(maara>=100)
            {
            ascii((temp3/100)/10,9,3); ascii((temp3/100)%10,11,1); maa-
            ra=0;temp2=temp; tosi=true;
            lcdkirj(); temp3=0;
            }
}
```

Ensimmäisessä ehdossa tarkistetaan, onko lämpötila muuttunut yli viisi astetta verrattuna edelliskertaan kun lämpötilasta on laskettu sadan otannan keskiarvo. Mikäli lämpötila on muuttunut yli viisi astetta, tätä mittausta ei oteta huomioon. Mikäli käynnissä on ensimmäinen keskiarvojen laskenta, niin edellistä vertailukohtaa ei ole ja if-silmukasta päästään läpi niin pitkään, kunnes saadaan vertailukohta. Numeraalinen luku muunnetaan ASCII-muotoiseksi, kun 100 lämpötilatietoa saadaan summattua *temp3*-muuttujaan. Ensimmäisellä kerralla *ascii*-funktioon lähetetään *temp3* jaettuna 100:lla, jonka jälkeen luku jaetaan vielä kymmenellä, jolloin ASCII-muotoon muunnetaan vain kokonaisluvut lämpötilasta. Toisella kerralla ascii-funktioon lähetetään myös *temp3* jaettuna 100:lla, mutta tästä otetaankin jakojäännös kymmenellä, jolloin ASCII-muotoon muunnetaan vain desimaaliluvut, joita tässä tapauksessa on vain yksi. *Ascii*-funktiossa saadut arvot tallennetaan *lcdbuf*-taulukkoon.

3.3.5 Ohjelmiston testaaminen

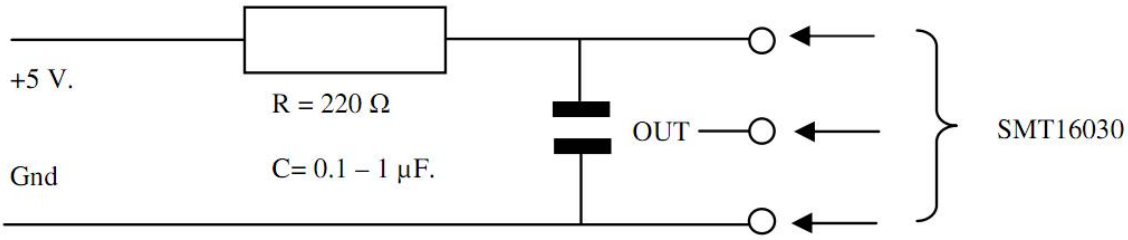
Ohjelmisto testattiin sen kehittämisen rinnalla. Käytännössä tämä tarkoittaa, että lisättäessä sovellukseen jokin ominaisuus tai muokattaessa jotain kohtaa, muokattua tai lisättyä kohtaa testattiin, toimiiko se kuin on tarkoitus.

Kuvassa 5 näytetään, miltä lämpötila näyttää näytöllä ohjelman lopullisessa versiossa.



Kuva 5. Lämpötila näytöllä

Eräässä testausvaiheessa näyttö ei näyttänyt enää mitään järkevää, sillä jossain vaiheessa lämpöanturi oli rikkoutunut. Uusi anturi kytkettiin kuvan 6 mukaiseen alipäästösuodin kytkentään turvaamaan anturin toimintaa.

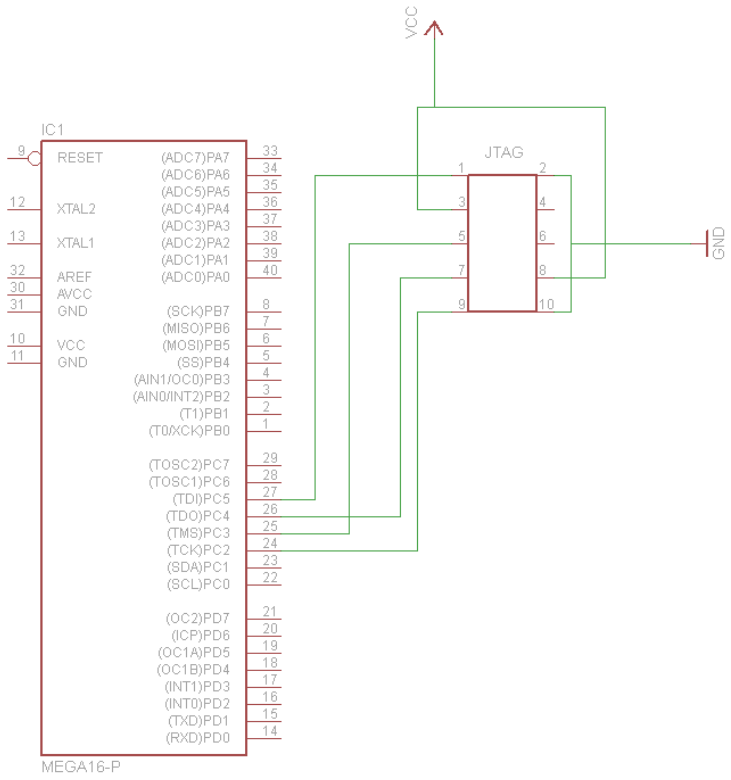


Kuva 6. Lämpöanturin alipäästösuodin /3/

Suotimen tarkoitus on estää anturin sisäisen oskillaattorin synkronoituminen. Vastus toimii myös virran rajoittajana ja rajoittaa maksimivirran 25 milliampeeriin.

4 LAITTEEN SUUNNITTELU JA VALMISTUS

Laite suunniteltiin ensin koekytkentäalustalle ja mikrokontrollerina käytetään ATmega 16:ta. Koekytkentäalustalle tehtiin myös kuvan 7 mukainen JTAG-kytkentä, jolla voidaan ohjelmoida kontrolleria.



Kuva 7. JTAG-kytkentä

LCD-näyttö jätettiin pois, sillä lopullisessa laitteessa lämpötilatieto luetaan joko tietokoneen näytöltä tai matkapuhelimella.

Ohjelman modifiointivaiheessa ATmega 128:lta ATmega 16:lle tuli ilmi JTAG:n erikoinen ongelma. JTAG ei pääse käsiksi oleellisiin sarjaportin rekistereihin *UCSRC* ja *UBRRH*. Lopullinen varmistus asialle tuli Atmelin tukisivulta, joten koululta lainattiin STK500-kehitysalusta ja sillä ohjelma toimii hyvin. /6/

Ohjelman ollessa valmiina reikälevylle tehtiin koekytkentäalustan mukainen kytkentä. Lämpömittari lähetti lämpötiedon oikein sarjaväylän avulla, joten reikälevylle tehtiin toimiva kytkentä. Laitteen virrankulutus on noin 40 milliampeeria, josta mikrokontrolleri vie suurimman osan.

5 LÄMPÖTILAN LUENTAOHJELMISTO

Lämpömittarin tuottama mittaustieto voidaan lukea tietokoneella erillisen ohjelman avulla. Tämä luku käsittelee Qt-ympäristössä tehdyn ohjelman keskeisiä toimintoja.

5.1 Pääohjelma

Pääohjelman alussa luetaan tekstitiedostosta, mitkä ovat ne kaksi sarjaporttia, joita ohjelmisto käyttää. Ensimmäiseltä riviltä luetaan Bluetoothin ja toiselta riviltä sarjaportin käyttämä COM-portti. Bluetooth käyttää niin sanottua virtuaalista sarjaporttia. /7/

```
QFile file("config.txt");
file.open(QIODevice::ReadOnly);
QTextStream in(&file);
QString line1,line2;
line1=in.readLine();
line2=in.readLine();
file.close();
```

Text-luokalle tehdään instanssit *blue* ja *serial*, joissa viedään tiedot, mitä COM-portteja ne käyttävät, ja tieto, onko instanssi Bluetoothin vai sarjaportin käytössä.

```
text blue("blue",line1),serial("serial",line2);
```

Pääohjelmassa myös yhdistetään käytetyt signaalit ja slotit, jonka jälkeen aloitetaan säikeitten suorittaminen. /8, s. 355/

```
QObject::connect(&serial,SIGNAL(pass(const QString&,const QString&))
,&blue, SLOT(bluesend(const QString&, const QString& )));
QObject::connect(&serial,SIGNAL(empty(const QString&))
,&blue, SLOT(empty2(const QString& )));
QObject::connect(&serial,SIGNAL(wtext(const QString&))
,&w, SLOT(sread(const QString& )));
blue.start();
serial.start();
```

Ohjelma on pakko tehdä säikeistettynä, eli rinnakkain ajettavana, koska muuten ei pystyisi helposti samaan aikaan sekä lukemaan että kirjoittamaan eri sarjaväyliin. Rinnakkaisajolla annetaan myös muille mahdollisille prosesseille suoritusaikaa.

5.2 Sarjaportin käyttäminen ja rinnakkaisajaminen

Text-luokan konstruktorissa kopioidaan *valinta*-muuttujaan tieto, onko tämä instanssi sarjaportin vai Bluetoothin käyttöä varten. Ennen varsinaista sarjaportin alustusta, on käytettävän COM-portin tieto muutettava Qstringistä char-muuttujaksi, koska sarjaportin alustuksessa saa käyttää vain char-muuttujia.

```
const char *port;
QByteArray ba=temp.toLatin1();
port=ba.data();
```

Sarjaportin käyttöä varten on käytettävä windows.h-kirjastoa. Sarjaportin avaamisen koodi käyttöä varten näyttää seuraavalta:

```
hSerial = CreateFileA(port,GENERIC_READ | GENERIC_WRITE,0,0,
OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
if(hSerial==INVALID_HANDLE_VALUE){
if(GetLastError()==ERROR_FILE_NOT_FOUND){
cout<<"porttia ei ole \n";}
cout<<"jotain ihan muuta"<<endl;
}
```

Portti asetetaan lukevaksi ja kirjoittavaksi. Parametrin *OPEN_EXISTING* avulla kerrotaan, että avataan vain olemassa olevia sarjaportteja, ja mikäli annettua sarjaporttia ei löydy, alustus epäonnistuu.

Sarjaportin parametrit asetetaan seuraavasti:

```
serialparameters.BaudRate=CBR_9600;
serialparameters.ByteSize=8;
serialparameters.StopBits=ONESTOPBIT;
serialparameters.Parity=NOPARITY;
```

Sarjaportin parametriksi laitetaan 9600 bit/s, 8 bit/merkki, yksi loppubitti, eikä pariteettiä.

Sarjaporttiin on myös asetettava intervalliajat tarkoittaen aikaa, kuinka pitkään sarjaväylää luetaan, ennen kuin mennään ohjelmassa eteenpäin, koska muuten ohjelma jää odottamaan dataa sarjaväylältä. Jos dataa ei ole, niin ohjelma käytännössä jumittuu siihen paikkaan.

```
timeouts.ReadIntervalTimeout=10;
timeouts.ReadTotalTimeoutConstant=10;
timeouts.ReadTotalTimeoutMultiplier=10;
```

ReadIntervalTimeout määrittelee, kuinka pitkä tauko pidetään vastaanotettujen kirjainten välissä. *ReadTotalTimeoutConstant* määrittelee, kuinka pitkä tauko pidetään viimei-

sen vastaan otetun kirjaimen jälkeen. Muuttuja *ReadTotalTimeoutMultiplier* määrittelee kertoimen viimeisen kirjaimen jälkeiselle odotusajalle. /9/

Sarjaportin lukeminen ja kirjoittaminen tapahtuu *run*-funktiossa. Mikäli *run*-funktiota käytetään sarjaportin lukemiseen tarkoitetussa instanssissa, on koodi seuraava:

```
if(valinta=="serial")
    {
    while(!m_stop)
    {
        memset(buffer, 0, 50);
        DWORD bytesread;
        ReadFile(hSerial,buffer,50,&bytesread,NULL);
        if(bytesread>0)
        {
            m_text=buffer;
            emit wtext(m_text);
            emit pass(m_text,"ok");
        }
        else emit {m_text="fail"; empty(m_text);}
        sleep(1);
    }
```

While-silmukkaa suoritetaan, kunnes ohjelman ajo lopetetaan, sillä ohjelmassa ei pysäytetä säikeitä koskaan. *Memset*-funktiolla tyhjennetään char-*buffer*-taulukko aina mentäessä silmukan alkuun. Mikäli sarjaväylässä oli dataa, saatu data lähetetään näytölle kirjoittamista varten olevalle funktiolle ja Bluetoothin kirjoittamista varten olevalle säikeelle. Data muunnetaan Qstring-muotoon, koska Qstring on helpompi lähettää eteenpäin kuin char-taulukko. Mikäli dataa ei ole linjalla, lähetetään Bluetoothin säikeelle tieto, ettei turhaan lähetä uutta dataa matkapuhelimelle. Lopussa säie asetetaan nukkumaan sekunnin ajaksi, jonka jälkeen palataan silmukan alkuun. /8, s. 358/

Bluetoothin käyttöön tarkoitetussa instanssissa koodi on seuraava:

```
else if(valinta=="blue")
    {
        while(!m_stop)
        {
            if(m_text=="ok")
            {
                byteswrite=5;
                const char *str;
                QByteArray ba=to_phone.toLatin1();
                str=ba.data();
                WriteFile(hSerial,str,5,&byteswrite,NULL);
            }
         sleep(1);
        }
    }
```

Bluetoothin käyttämään sarjaporttiin kirjoitetaan vain, jos sarjaportti on saanut dataa lämpömittarilta. Kirjoittaessa sarjaporttiin tarvitsee Qstring-muodossa saatu data muuttaa takaisin char-muotoon, koska kirjoitus-funktiossa voi vain käyttää char-muuttujia.

5.3 Pääikkunan funktiot

Pääikkunan konstruktorissa alustetaan *eka*-muuttuja todeksi, jotta voitaisiin tallentaa heti ohjelman käynnistyttyä senhetkinen lämpötila ja kellonaika.

Lämpötila tiedon ja tiedon toimittaminen näytölle tapahtuu *sread*-funktiossa seuraavasti:

```
void MainWindow::sread(const QString& number)
{
    ui->lcdNumber->display(number);
    aika++;
    if(aika==600 || eka==true)
    {
        eka=false;
        aika=0;
        QDate date=QDate::currentDate();
        QString datestring=date.toString("dd_MM_yyyy");
        QTime time=QTime::currentTime();
        QString timestring=time.toString("hh.mm");
        QFile file(datestring+ ".txt");
        file.open(QIODevice::WriteOnly | QIODevice::Append | QIODevice::Text);
        QTextStream out(&file);
        out <<number<<'\t'<<timestring<<'\n';
        file.close();
    }
}
```

Sread-funktioon tullaan aina sarjaportin saatua dataa lämpöanturilta. Saatu data päivitetään aina näytölle. Mikäli kyseessä on ensimmäinen kerta kun funktiota käytetään tai aikaa on kulunut noin 29 min edellisestä datan tallennuksesta, tallennetaan senhetkinen lämpötila ja aika tiedostoon. Mikäli tiedostoa ei ole, se luodaan ja tiedoston nimeksi tulee ”päivä_kuukausi_vuosi.txt” ja mikäli päivä on vaihtunut edellisen ja seuraavan tallennuksen välillä, data tallennetaan uuteen tiedostoon.

5.4 Sovelluksen käyttöliittymä ja testaaminen

Sovellusta testattiin jokaisessa kehityksen vaiheessa. Ohjelmaan lisättäessä jokin ominaisuus, niin lisätyn ominaisuuden koodia kehitettiin ja testattiin niin pitkään kunnes se toimii halutulla tavalla. Sovelluksen lopullinen testaaminen tapahtui pitämällä tietokone päällä yöllä ja päivällä, jolloin pystyy katsomaan onko sovellus tallentanut lämpötilat ja

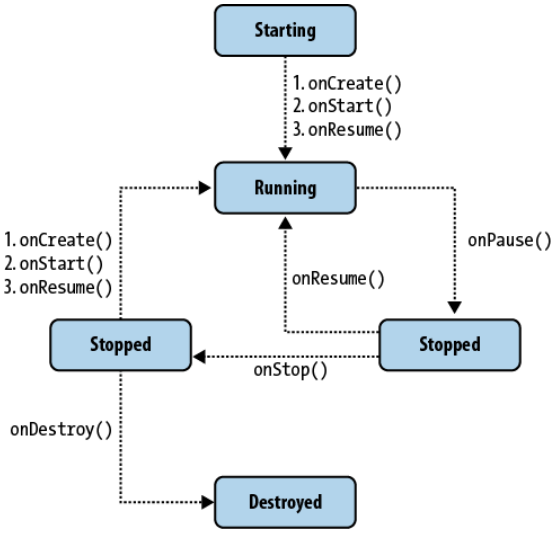
kellonajat. Mittaustietojen näyttäessä oikealta, voi todeta että ohjelmaa toimii kuten pitääkin. Ohjelman lopullinen käyttöliittymä kuvan 8 mukainen.

Kuva 8. Lopullinen käyttöliittymä

Ohjelman aikaisemmissa käyttöliittymä versioissa kerrottiin myös senhetkinen päivämäärä, mutta se jätettiin lopullisesta versiosta pois.

6 ANDROID-SOVELLUS

Android-sovelluksen elinkaari esitetään kuvassa 9.



Kuva 9. Aktiviteetin elinkaari /10, s.29/

Ohjelma on aloitustilassa, mikäli aktiviteettiä ei löydy muistista. Aloituksen jälkeen tehdään kaikki tarvittavat toimenpiteet ja käynnistetään itse ohjelman ajaminen. Ajamistilassa ollaan niin pitkään kuin näyttö on interaktiivisessa tilassa, eli ohjelmaa käytetään. Aktiviteetin ollessa näkyvillä, muttei käytettävissä, ollaan pause-tilassa ja tällä tilalla on suuri prioriteetti saada resursseja käyttöön. Käyttäjän poistuttua ohjelmasta aktiviteetti jää vielä muistiin ja ollaan stop-tilassa. Stop-tilan jälkeen aktiviteetille jää kaksi vaihtoehtoa, joko aktiviteetti poistetaan muistista tai aktiviteetti palautetaan aktiiviseksi. /10, s.29/

6.1 Sovelluksen toiminnot

Aktiviteetin alussa alustetaan kaikki tarpeelliset muuttujat seuraavasti:

```
TextView textView1;
private static final String TAG ="Hammas";
private static final boolean D=true;
private BluetoothAdapter mBluetoothAdapter=null;
private BluetoothSocket btsocket=null;
private InputStream instream=null;
private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
private static String address="xx:xx:xx:xx:xx:xx";
boolean isrunning = false;
```

MY_UUID-muuttujassa luodaan Bluetooth-yhteydelle uniikki ID-tunnus, jonka avulla se tunnistautuu. *Adress*-muuttujassa puolestaan asetetaan yhdistettävän laitteen MAC-osoite.

Bluetoothin tila tarkistetaan *onCreate*-vaiheessa. Mikäli Bluetooth ei ole käytettävissä, sovellus lopetetaan ja annetaan ilmoitus, ettei Bluetooth ole päällä. Bluetoothin tilan tarkistaminen on toteutettu seuraavasti:

```
BluetoothAdapter=BluetoothAdapter.getDefaultAdapter();
        if(!mBluetoothAdapter.isEnabled()){
           Toast.makeText(this, "ENABLE BLUETOOTH AND COME BACK
AGAIN",Toast.LENGTH_LONG).show();
           finish();
        }
```

Bluetooth-yhteyttä valmistellaan *onStart*-vaiheessa. Yhteyden muodostamista varten tarvitsee saada haltuun *BluetoothDevice*-objekti, joka vastaa etälaitetta:

```
BluetoothDevice device=mBluetoothAdapter.getRemoteDevice(address);
```

Tämän jälkeen voidaan luoda RFCOMM-socket käyttäen omaa ID-tunnusta.

```
try {
            btsocket = device.createRfcommSocketToServiceRecord(MY_UUID);

            } catch (IOException e){
                        Log.e(TAG,"++ EPIC FAIL!! ++");
            }
            Log.d(TAG,"++ Socket connected ++");
    }
```

Varsinainen yhteyden muodostus tapahtuu ohjelman ollessa *onResume*-kohdassa. Yhteys muodostetaan *btsocket.connect()*-komennolla, ja mikäli yhteyden muodostus ei onnistu, debuggeriin lähetetään tieto, että yhteydenmuodostus epäonnistui ja samalla luotu socket suljetaan. Sisääntulevan datavirran talteenotto tapahtuu seuraavalla käskyllä:

```
try{
            instream=btsocket.getInputStream();

            } catch (IOException e) { }
            if (D) Log.e(TAG,"++ lol ++");
```

Tämän jälkeen voidaan aloittaa datavirran lukemiseen tarkoitetun säikeen ajaminen. Säikeen ollessa käynnissä dataa luetaan seuraavalla tavalla:

```
try{
    bytes=instream.read(buffer);
    Message msg=handler.obtainMessage(1,bytes,-1,buffer);
    handler.sendMessage(msg);
```

```
if (D) Log.e(TAG,"++ luettu ++");

} catch (IOException e) {
break;
}
if (D) Log.e(TAG,"++ jep ++");
Thread.sleep(100);
}

} catch (Throwable t){ }
```

Data luetaan ensin *byte*-taulukkoon, jonka jälkeen saatu data lähetetään handlerille. Luennan jälkeen säie nukkuu 100 ms. /11/

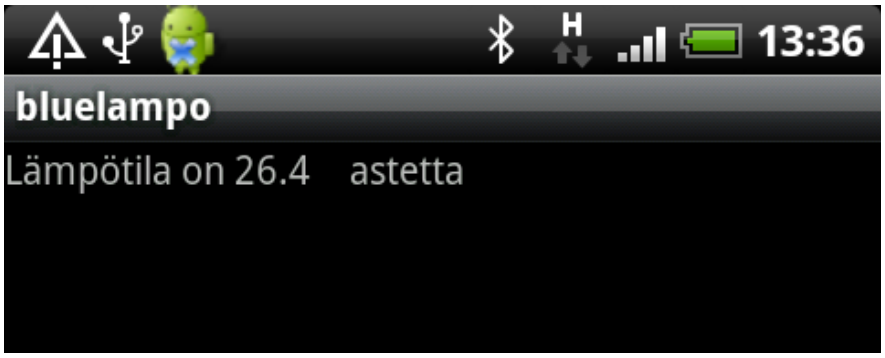
Data luetaan handlerissä, jonka Bluetoothin luentasäie on sille lähettänyt ja saatu data näytetään matkapuhelimen näytöllä.

```
@Override
public void handleMessage(Message msg){
byte[] readbuf=(byte[]) msg.obj;
String readmessage= new String(readbuf,0,msg.arg1);
textView1.setText("Lämpötila on "+ readmessage + "astetta");
if (D) Log.e(TAG,"++ on read message ++");
}};
```

Ennen ohjelman ajoa tarvitsee *AndroidManifest.xml*-tiedostoon kirjoittaa käyttöoikeudet Bluetoothille, koska ilman käyttöoikeuksia ohjelma ei toimi.

6.2 Android-sovelluksen testaus ja käyttöliittymä

Android-sovellus testattiin melko lailla samalla tavalla kuin muitakin sovelluksia, eli jokaista eri osa-aluetta kehitettiin ja testattiin, kunnes saatiin toimivaksi. lopullista versiota testattiin samaan aikaan tietokoneen sovelluksen kanssa, ja samalla oikeastaan testattiin myös lähettääkö tietokone lämpötietoa matkapuhelimelle. Matkapuhelin saa datan hyvin vastaan ja ilmoittaa hyvin pienellä viiveellä lämpötila muutokset verrattuna tietokoneen näyttämään lämpötilaan. Kuten kuvasta 10 käy ilmi, sovelluksen lopullinen käyttöliittymä ei ole mitenkään monimutkainen, mutta siinä kerrotaan kaikki oleellinen eli sen hetkinen lämpötila.



Kuva 10. Android-sovelluksen käyttöliittymä

7 YHTEENVETO

Työn tavoitteena oli rakentaa lämpötilamittari, jota voidaan lukea matkapuhelimelta ja tietokoneelta. Työn tavoitteessa onnistuttiin, vaikkakin työtä pystyisi jatkokehittämään vielä melko paljon.

Työn helpoin osuus oli mikrokontrolleriosan kehittäminen, koska mikrokontrollerin ohjelmoinnista oli enemmän kokemusta, kuin Qt:lla tai Java:lla ohjelmoimisesta. AVR studiossa kannattaa aluksi ottaa kääntäjästä optimointi pois, sillä optimoinnin takia ohjelmisto ei aluksi toiminut halutulla tavalla. Heti, kun optimointi otettiin pois käytöstä, ohjelma alkoi toimia.

Qt:lla ohjelmointi on melko helppoa, sillä se on käytännössä vain C++-ohjelmointia, jossa käytetään erillisiä kirjastoja. Qt:lla sovelluksen kääntäminen suoritettavaksi tiedostoksi on periaatteessa helppoa, sillä testattaessa ohjelmaa Qt Creator tekee aina suoritettavan tiedoston valittuun kansioon, mutta tiedostoa ei voida ajaa, ennen kuin samasta kansiosta löytyy kaikki vaadittavat dll-tiedostot. Tiedostojen saamiseksi kansioon saattaa olla parempiakin keinoja, mutta työssä käytettiin manuaalista keinoa, eli dll-tiedostot kopioitiin yksitellen Qt Creatorin kansiosta ajettavan ohjelman kansioon, jolloin voidaan ajaa ohjelmaa ilman Qt Creatoria.

Android-sovelluksen tekeminen oli kaikista aikaa vievin osuus, koska Java-ohjelmointi piti opetella perusteista asti. Java-ohjelmointi alkoi sujua pienen harjoittelun jälkeen ja aikaan saatiin ihan toimiva sovellus.

8 LÄHTEET

/1/ Nokia Corporation. Qt Creator IDE and tools. http://qt.nokia.com/products/developer-tools (10.5.2011)

/2/ Eclipse Foundation. Eclipse wiki. http://wiki.eclipse.org/Main_Page (10.5.2011)

/3/ Smartec. SMT160 Data Sheet. http://www.smartec.nl/pdf/DSSMT16030.PDF (11.5.2011)

/4/ Olimex Ltd. AVR-MT-128 Development Board. http://www.olimex.com/dev/avr-mt128.html (11.5.2011)

/5/ Doctronics. 555-Timer. http://www.doctronics.co.uk/555.htm (12.5.2011)

/6/ Atmel Corporation. JTAGICE mkII Special Considerations. http://support.atmel.no/knowledgebase/avrstudiohelp/mergedProjects/JTAGICEmkII/mkII/Html/JTAGICE_mkII_Special_Considerations.htm (14.5.2011)

/7/ Nokia Corporation. How to write data to a file in Qt. http://wiki.forum.nokia.com/index.php/How_to_write_data_to_a_file_in_Qt (15.5.2011)

/8/ J. Thelin: Foundations of Qt Development. Apress, 2007. 528 sivua.

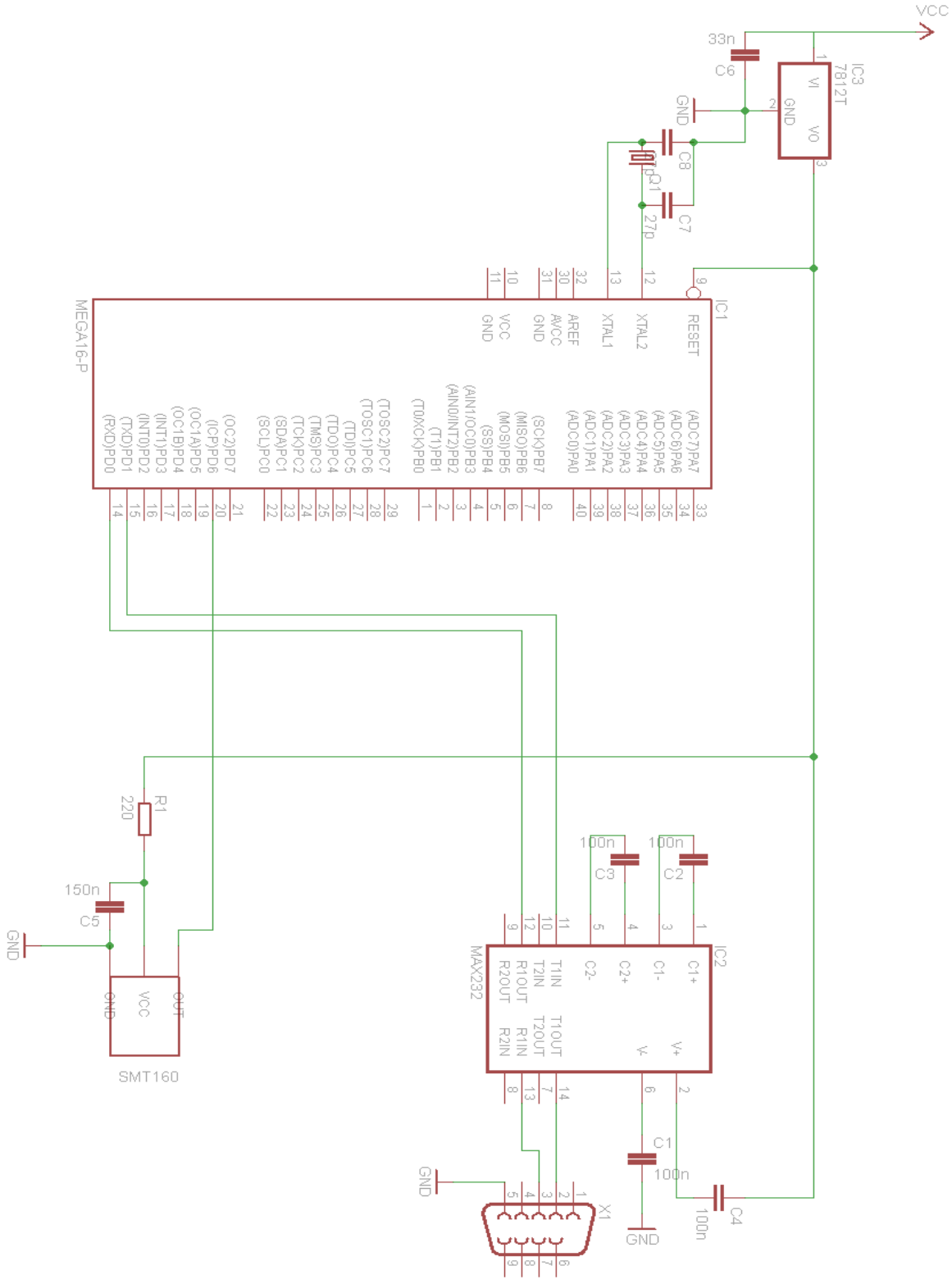
/9/ Robertson Bayer. Windows Serial Port Programming. http://www.robbayer.com/files/serial-win.pdf (16.5.2011)

/10/ M. Gargenta: Learning Android. O'reilly, 2011. 245 sivua.

/11/ Threading with Handlers. http://mobileorchard.com/android-app-developmentthreading-part-1-handlers/ (18.5.2011)

9 LIITTEET

Liite 1. Laitteen piirikaavio



Liite 2. Laitteen lähdekoodi

```
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include "delay.h"
#include <stdbool.h>


volatile unsigned char data;
volatile unsigned char lcdbuf[32]={" "};
volatile unsigned char kesklippu=0;
volatile unsigned  int maara=0;
volatile unsigned long  freq=0;
volatile unsigned long eka_r;
volatile unsigned long toka_r;
volatile unsigned long leveys;
volatile unsigned long leveys2;
volatile float jakso;
volatile float cycle;
volatile unsigned int temp;
volatile unsigned int temp2;
volatile unsigned long temp3;
volatile unsigned char suunta=1;
volatile float hightime;
volatile unsigned char scount=0;


volatile unsigned int b=0;
volatile unsigned int ovf=0;
const long dek[]={1,10,100,1000,10000,100000,1000000,10000000,100000000};
void lcdinit();
void sendchar(unsigned char );
void sendcmd(unsigned char);
void lcdkirj();
void e_puls();
void ascii(long arvo,unsigned int paikka, unsigned int lkm);
void uart_init();


ISR(TIMER1_COMPA_vect)
{
                        OCR1A +=20000;
                        scount++;
                        if(scount==100)
```

```
		{
				if(lcdbuf[7]!='0'){ UDR=lcdbuf[7]; while(!(UCSRA & (1<<UDRE)));}
				if(lcdbuf[8]!='0'){ UDR=lcdbuf[8]; while(!(UCSRA & (1<<UDRE)));}
				UDR=lcdbuf[9]; while(!(UCSRA & (1<<UDRE)));
				UDR='.'; while(!(UCSRA & (1<<UDRE)));
				UDR=lcdbuf[11];
				scount==0;

		}

}

ISR(TIMER1_OVF_vect)
{
		ovf +=1;

}

ISR(TIMER1_CAPT_vect)
{
		b=ICR1;
		if(suunta==0) {leveys=((unsigned long)ovf<<16)+(unsigned long)b;leveys2=eka_r;
		TCCR1B=0b01000010; suunta=1; TIFR^=0b00100000;
		hightime=(leveys-eka_r)*8*0.0000000625;  }
		else {toka_r=eka_r; eka_r=((unsigned long)ovf<<16) +(unsigned long)b ;TCCR1B=0b00000010;
		suunta=0;TIFR^=0b00100000;
		kesklippu++;
}

void uart_init()
{
		unsigned int BaudRate =103;
		UBRRH = (unsigned char) (BaudRate>>8);
		UBRRL = (unsigned char) BaudRate;

		UCSRB = UCSRB | 0b10011000;
		UCSRC = UCSRC | 0b10000110;
}


int main()
{

		PORTA=0;
		DDRA=0xff;
```

```
DDRC=0xff;
PORTC=0;
DDRD = 0b00001000;

TCCR1A=0b01000000;
TCCR1B=0b11000010;
TIMSK=0b00110100;

OCR1A=0x20;
lcdinit();
lcdbuf[10]=',';
bool tosi=false;
uart_init();

sei();
while(1)
{
if(kesklippu>=1 &&ovf !=1)
{
ovf=0;

jakso=(eka_r-toka_r)*8*0.0000000625;
freq=1/jakso;
cycle=(((hightime))/(jakso));
temp=((cycle-0.320)/0.00470)*10;

if(!(temp-temp2>=10) || !(temp-temp2<=-10 )|| tosi==false)
{
maara++;
temp3+=temp;
if(maara>=100)
{
ascii((temp3/100)/10,9,3); ascii((temp3/100)%10,11,1); maara=0;temp2=temp; tosi=true;
lcdkirj(); temp3=0;

}
}
kesklippu=0;

 }

}

}
```

```
void lcdinit()
{
                    /*PORTC=0b00110000;
                    e_puls();
                    _delay_ms(6);

                    PORTC=0b00110000;
                    e_puls();
                    _delay_ms(6);

                    PORTC=0b00110000;
                    e_puls();
                    _delay_ms(6);

                    PORTC=0b00100000;
                    e_puls();

                    sendcmd(0x28); _delay_ms(2);
                    sendcmd(0x08); _delay_ms(2);
                    sendcmd(0x01); _delay_ms(3);
                    sendcmd(0x06); _delay_ms(2);
                    sendcmd(0x0C); _delay_ms(2);
                    sendcmd(0x80); _delay_ms(1);
*/
}

void e_puls()
{
                    PORTC |=1<<2;
                    _delay_us(5);
                    PORTC ^=1<<2;
}

void sendcmd(unsigned char a)
{
                    cli();
                    data = 0b00001111 | a;
                    PORTC = (PORTC | 0b11110000) & data;
                    PORTC = PORTC & 0b11111110;
                    e_puls();
                    data = a<<4;
                    PORTC = (PORTC & 0b00001111) | data;
                    PORTC = PORTC & 0b11111110;
                    e_puls();
                    sei();
```

```
}

void sendchar(unsigned char a)
{
		cli();
		data = 0b00001111 | a;
		PORTC = (PORTC | 0b11110000) & data;
		PORTC = PORTC | 0b00000001;
		e_puls();
		data = a<<4;
		PORTC = (PORTC & 0b00001111) | data;
		PORTC = PORTC | 0b00000001;
		e_puls();
		sei();
}

void lcdkirj()
{
		/*
		int a;

		sendcmd(128); _delay_ms(2);
		sendcmd(2); _delay_ms(2);
		for(a=0;a<=31;a++)
		{
				sendchar(lcdbuf[a]); _delay_us(40);
				if(a==15){ sendcmd(0b11000000); _delay_us(40);}

		}
		*/
}
void ascii(long arvo,unsigned int paikka, unsigned int lkm)
{
		unsigned int digi;
		lkm--;
		while(lkm>0)
		{
				digi=arvo/dek[lkm];
				lcdbuf[paikka-lkm]=digi+'0';
				arvo=arvo-digi*dek[lkm];
				lkm--;
		}
		lcdbuf[paikka]=arvo+'0';
}
```

Liite 3. PC:n lämpötilaluennan lähdekoodi

```
//*****main.cpp*****

#include <QtGui/QApplication>
#include "mainwindow.h"
#include "device.h"
#include <QTimer>
#include <QMetaType>
#include <QFile>
#include <QTextStream>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QFile file("config.txt");
    file.open(QIODevice::ReadOnly);
    QTextStream in(&file);
    QString line1,line2;
    line1=in.readLine();
    line2=in.readLine();
    file.close();
    text blue("blue",line1),serial("serial",line2);
    MainWindow w;
    QObject::connect(&serial,SIGNAL(pass(const QString&,const QString&))
    ,&blue, SLOT(bluesend(const QString&, const QString& )));
    QObject::connect(&serial,SIGNAL(empty(const QString&))
    ,&blue, SLOT(empty2(const QString& )));
    QObject::connect(&serial,SIGNAL(wtext(const QString&))
    ,&w, SLOT(sread(const QString& )));
    blue.start();
    serial.start();
    w.show();

    return a.exec();
}



//********mainwindow.cpp********
#include "mainwindow.h"
#include "ui_mainwindow.h"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    eka=true;
    aika=0;
}
MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::sread(const QString& number)
{
    ui->lcdNumber->display(number);
    aika++;
    if(aika==600 || eka==true)
    {
        eka=false;
        aika=0;
        QDate date=QDate::currentDate();
        QString datestring=date.toString("dd_MM_yyyy");
        QTime time=QTime::currentTime();
        QString timestring=time.toString("hh.mm");
        QFile file(datestring+ ".txt");
        file.open(QIODevice::WriteOnly | QIODevice::Append | QIODevice::Text);
        QTextStream out(&file);
        out <<number<<'\t'<<timestring<<'\n';
```

```
        file.close();
    }
}


//********device.cpp*********
#include <device.h>
text::text(const QString& text, const QString &temp)
{
    valinta=text;
    m_stop=false;
         const char *port;
         QByteArray ba=temp.toLatin1();
         port=ba.data();
         hSerial = CreateFileA(port,GENERIC_READ | GENER-
IC_WRITE,0,0,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
         if(hSerial==INVALID_HANDLE_VALUE){
         if(GetLastError()==ERROR_FILE_NOT_FOUND){
         cout<<"porttia ei ole \n";}
         cout<<"jotain ihan muuta"<<endl;
         }
         DCB serialparameters = {0};
         serialparameters.DCBlength=sizeof(DCB);
         if(!GetCommState(hSerial, &serialparameters)){ cout<<"ERROR"<<endl;}
         serialparameters.BaudRate=CBR_9600;
         serialparameters.ByteSize=8;
         serialparameters.StopBits=ONESTOPBIT;
         serialparameters.Parity=NOPARITY;
         if(!SetCommState(hSerial, &serialparameters)){ cout<<"ERROR"<<endl;}
         COMMTIMEOUTS timeouts={0};
         timeouts.ReadIntervalTimeout=10;
         timeouts.ReadTotalTimeoutConstant=10;
         timeouts.ReadTotalTimeoutMultiplier=10;
         if(!SetCommTimeouts(hSerial, &timeouts)){cout<<"lol";}
}
void text::stop()
{
    m_stop=true;
}
void text::run()
{
    if(valinta=="serial")
    {
    while(!m_stop)
    {
        memset(buffer, 0, 50);
        DWORD bytesread;
        ReadFile(hSerial,buffer,50,&bytesread,NULL);
        if(bytesread>0)
        {
            m_text=buffer;
            emit wtext(m_text);
            emit pass(m_text,"ok");
        }
        else emit {m_text="fail"; empty(m_text);}
        sleep(1);
    }
}
    else if(valinta=="blue")
    {
        while(!m_stop)
        {
            if(m_text=="ok")
            {
                byteswrite=5;
                const char *str;
                QByteArray ba=to_phone.toLatin1();
                str=ba.data();
                WriteFile(hSerial,str,5,&byteswrite,NULL);
```

```
            }
         sleep(1);
        }
    }
}
void text::bluesend(const QString &temp,const QString &ok)
{
    to_phone=temp;
    m_text=ok;
}
void text::empty2(const QString &fail)
{
    m_text=fail;
}
```

```
//******device.h*********
#ifndef DEVICE_H
#define DEVICE_H
#include <QThread>
#include <windows.h>
#include <iostream>
#include <string.h>
using namespace std;
class text : public QThread
{
    Q_OBJECT
public:
    text(const QString& text,const QString&   );
    void run();
    void stop();
signals:
    void wtext(const QString&);
    void pass(const QString&,const QString& );
    void empty(const QString&);
public slots:
    void bluesend(const QString&,const QString& );
    void empty2(const QString&);
private :
        char buffer[50];
        QString m_text;
        bool m_stop;
        HANDLE hSerial;
        QString valinta;
        DWORD byteswrite;
        QString to_phone;
};
#endif // DEVICE_H
```

```
//*******mainwindow.h*********
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QDateTime>
#include <QFile>
#include <QTextStream>
#include <iostream>
namespace Ui {
    class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
```

```
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
public slots:
    void sread(const QString&);
signals:

private:
    Ui::MainWindow *ui;
    QString compare;
    bool eka;
    int aika;
};
#endif // MAINWINDOW_H
```

Liite 4. Android-sovelluksen lähdekoodi

```
package blue.lampo;

import java.util.UUID;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothSocket;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;


public class bluemain extends Activity {
    /** Called when the activity is first created. */
            TextView textView1;
            private static final String TAG ="Hammas";
            private static final boolean D=true;
            private BluetoothAdapter mBluetoothAdapter=null;
            private BluetoothSocket btsocket=null;
            private InputStream instream=null;
            private static final UUID MY_UUID = UUID.fromString("00001101-
            0000-1000-8000-00805F9B34FB");
            private static String address="00:02:72:20:35:22";
            boolean isrunning = false;

            Handler handler = new Handler(){

            @Override
            public void handleMessage(Message msg){
            byte[] readbuf=(byte[]) msg.obj;
            String readmessage= new String(readbuf,0,msg.arg1);
            textView1.setText("Lämpötila on "+ readmessage + "astetta");
            if (D) Log.e(TAG,"++ on read message ++");
                        }

            };

            @Override
             public void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            textView1 =(TextView) findViewById(R.id.textView1);
            textView1.setText("Lämpötila on "+ "astetta");
            if (D) Log.e(TAG,"++ ON CREATE ++");

            mBluetoothAdapter=BluetoothAdapter.getDefaultAdapter();

            if(!mBluetoothAdapter.isEnabled()){
            Toast.makeText(this, "ENABLE BLUETOOTH AND COME BACK
            AGAIN",Toast.LENGTH_LONG).show();
            finish();
        }


    }

             @Override
             public void onStart(){
```

```
super.onStart();
isrunning=true;
if (D) Log.e(TAG,"++ on start ++");

BluetoothDevice device=mBluetoothAdapter.getRemoteDevice(address);

try {
btsocket = device.createRfcommSocketToServiceRecord(MY_UUID);

} catch (IOException e){
            Log.e(TAG,"++ EPIC FAIL!! ++");
}
Log.d(TAG,"++ Socket connected ++");
}

@Override
public void onResume(){
super.onResume();
try {
            btsocket.connect();
            Log.e(TAG,"link connected ready to transfer");


} catch (IOException e){
            Log.e(TAG,"++ error connection!! ++");
            try {
            btsocket.close();

}           catch (IOException e2){
            Log.e(TAG,"++ Fail to close socket!! ++",e2);
}
return;
}

if (D) Log.e(TAG,"++ on resume ++");


try{
            instream=btsocket.getInputStream();

} catch (IOException e) { }
if (D) Log.e(TAG,"++ lol ++");
Thread background = new Thread(new Runnable(){
public void run() {

try{


while(isrunning){

byte[] buffer = new byte[1024];
int bytes;

//String data = null;


try{

bytes=instream.read(buffer);
Message msg=handler.obtainMessage(1,bytes,-1,buffer);
handler.sendMessage(msg);
if (D) Log.e(TAG,"++ luettu ++");

} catch (IOException e) {
break;
}
if (D) Log.e(TAG,"++ jep ++");
Thread.sleep(100);
}
```

```
                } catch (Throwable t){

                }
                if (D) Log.e(TAG,"++ on thread ++");

                }


                });
                if (D) Log.e(TAG,"++ on beforeback ++");
                background.start();
                if (D) Log.e(TAG,"++ on backstart ++");
        }

                @Override
                public void onPause(){
                super.onPause();
                isrunning=false;
                try {
                instream.close();
                } catch (IOException e1) {


                }


                try {
                btsocket.close();

                }catch (IOException e){
                Log.e(TAG,"++ error close!! ++");
                }

                if (D) Log.e(TAG,"++ on pause ++");
        }

                @Override
                public void onStop(){
                super.onStop();
                if (D) Log.e(TAG,"++ ON STOP ++");
        }

                @Override
                public void onDestroy(){
                super.onDestroy();
                if (D) Log.e(TAG,"++ ON Destroy ++");
}

}
```