OULUN SEUDUN
AMMATTIKORKEAKOULU

Esa-Matti Ollila

LANGATTOMAN SENSORIVERKON JATKOKEHITYS

LANGATTOMAN SENSORIVERKON JATKOKEHITYS

Esa-Matti Ollila
Opinnäytetyö
Kevät 2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikka, langaton tietoliikenne

Tekijä: Esa-Matti Ollila
Opinnäytetyön nimi: Langattoman sensoriverkon jatkokehitys
Työn ohjaaja: Riitta Rontu
Työn valmistumislukukausi ja -vuosi: Kevät 2012 Sivumäärä: 46

Tämän työn tarkoituksena oli ottaa käyttöön kolme erilaista sensoria ja suunnitella niille toimiva sovellus Matrix Multimedian kehittämään E-Blocks-ympäristöön. E-Blocks-ympäristössä sensoritiedot lähetettiin käyttäen ZigBee-tekniikkaa laitteelta toiselle. Sovellusten ohjelmien suunnitteluun käytettiin saman yrityksen kehittämää Flowcode-ohjelmointiympäristöä. Työn tilaajana toimi Juha Juuti Oulun seudun ammattikorkeakoulun tietotekniikan osastolta.

Työssä käytettäviksi sensoreiksi valittiin kosteus- ja happisensori sekä sykemittari. Jokaisen sensorin ohjelmisto suunniteltiin erikseen käyttäen Flowcode-ohjelmointiympäristöä. Tuloksena saatiin kolme toimivaa sensorisovellusta, jotka lähettivät sensoreitten mittaustiedon laitteelta toiselle. Lähetettävät mittaustiedot olivat suhteellinen kosteus, ympäröivän ilman happipitoisuus sekä sydämen syke minuutissa.

Työtä voidaan hyödyntää koulussa esimerkiksi tulevilla opintojaksoilla. Opintojaksoilla voidaan havannoillistaa sensoreitten toimintaa sekä Flowcode-ympäristön käyttöä työn tulosten avulla. Myös mahdollista on sovellusten ominaisuuksien jatkokehittäminen edelleen.

Asiasanat: Sensoriverkot, ohjelmointiympäristö, mikrokontrollerit

ALKULAUSE

Haluan kiittää työn aiheesta lehtori Riitta Rontua ja projektisuunnittelija Juha Juutia. Työn tekstin ja kieliopin ohjaamisesta kiitokset lehtori Tuula Hopeavuorelle. Kiitos Oulun seudun ammattikorkeakoulun tekniikan yksiköllle mahdollisuudesta käyttää tietoliikennelaboratoriota työn tekemiseen.

Oulussa 31.5.2012

Esa-Matti Ollila

SISÄLLYS

1 JOHDANTO

Tekniikan käyttö yhteiskunnan jokaisella osa-alueella on yleistynyt huomattavasti viimeisen kahdenkymmenen vuoden aikana. Monenlaiset tehtävät ovat siirtyneet ihmisen suorittamasta työstä tekniikan hoidettavaksi. Tärkeä osa teknisten laitteiden ihmiselle tuottamaa hyötyä on automaatio. Erilaisten mittausten suorittaminen on helpottunut ja automatisoitunut, sekä valvonta ja etäohjaus tulleet entistä helpommaksi.

Sensorit muodostavat tärkeän osan edellämainittujen osa-alueiden toiminnasta. Varsinkin kun sensoreihin yhdistetään langaton tiedonsiirto, voidaan rakentaa helposti tarpeita vastaava, hyvin joustava ja monipuolinen järjestelmä.

Tämän työn tarkoituksena oli valita kolme erityyppistä sensoria ja tehdä niille toimiva sovellus Matrix Multimedian kehittämään E-Blocks-ympäristöön. E-Blocks-ympäristössä tarkoituksena oli lähettää mittausdata solmulta toiselle langattomasti ja näyttää tulos graafisella LCD-näytöllä. Langattomassa tiedonsiirrossa käytettiin apuna ZigBee-tekniikkaa. Työ toteutettiin Oulun seudun ammattikorkeakoulun tekniikan yksikön tietoliikennelaboratoriossa.

Työ aloitettiin antureiden valinnalla. Alkuvaiheen miettimisen jälkeen päädyttiin valitsemaan kosteus- ja happisensori sekä käsistä sydämen sykettä mittaava sensori.

Aihe oli ennestään perusteiltaan tuttua, koska E-Blocks ympäristöä, ZigBeetä ja Flowcode-ohjelmointityökalua oli käytetty jo koulussa aikaisemmin olleella opintojaksolla.

2 LANGATTOMAT SENSORIVERKOT

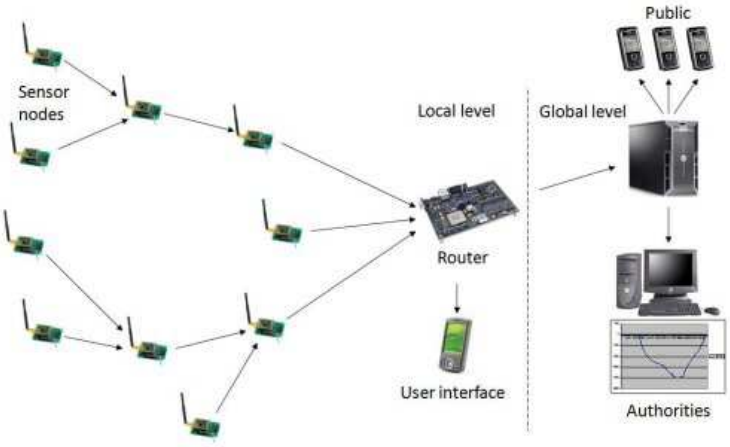
Monien teknisten järjestelmien toiminta perustuu jonkin mitattavan muutoksen tai luonnonilmiön pohjalle. Luonnonilmiön muutos voi esimerkiksi tapahtua lämpötilassa, paineessa tai kosteudessa. Sensoriverkon avulla voidaan järjestää suureiden mittaus, ja tuloksia voidaan käyttää sellaisenaan tai jonkin järjestelmän osana. Toinen esimerkki muutoksesta voi olla esimerkiksi oven avaaminen, säiliön täyttöaste tai liiketunnistimen havaitsema liike. Näitä tietoja voidaan käyttää hyväksi etähallinnassa ja valvonnassa.

Langattomat sensoriverkot ovat kasvava ja nopeasti kehittyvä tuote- ja tutkimuskehitysalue. Sensoriverkkojen yleistymistä ovat edistäneet teknologian kehittyminen anturi- ja prosessoritekniikassa sekä langattoman tietoliikenteen alalla. Kehityksen mukana hinnat ovat laskeneet, mikä osaltaan on vaikuttanut yleistymiseen. (1.)

Yksittäinen sensoriverkkoon kuuluva solmu (node) koostuu yleisesti mikrokontrollerista tai prosessorista, sensorista, virtalähteestä ja langattoman tiedonsiirron hoitavasta osasta (kuva 1). Solmu voi olla kytketty verkkovirtaan, tai siinä voi olla oma sisäinen akku tai patteri.

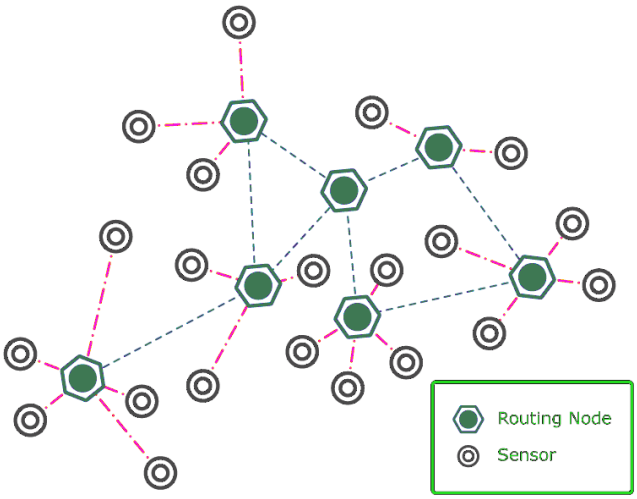
KUVA 1. Aluevalvontaan käytettävä solmu (2)

Sensoriverkon laajuus voi vaihdella käyttötarkoituksen mukaan muutamasta sensorikohteesta tuhansien sensoreiden muodostamaan laajaan verkkoon. Suurissa järjestelmissä dataa ei välttämättä lähetetä suoraan yhteen kohteeseen, vaan lähete voi koostua usean sensorin tiedoista, jotka kootaan keskussolmun avulla ja lähetetään taas eteenpäin (kuva 2). Pienemmissä järjestelmissä tälläinen järjestely ei välttämättä ole tarpeellista. (1.)



KUVA 2. Sensoriverkko (3)

Oman rajoituksensa verkkojen laajuuteen antaa langattoman yhteyden kantomatkan rajallisuus. Laajemmissa järjestelmissä on siis välttämätöntä käyttää useampaa keskussolmua, joiden avulla mittausaluetta voidaan laajentaa (kuva 3). (1.)



KUVA 3. Verkon laajentaminen keskussolmujen avulla (4)

Sensoriverkkojen langattomia yhteyksiä varten on kehitetty ja ollaan kehittämässä erilaisia tekniikoita. Osa tekniikoista on muistakin yhteyksistä tuttuja, kuten yleinen lähiverkkotekniikka WLAN ja matkapuhelimissa käytetty Bluetooth. Erityisesti sensoriverkkoja varten kehitettyihin tekniikoihin kuuluu ZigBee, jota käytetään tämän työn toteuttamisessa. (1.)

3 ZIGBEE

ZigBee on langaton IEEE 802.15.4 -standardiin perustuva tietoverkkotekniikka, jonka ensimmäinen spesifikaatio ratifoitiin joulukuussa 2004. Se on suunniteltu käytettäväksi erityisesti teollisuuden kohteissa joissa pieni virrankulutus, luotettavuus, halpa hinta ja yksinkertainen käytettävyys nousevat tiedonsiiirtonopeutta tärkeämmiksi. Tällaisia kohteita voivat olla esimerkiksi sensoriverkot, kaukosäätökäyttö ja automaatio. (5; 6, s. 6.)

3.1 IEEE 802.15.4

IEEE 802.15.4 -standardin langattomia laitteita varten kehitti IEEE, joka on kansainvälinen tekniikan alan järjestö. Standardi määrittää protokollan fyysisen ja MAC-kerroksen verkoille, joissa tarvitaan pientä virrankulutusta ja käytetään vain pientä tiedonsiirtonopeutta. Ylemmät kerrokset kehitetään jokaisen standardia käyttävän tekniikan omien vaatimusten mukaan. ZigBeen lisäksi muita IEEE 802.15.4 -standardia käyttäviä verkkotekniikoita ovat esimerkiksi WirelessHART ja MiWi. (6, s. 10; 7.)

Fyysinen kerros (PHY)

Fyysisen kerroksen spesifikaatiossa määritellään ilmarajapinnassa käytettävät radiotaajuudet. Fyysisen kerroksen tehtäviä ovat yhteyksien muodostaminen sekä yhteyden laadun tarkkailu ja säätö. (6, s. 10.)

Käytössä on kolme lisensoimatonta radiotaajuutta, joiden taajuudet ovat 2,4 GHz, 915 MHz ja 868 MHz. 915 MHz:n taajuus on tarkoitettu käytettäväksi USA:ssa, kun taas 868 MHz:n taajuutta käytetään Euroopassa ja Japanissa. Taajuudet eroavat toisistaan myös tiedonsiirtonopeudellaan. 2,4 GHz:n taajuuden suurin nopeus on 250 kb/s, 915 MHz:n taajuuden 40 kb/s ja 868 MHz:n taajuuden 20 kb/s. (6, s. 10.)

Standardissa määritellään käytettäväksi digitaalisia modulaatiomenetelmiä, joiden toiminta perustuu signaalin arvon muutteluun. Signaalin arvo voi siis olla joko looginen 0 tai 1. Käytettäväksi modulaatiotekniikoiksi on määritelty O-QPSK (Offset Quadrature Phase Shift Keying) ja DSSS (Direct Sequence Spread Spectrum), jota käytetään ZigBeessä. (6, s.10–11.)

DSSS-menetelmä perustuu hajaspektritekniikkaan. Hajaspektritekniikoissa kapeakaistainen digitaalinen hyötysignaali hajautetaan leveälle taajuuskaistalle lähetystä varten. (8.)

MAC-kerros

MAC (Medium Access Layer) -kerroksella hoidetaan radioaaltojen jako IEEE 802.15.4 -standardia käyttävien laitteiden kesken. Lähetysvuorojen määrittelyssä käytetään CSMA-CA (Carrier Sensing Multiple Access with Collision Avoidance) -tekniikkaa. (6, s. 10.)

MAC-kerroksella hoidetaan myös mahdollisten käytössä olevien turvallisuusprotokollien hallinta. IEEE 802.15.4 -standardin paketin suurin mahdollinen koko on 127 tavua, johon sisältyy myös CRC (Cyclic Redundancy Check) -arvo, jonka pituus on 2 tavua. (6, s. 10.)

Laitetyypit

Standardissa määritellään käytettäväksi kaksi erilaista laitetyyppiä, FFD (Full Function Device) ja RFD (Reduced Function Device). (6, s. 12.)

FFD:n tehtäviä ovat reititys, verkon muodostaminen sekä muut ylläpitotehtävät. RFD toimii dataa lähettävänä solmuna, kuten esimerkiksi sensorisovelluksessa mittaustietoja tietyin ajanvälein lähettävänä laitteena. (6, s. 12.)

RFD:n ei tarvitse olla koko aikaa aktiivisena kuten verkkoa ylläpitävä FFD, joten RFD:n virrankulutusta voidaan näin pitää tarvittaessa matalana. Tämä on etu käyttökohteissa joissa käytetään akkuja tai paristoja jatkuva-aikaisen virtalähteen sijaan. (6, s. 12.)

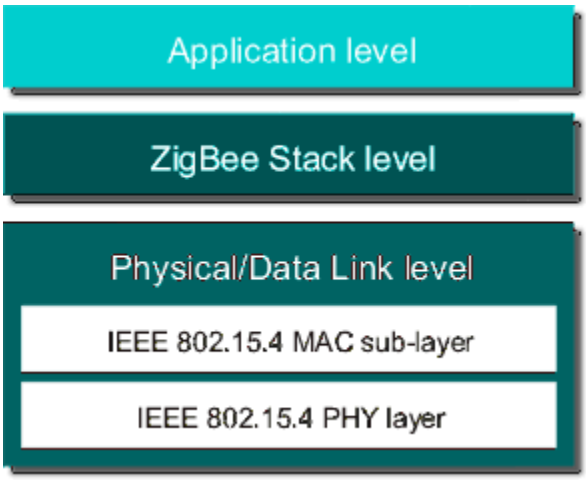
3.2 Protokolla

ZigBeen protokolla voidaan jakaa kolmeen erilliseen kerrokseen, joilla kullakin on omat yksityiskohtaiset tehtävänsä (kuva 4).

Ylimpänä kerroksena protokollassa on sovelluskerros (Application level) jota yleensä ohjaa mikrokontrolleri. Sovelluskerroksen tehtävänä on ohjata ZigBee-laitetta käyttäviä sovelluksia, kuten esimerkiksi mittausta sensorin avulla. (6, s. 9.)

Toiseksi alimpana kerroksena on ZigBeen pinokerros (ZigBee Stack level). Pinokerroksella sijaitsevat protokollat jotka hoitavat tietoturvaa sekä tiedon reititystä verkossa. Mikrokontrolleri hoitaa osaa pinokerroksen tehtävistä. (6, s. 9.)

Alimmassa kerroksessa sijaitsevat IEEE 802.15.4 -standardin mukaiset fyysinen ja MAC-kerros.



KUVA 4. ZigBeen protokollapino (9)

3.3 Loogiset laitteet

ZigBee-verkossa on mahdollisuus käyttää kolmea erityyppistä loogista laitetta, joilla kullakin on erilaisia ominaisuuksia. Jokainen laite on myös jaoteltu IEEE 802.15.4 -standardin mukaisesti FFD- tai RFD-laitteeksi. (6, s. 13.)

Co-ordinator

Co-ordinator eli koordinaattori on FFD-laite. Sen tehtävänä on verkon luominen, verkko-osoitteiden jakaminen, kanavan määrittäminen ja uusien nodejen liittäminen verkkoon. Yhdessä ZigBee-verkossa voi olla vain yksi koordinaattori. (6, s. 13.)

Normaalisti koordinaattori saa virtansa sähköverkosta ja on päällä jatkuva-aikaisesti, joten sitä käytetään pääasiallisena liityntäpisteenä verkkoon. Tällöin uusien verkkoon vielä kuulumattomien nodejen on mahdollista liittyä verkon jäseneksi. Jos kuitenkin koordinaattori kytketään pois päältä, pystyvät ainoastaan jo verkko-osoitteen saaneet nodet kommunikoimaan keskenään, koska koordinaattori ei ole jakamassa uusia osoitteita. (6, s. 13.)

Router

Router eli reititin on FFD-laite. Reitittimen tehtäviä verkossa ovat pakettien puskurointi ja signaalien reititys nodejen välillä. Reitittimen avulla voidaan verkon kantomatkaa ja aluetta laajentaa. (6, s. 13.)

Tehtävien luonteen takia reititin on usein kytketty verkkovirtaan, jotta jatkuva-aikainen toimivuus voidaan varmistaa. Koska reititin on yleensä jatkuva-aikaisesti päällä, voidaan sitä tarvittaessa myös käyttää liittymispisteenä verkkoon. (6, s. 13.)

End device

End device on päätelaite ZigBee-verkossa, ja se voi olla joko FFD- tai RFD-laite. Päätelaitteet voidaan asettaa sensoreiden, kytkimien, näyttöjen jne. yhteyteen, joilloin niiden tehtäväksi muodostuu tiedon kuljettaminen tarvittaviin verkko-osoitteisiin. (6, s. 13.)

Päätelaitteet on suunniteltu kuluttamaan vähän virtaa, ja ne ovatkin yleisesti akku- tai patterikäyttöisiä. Virrankulutusta voidaan vähentää entisestään asettamalla päätelaite sleep-tilaan eli nukkumaan, jolloin se herää vain halutuin väliajoin lähettämään ja vastaanottamaan. (6, s. 13.)

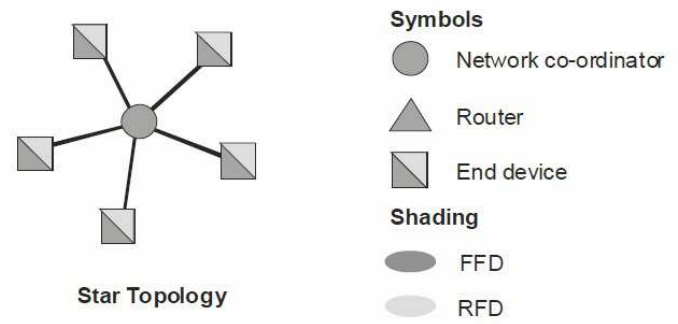
Päätelaitteet eivät pysty suoraan kommunikoimaan keskenään. Niiden väliseen kommunikointiin tarvitaan välittäjäksi koordinaattori tai reititin. (6, s. 13.)

3.4 Topologiat

ZigBeessä käytettävät topologiat on määritelty protokollan verkkokerroksella. Käytettäviä topologioita ovat star-, cluster tree- ja mesh-topologiat.

Star

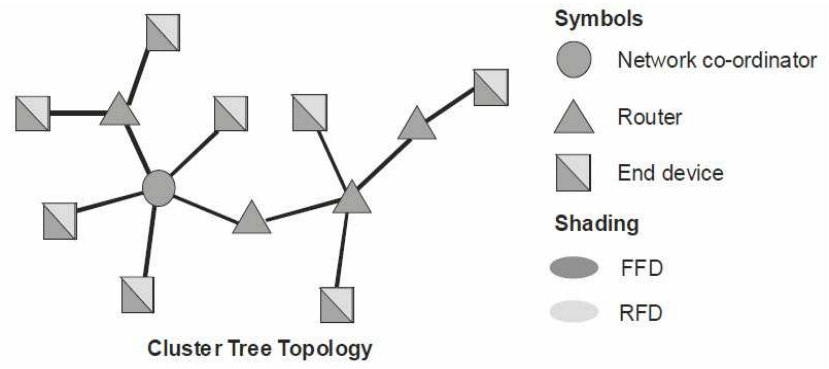
Star- eli tähtitopologia koostuu koordinaattorista ja siihen yhteyden ottavista päätelaitteista (kuva 5). Tieto päätelaitteelta toiselle kulkee aina koordinaattorin kautta. Jos koordinaattori sammutetaan tai sen toiminta muuten estyy, päätelaitteiden välinen kommunikointi ei ole mahdollista. (6, s. 14.)



KUVA 5. Tähtitopologia (6)

Cluster tree

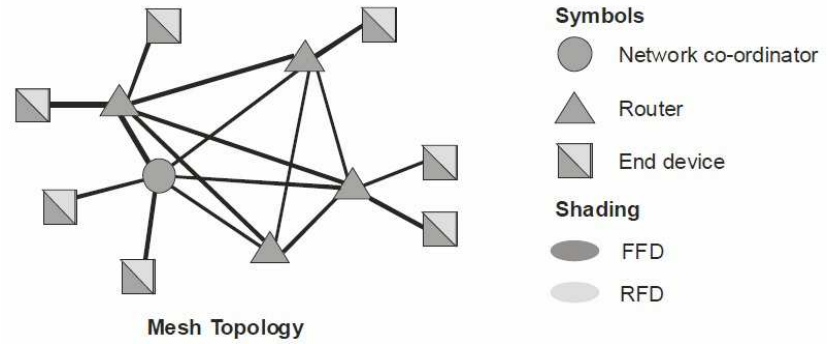
Cluster tree eli puutopologiassa liityntäpisteitä verkkoon laajennetaan reitittimien avulla. Koordinaattori on yhteydessä reitittimiin joiden avulla verkon laajennus tapahtuu, jolloin reitittimien muodostaman ketjun varrella laitteet voivat liittyä verkkoon (kuva 6). Päätelaitteet eivät voi keskustella suoraan keskenään, vaan yhteys on muodostettava reitittimen tai koordinaattorin kautta. (6, s. 15.)



KUVA 6. Puutopologia (6)

Mesh

Mesh-toplologia on muunnelma puutopologiasta. Puutopologiassa verkkoon voi helposti tulla katkos esimerkiksi yhden reitittimen pudotessa pois verkosta, jolloin verkon osat eriytyvät toisistaan. Mesh-topologiassa koordinaattori ja kaikki reitittimet ovat yhteydessä toistensa kanssa, joten yhden reitittimen vikaantuminen ei aiheuta samanlaista katkosta verkon toimintaan (kuva 7). Luotettavamman toiminnan lisäksi mesh-topologian etuna on reitityksen parempi optimointi, jolloin latenssiajat saadaan pienemmiksi. (6, s. 15.)



KUVA 7. Mesh-topologia (6)

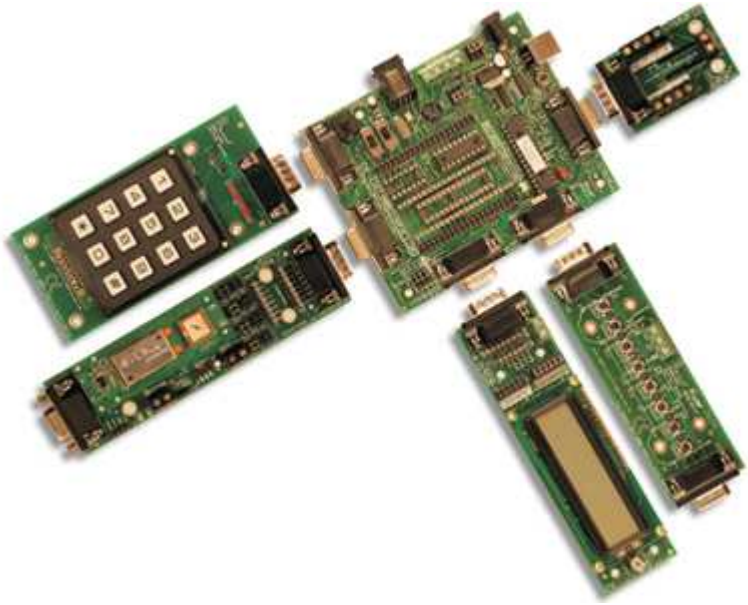
4 MATRIX MULTIMEDIA

Matrix Multimedia on englantilainen vuonna 1993 perustettu yritys, joka on erikoistunut valmistamaan sekä kehittämään elektroniikan koulutuskäyttöön tarkoitettuja ratkaisuja. Yritys käyttää liikevaihdostaan joka vuosi noin 25 % uusien tuotteiden kehitykseen. Tunnetuimpia yrityksen tuotteiteita ovat E-Blocks-rakennussarja sekä Flowcode-ohjelmisto, joka on helppokäyttöinen graafinen ohjelmointiympäristö. (10.)

4.1 E-Blocks-rakennussarja

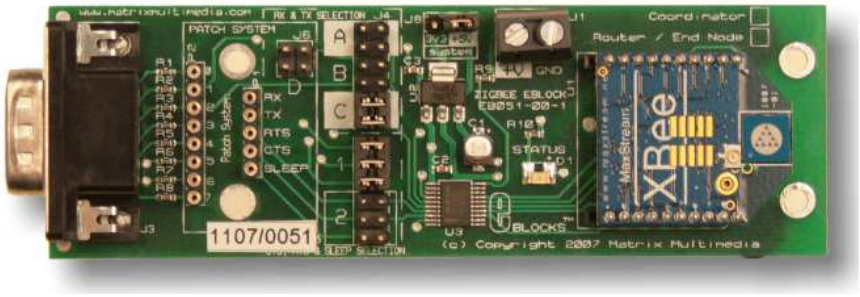
E-Blocks-rakennussarja koostuu erilaisista yhdistettävistä piirilevyistä, joilla jokaisella voi olla erilaisia toimintoja. Piirilevyjä kutsutaan lohkoiksi (Blocks), josta rakennussarjan nimi on peräisin.

Lohkojen toimintoja ohjataan mikrokontrollerilla, joka toimii piirilevyjen kytkentäalustana. Oulun seudun ammattikorkeakoulun Tekniikan yksikössä mikrokontrollerina on käytössä Atmelin AVR-perheeseen kuuluva ATMEGA 324P, joka toimii 20 MHz:n taajuudella. Lohkoille voi olla toteutettu esimerkiksi näppäimistö, LCD-näyttö tai painonappeja, joita ohjataan mikrokontrollerin avulla (kuva 8). (11.)



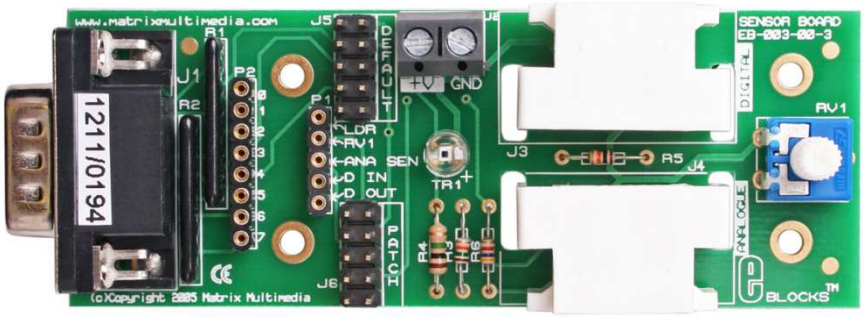
KUVA 8. E-Blocks-rakennussarja (11)

Saatavilla on myös tiedonsiirron mahdollistavia lohkoja (esim. Ethernet, ZigBee, Bluetooth, WLAN) joiden avulla yksittäiset E-Blocks-kokonaisuudet voidaan kytkeä keskustelemaan toistensa tai muiden samaa tiedonsiirtotekniikkaa käyttävien laitteiden kanssa (kuva 9). Tällöin jokainen mikrokontrollerin ohjaama kokonaisuus toimii omana solmunaan (node). (11.)



KUVA 9. ZigBee-liitäntälohko (12)

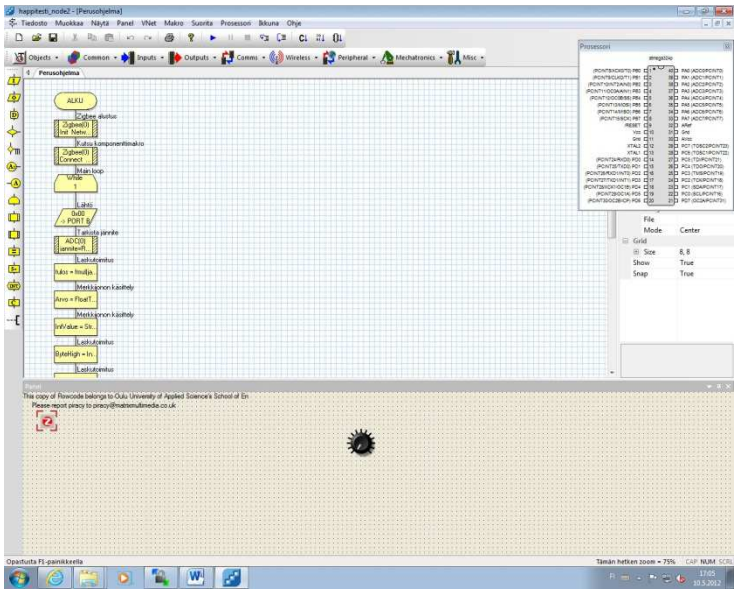
Toimintoja voidaan vielä laajentaa esimerkiksi lohkolla, johon voidaan kytkeä sensoreita analogisen tai digitaalisen tulon kautta (kuva 10). Käytettäviä sensoreita on tarjolla monia erilaisia, kuten lämpö-, syke- ja ilmanpainesensorit. (11.)



KUVA 10. Sensoriliitäntälohko (13)

4.2 Flowcode

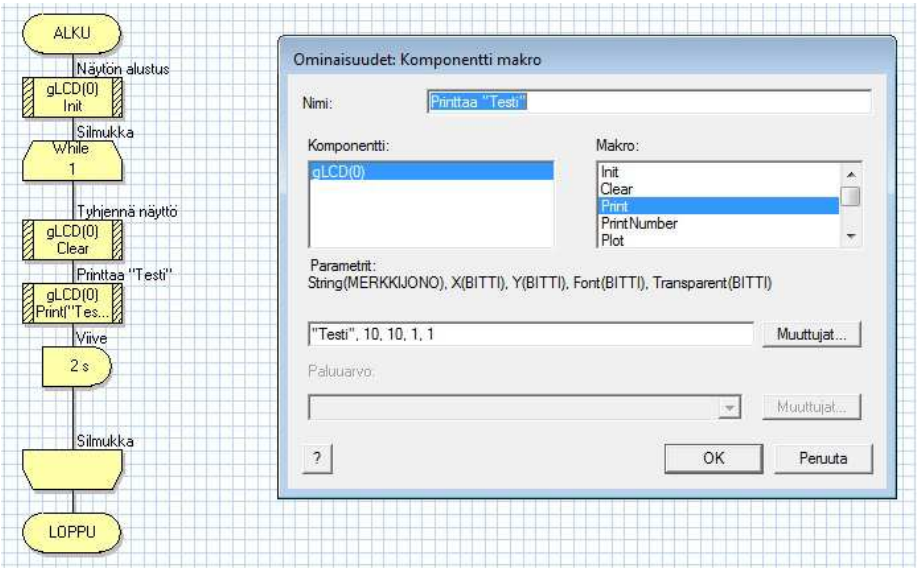
Flowcode on erittäin helppokäyttöinen ohjelmointityökalu jossa ohjelmointi tapahtuu graafisessa ympäristössä (kuva 11). Tavoitteena on mahdollistaa vähän ohjelmointikokemusta omaaville tai sitä vailla oleville henkilöille helppo tapa tutustua ohjelmointiin. Flowcodea voidaan käyttää myös E-Blocksin kanssa teollisuudessa apuna tuotekehityksen nopeuttamisessa. (14.)



KUVA 11. Flowcode-ympäristö

Flowcode-ohjelmoinnin perustana on vuokaavio, johon kerätään haluttuja ominaisuuksia sisältäviä lohkoja. Lohko voi olla esimerkiksi komponenttimakro, jonka avulla voidaan ohjata LCD-näytön toimintaa. Ohjelmaan voidaan myös lisätä omaa C-koodia sitä varten suunnitellun lohkon avulla.

Lohkot vedetään vuokaavioon vasemmalla puolella olevasta palkista, jonka jälkeen niiden asetuksia päästään muokkaamaan. Asetusten muokkaaminen on toteutettu yksinkertaisesti liukuvalikoilla ja valmiiksi määritetyillä parametreillä, joiden arvot voidaan säätää halutuiksi (kuva 12). Kuvassa on toimiva E-Blocksiin tarkoitettu ohjelma joka tulostaa graafiselle LCD-näytölle (gLCD) kahden sekunnin välein tekstin ”Testi”.



KUVA 12. Esimerkkiohjelma

Kun ohjelma on valmis, se pitää kääntää prosessorin ymmärtämään muotoon kääntäjän avulla. Flowcode-ympäristössä kääntäjä kääntää ohjelman aluksi C-kielelle, sen jälkeen Assemblyksi ja lopuksi heksatiedostoksi.

E-blocks-ympäristön AVR-prosessoria varten tarvitaan vielä ISP (In-System Programmer), joka kytketään E-blocksin ja tietokoneen väliin. Tietokoneeseen ISP kiinnitetään USB-kaapelin avulla ja E-blocksiin omalla kuusipinnisellä liittimellään (kuva 13). ISP:n tehtävänä on kääntää ohjelma AVR:n ymmärtämään konekieliseen muotoon.

KUVA 13. In-System Programmer (15)

5 TYÖSSÄ KÄYTETTÄVÄT SENSORIT

Kaikki tässä työssä käytettävät sensorit ovat Vernierin vamistamia. Matrix Multimedia toimii jälleenmyyjänä sensoreille, ja E-blocksin sensoriliitäntälohko on suunniteltu Vernierin sensoreissaan käyttämää liitäntää varten.

5.1 Kosteussensori

Kosteussensoria käytetään ilman suhteellisen kosteuden mittaamiseen (kuva 14). Sensorin toiminta perustuu integroituun piiriin, joka käyttää kapasitiivista polymeeriä kosteuden tunnistamiseen. Kosteuden arvon mittaaminen onnistuu integroidun piirin muodostaman jännitteen avulla, joka vaihtelee kosteuspitoisuuden mukaan. (16.)

KUVA 14. Vernierin kosteussensori (16)

5.2 Happisensori

Ympäröivän ilman happipitoisuutta voidaan mitata happisensorin avulla (kuva 15). Sensori toimii sähkökemiallisen solun avulla, joka sisältää lyijyanodin ja kultakatodin upotettuna elektrolyyttiin. Anodin ja katodin välillä tapahtuvan reaktion seurauksen muodostuu jännite, jonka avulla happipitoisuus voidaan määrittää. Happipitoisuutta on mahdollista mitata asteikolla 0–27%. (17.)

KUVA 15. Happisensori (17)

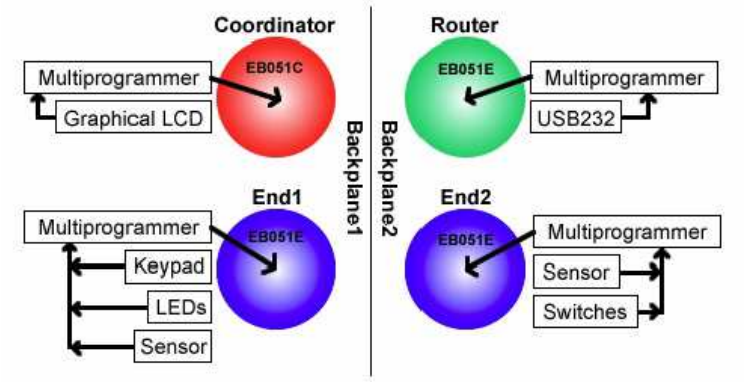
5.3 Sykemittari

Sykemittarin avulla voidaan määrittää henkilön sydämen lyöntitiheys eli syke. Työssä käytettävä anturi mittaa signaaleja käsissä pidettävien elektrodien avulla (kuva 16). Sydämen lyödessä iholla kulkee sähköinen signaali, jonka perusteella syke on mahdollista määrittää. Sydämen lyödessä mittari lähettää langattomasti matalataajuisella sähkömagneettisella tekniikalla vastaanottimeen pulssin. Vastaanotin lähettää edelleen jokaisen pulssin vastaanotettuaan 3 voltin pulssin liitäntäjohtoa pitkin vastaanottavalle laitteelle. (18.)

KUVA 16. Sykemittari ja vastaanotin (18)

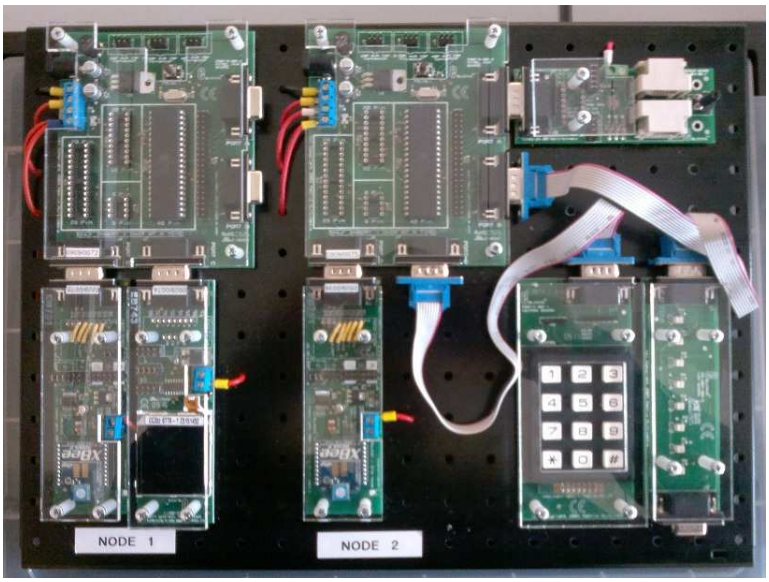
6 TYÖN TOTEUTUS

Työ toteutettiin käyttämällä Matrix Multimedian ZigBee-kurssin ohjeen mukaan koottua E-blocks-kokoonpanoa. Kurssissa käytettäväksi on määritelty neljä erilaista nodea joihin kuhunkin on liitetty erilaisia lohkoja (kuva 17).



KUVA 17. ZigBee-kurssin kokoonpanot (2)

Työssä käytetiin kahta eri nodea, jotka olivat kiinnitettynä samalle peltialustalle. Node 1:een oli kiinnitetty ZigBee- ja gLCD-näyttölohkot, ja node 2:ssa kiinnitettyinä olivat ZigBee-, näppäimistö-, LED- ja sensoriliitäntälohko (kuva 18). ZigBee-lohkojen asetukset oli määritetty siten, että node 1 toimi koordinaattorina ja node 2 päätelaitteena.

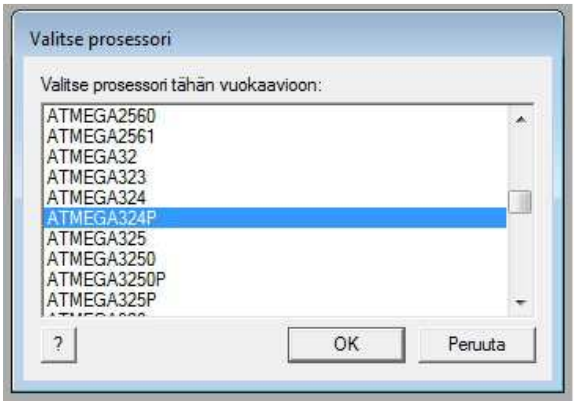


KUVA 18. Työssä käytetty kokoonpano

6.1 Alkujärjestelyt

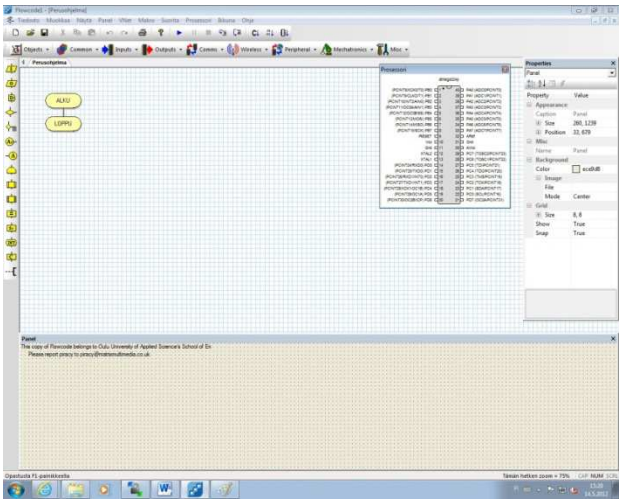
Työn tekeminen aloitettiin perehtymällä Matrix Multimedian ZigBee Student Notes -ohjeeseen, joka oli tietopaketti E-Blocksille tarkoitettua ZigBee-kurssia varten. Ohjekirjassa oli Flowcodea varten esimerkkitehtäviä, joiden avulla pääsi työn alkuun.

Uusi Flowcode-projekti aloitettiin alkuasetusten määrittämisellä. Ensimmäisenä valitaan prosessori, joksi valittiin listalta Oulun ammattikorkeakoulun tekniikan yksikössä käytössä oleva ATMEGA324P (kuva 19).



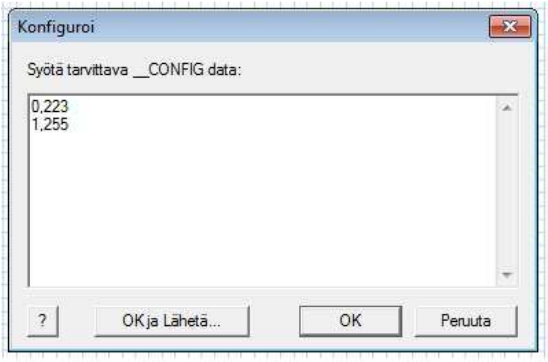
KUVA 19. Prosessorin valinta

Prosessorin valinnan jälkeen avautuu kehitysympäristö, johon itse ohjelmaa voidaan alkaa rakentaa (kuva 20). Ennen ohjelman rakentamisen aloittamista kuitenkin tehdään loputkin tarvittavista alkuasetuksista.



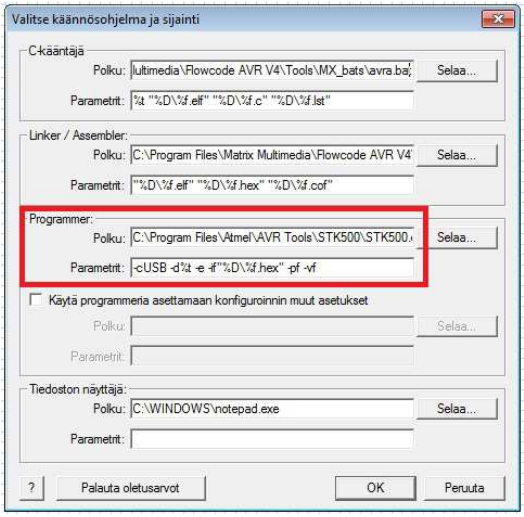
KUVA 20. Tyhjä Flowcode-projekti

Prosessorin asetuksiin lisätään ZigBee-kurssin ohjeessa annetut arvot 0,223 ja 1,255 (kuva 21). Nämä arvot ovat ainoastaan ATmega324P-prosessoria varten.



KUVA 21. Prosessorin asetukset

Tämän jälkeen prosessorin ohjelmoijalle heksatiedoston lähettävän, tietokoneella sijaitsevan ohjelman polku vaihdetaan oikeaksi. Polun korjauksen yhteydessä myös varmistetaan, että tarvittavat parametrit ovat ohjeistuksen mukaisia (kuva 22).



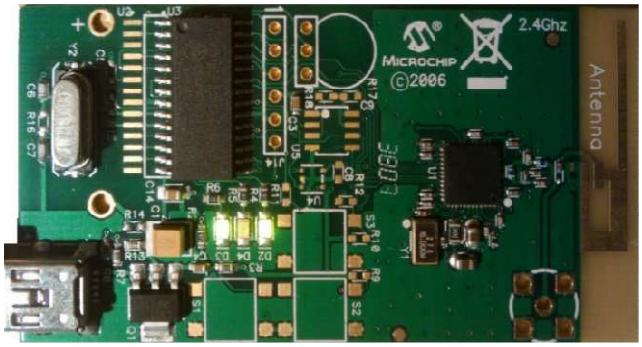
KUVA 22. Ohjelmoijan polku ja parametrit

Alkuasetusten määrittämisen jälkeen päästiin aloittamaan harjoitustehtävän tekeminen. Kokoonpanon toiminta varmistettiin ZigBee-kurssin materiaalissa olevan Exercise 2 -harjoitustehtävän avulla (19, s. 17–19). Harjoitustehtävän tarkempaa rakennetta ei käydä läpi, koska sen avulla oli vain tarkoitus testata kokoonpanon toimivuutta.

Harjoitustehtävässä tarkoituksena oli muodostaa yhteys kahden noden välille. Lopputuloksena oli ohjelma, jossa päätelaitteen (node 2) näppäimistöllä painettu merkki lähetetään ZigBeen avulla koordinaattorille (node 1), jossa se tulostuu gLCD-näytölle.

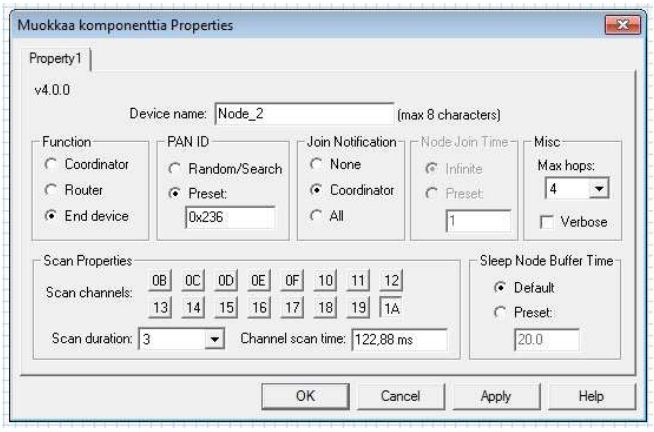
Ohjelma saatiin toimimaan kuten pitikin, mutta lähetyksessä esiintyi välillä häiriöitä. Välillä nodet eivät saaneet toisiinsa yhteyttä, eikä näppäimistön merkin lähetys toiminut ollenkaan.

Mahdollisten ZigBee-verkossa esiintyvien häiriöiden selvittämiseksi käytettiin apuna Zena-verkkoanalysaattoria (kuva 23). Zenan avulla on mahdollista seurata IEEE 802.15.4 -standardin mukaisten verkkojen kuten ZigBeen liikennettä eri kanavilla. Zena kiinnitetään tietokoneeseen USB (Universal Serial Bus) -kaapelin avulla, jolle se lähettää mitatut tulokset. Zenan käyttö tapahtuu oman ohjelman avulla, joka on asennettu tietokoneelle. (20.)



KUVA 23. Zena-verkkoanalysaattori (20)

Zenan avulla todettiin, että monella kanavalla esiintyi ylimääräistä liikennettä. Soveltuvimmaksi kanavaksi ilmeni 1A, jolla liikennettä ei Zenan mukaan ollut ollenkaan. Kanavaa 1A päätettiin siis käyttää vastaisuudessa nodejen välisessä liikenteessä, joten sitä käytettiin kummankin noden Flowcode-koodin ZigBee-asetuksissa (kuva 24).

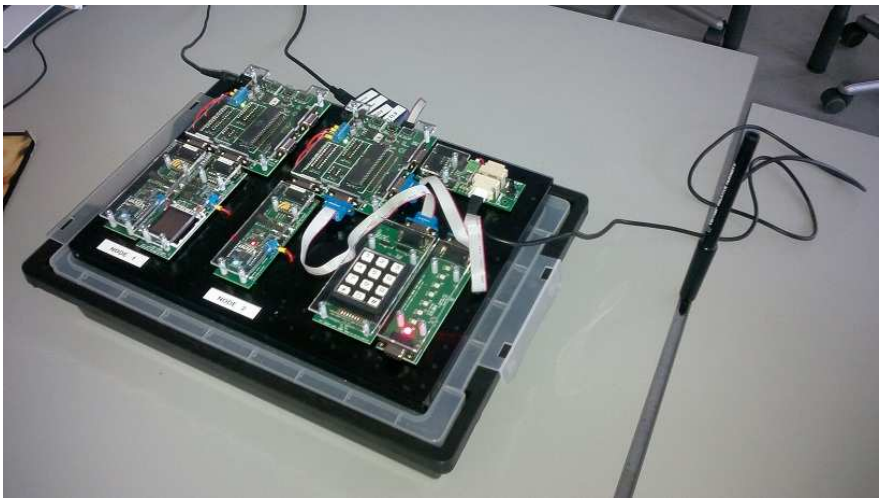


KUVA 24. Päätelaitteen ZigBee-asetukset

Koordinaattorille ja päätelaitteelle harjoitustehtävän avulla tehdyn toimivuustestin jälkeen aloitettiin jokaiselle sensorille yksilöllisesti räätälöityjen ohjelmien suunnittelu.

6.2 Kosteussensori

Tarkoituksena oli saada päätelaitteen sensoriliitäntälohkoon kytketyn kosteussensorin mittaama tulos muutettua esitettäväksi arvoksi (kuva 25). Muutoksen jälkeen tulos lähetettäisiin ZigBeen avulla koordinaattorille, jossa tulos esitettäisiin prosenttilukuna gLCD -näytöllä.



KUVA 25. Kosteussensori kytkettynä päätelaitteen sensoriliitäntälohkoon

Kosteussensorin ohjelman suunnittelu aloitettiin perehtymällä kosteusanturin datalehteen. Datalehdessä ilmoitettujen arvojen perusteella voitiin suunnitella tarvittava laskukaava oikean tuloksen saamiseksi. Datalehdestä saatujen arvojen perusteella muodostettiin kaava 1.

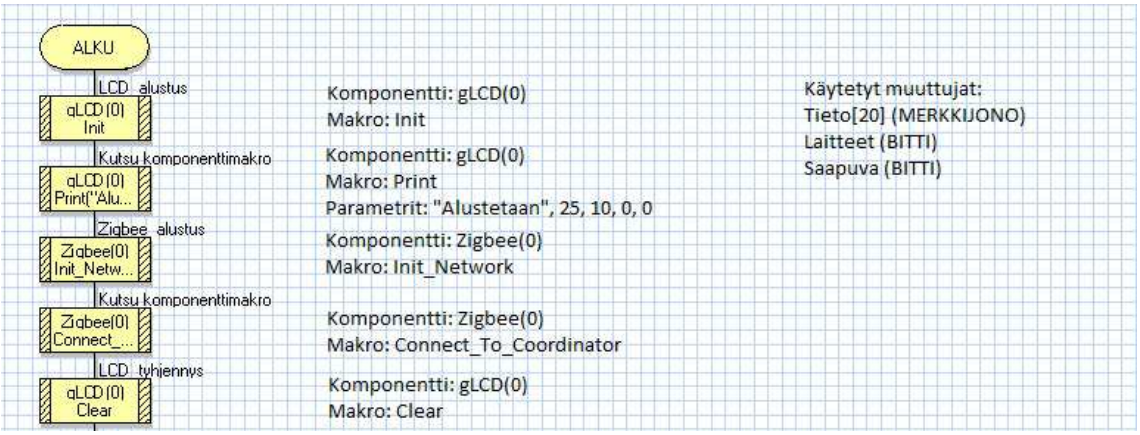
$$(jännite * 30{,}43) - 25{,}81 = kosteusprosentti$$

KAAVA 1. Kosteusprosentin laskeminen

Kaava perustuu kosteussensorin jännitteen muutoksiin, jolloin esitetyllä tavalla kertomalla ja vähentämällä saadaan haluttu tulos. Kaavasta saatava tulos on siis sensorin mittaama ilman suhteellinen kosteus prosentteina.

Kun tarvittava kaava oli saatu selvitetyksi, aloitettiin Flowcodella ohjelman vuokaavion suunnittelu. Ensimmäisenä tehtiin ohjelma koordinaattorille.

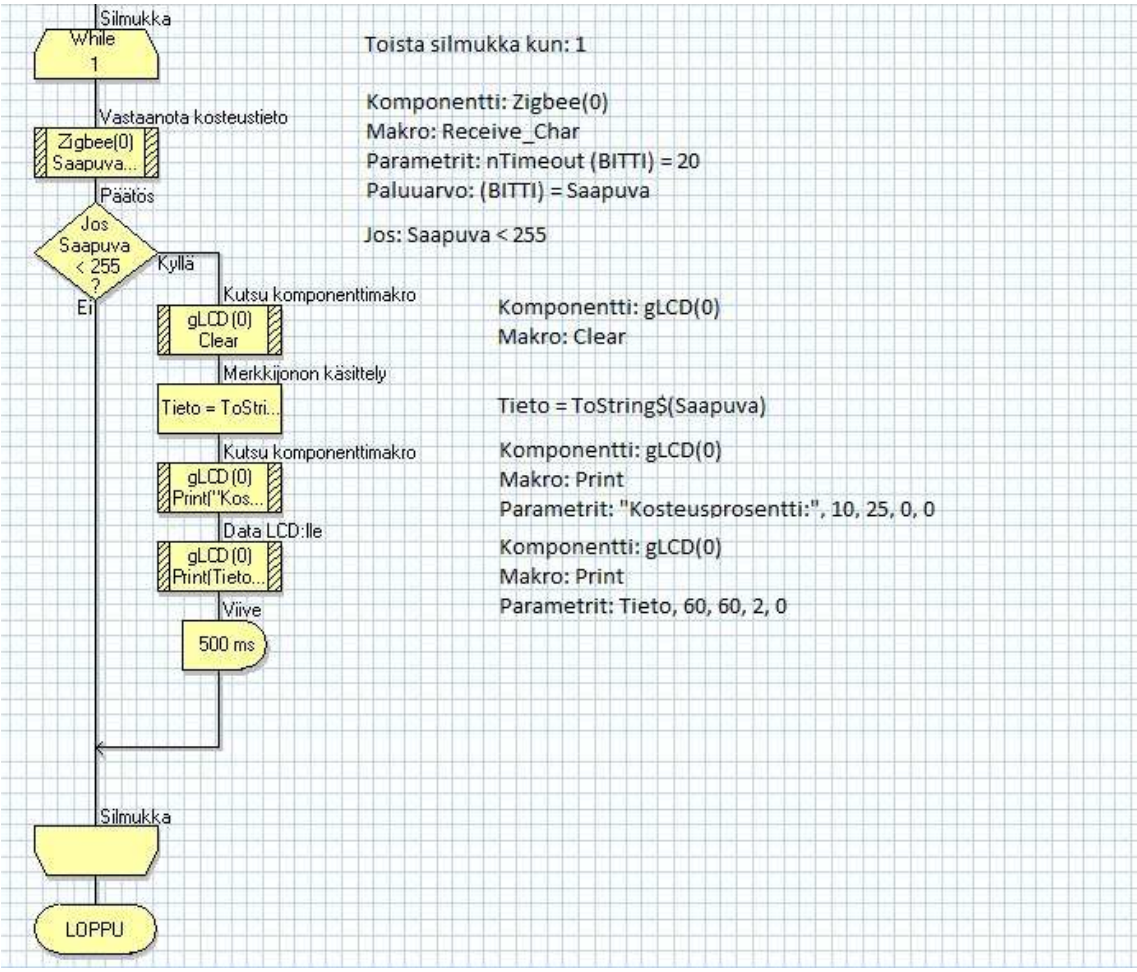
Koordinaattorin tehtävänä oli tiedon vastaanottaminen ZigBee-lohkon avulla ja tiedon tulostaminen gLCD-näytölle. Ensimmäisenä vuokaavioon liitettiin ZigBeen ja gLCD-näytön komponenttimakrot, joiden avulla kyseisten lohkojen toiminnat alustettiin (kuva 26). Alustuksen ajan näytöllä lukee ”Alustetaan”, jonka jälkeen näyttö tyhjennetään. Lohkojen alustuksen jälkeen ohjelma siirtyy silmukkaan, jota suoritetaan loputtomasti.



KUVA 26. Koordinaattorin vuokaavion alkuosa

Ohjelmaa jatkettiin rakentamalla silmukka, jossa tapahtuu ZigBee-komponenttimakroa käyttämällä päätelaitteelta lähetetyn tiedon vastaanotto. Silmukassa suoritataan myös tulostus gLCD-näytölle.

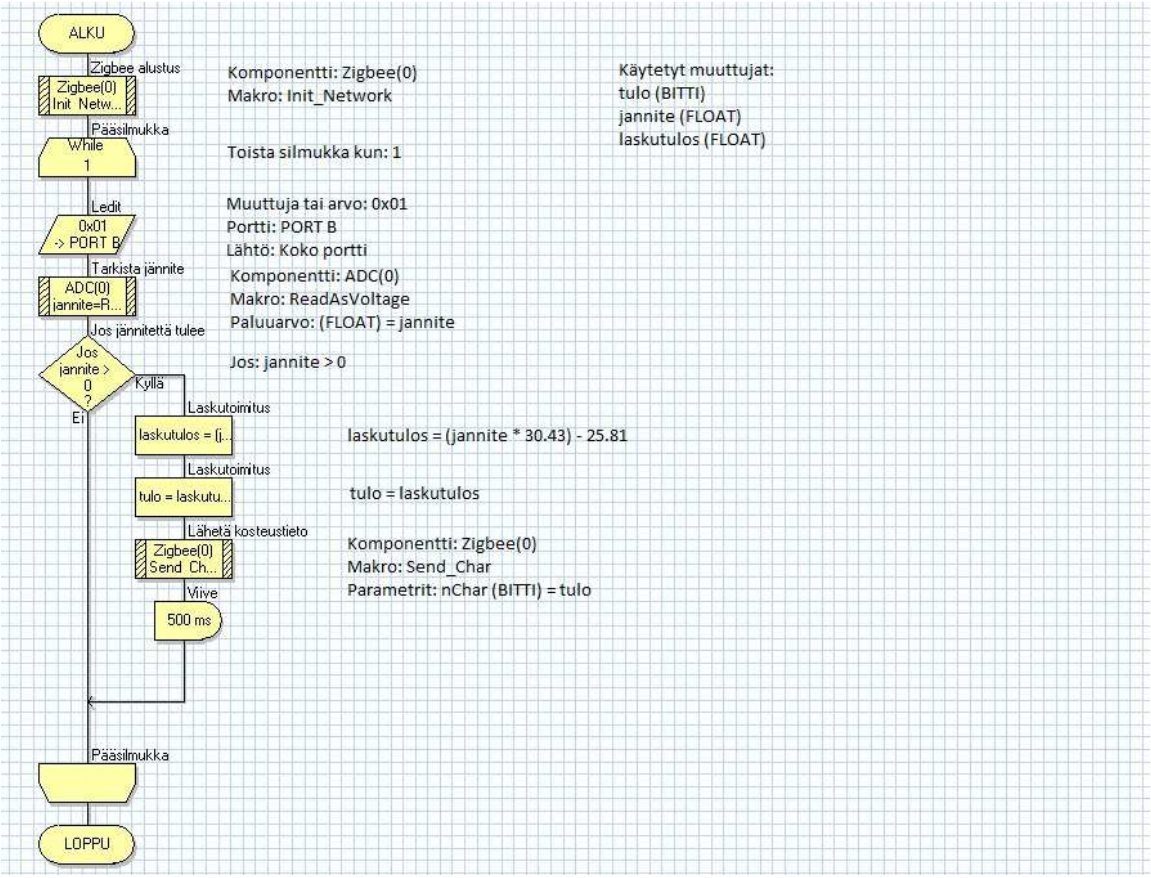
Silmukan ensimmäinen lohko on tiedon vastaanoton hoitava ZigBee-komponenttimakro. Vastaanoton jälkeen tarkastetaan, onko vastaanotetun tavun arvo pienempi kuin 255. Tarkistuksen avulla varmistetaan, että tavu sisältää tietoa eikä näytölle palaudu arvoa nolla. Jos arvo ei ole alle 255, alkaa silmukan suorittaminen alusta. Arvon ollessa pienempi kuin 255 suoritetaan vastaanotetun arvon muuntaminen merkkijonoksi tulostusta varten. Näytölle tulostetaan teksti ”Kosteusprosentti:” ja sen alapuolelle vastaanotettu kosteusarvo. Tämän jälkeen silmukka palautuu taas alkupisteeseen (kuva 27).



KUVA 27. Koordinaattorin vuokaavion loppuosa

Seuraavana vuorossa oli päätelaitteen ohjelma, jonka tehtävänä on hoitaa kosteussensorin antaman jännitearvon lukeminen sekä sen muuntaminen kaavan avulla kosteusprosentiksi (kuva 28). Tämän jälkeen kaavan avulla laskettu tulos (kosteusprosentti) lähetetään ZigBeen avulla koordinaattorille.

Päätelaitteen ohjelma aloitettiin käyttämällä komponenttimakroa ZigBee-lohkon alustamiseen. Alustamisen jälkeen siirrytään ikuiseen silmukkaan. Silmukka alkaa ledin sytyttämisellä, jonka avulla voidaan huomata päätelaitteen siirtyneen silmukan suoritukseen. Tämän jälkeen mitataan sensorilta tuleva jännitearvo. Jos mitään jännitettä ei tule, alkaa silmukan suoritus alusta. Jos taas jännitearvo on yli nolla, vastaanotettu jännite muutetaan kaavan avulla kosteusprosentiksi ja lähetetään käyttäen ZigBee-komponenttimakroa. Tämän jälkeen silmukka alkaa taas alusta.



KUVA 28. Päätelaitteen vuokaavio

Kun kummatkin ohjelmat olivat valmiina, siirrettiin ne omille nodeilleen, jonka jälkeen toimivuutta päästiin testaamaan. Lopputulokseksi saatiin toimiva ohjelma, joka tulostaa kosteusprosentin koordinaattorin näytölle (kuva 29).

KUVA 29. Lopputulos näytöllä

Kosteussensorin toimintaa testattiin vertaamalla sen antamaa arvoa koulun laitehuollosta lainatun yleismittarin antamaan arvoon. Kun sensorin mukaan testauspäivänä kosteusprosentti tietoliikennelaboratoriossa oli 32, yleismittari näytti kosteusprosentiksi 27,5. Kosteussensorin datalehden mukaan tehdaskalibroinnilla sensorin virhe on ±10 %, joten laskukaavan ja ohjelmiston voidaan olettaa toimivan oikealla tavalla.

6.3 Happisensori

Kosteussensorin ohjelman valmistumisen jälkeen siirryttiin tekemään ohjelmaa happisensorille. Tavoite oli siis saada näkymään ilman happipitoisuus koordinaattorin näytöllä päätelaitteen hoitaessa mittauksen.

KUVA 30. Happisensori kytkettynä päätelaitteen sensoriliitäntälohkoon

Happisensorin toimintatapa oli hyvin samankaltainen kuin kosteussensorin. Sensorin ohjelman suunnittelu aloitettiin selvittämällä happisensorin datalehden avulla tarvittavaan kaavaan tarvittavat muuttujat. Muuttujien löytämisen jälkeen voitiin kaava 2 määritellä.

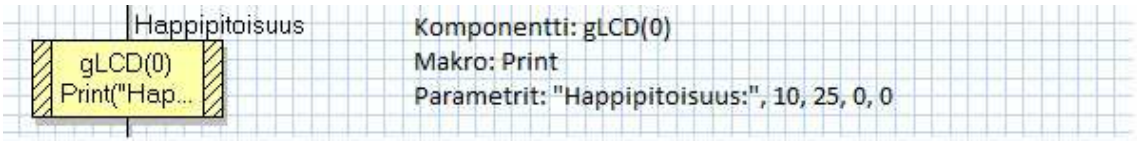
$$6{,}5625 * jännite = happipitoisuus$$

KAAVA 2. Happipitoisuuden määrittäminen

Happipitoisuuden määrittämisen kaava on yksinkertainen. Annettu kiinteä arvo kerrotaan jännitteellä, joka muuttuu sensorin tunnistaman happipitoisuuden mukaan.

Ohjelmien tekeminen aloitettiin koordinaattorin ohjelmalla. Koska koordinaattorin tehtäväksi jää tämänkin sensorin kohdalla vain tuloksen vastaanottaminen ja näytölle tulostaminen, voitiin pohjana käyttää kosteussensorin koordinaattorin ohjelmaa (kuvat 26, 27). Happisensoria varten ohjelmaan tarvitsi muuttaa ainoastaan silmukassa sijaitseva teksti

komponenttimakrosta, joka tulostaa näytölle kosteussensorin tapauksessa ”Kosteusprosentti:”. Happisensorin koordinaattoria varten tämä teksti vaihdettiin tekstiksi ”Happipitoisuus:” (kuva 31). Muuten ohjelma säilyi muuttumattomana.



KUVA 31. Kosteussensorin koordinaattorin ohjelmasta muutettu osa

Myös päätelaitteen ohjelman pohjana pystyttiin käyttämään kosteussensorin päätelaitteessa käytettyä ohjelmaa (kuva 28). Vuokaavion silmukassa oleva laskukaava muutettiin happipitoisuuden mittaukseen tarkoitetuksi kaavaksi, jolloin ohjelma soveltui happisensorin kanssa käytettäväksi (kuva 32). Muulta osin ohjelmaa ei muutettu, vaan se pysyi entisellään.

KUVA 32. Kosteussensorin päätelaitteen ohjelmasta muutettu osa

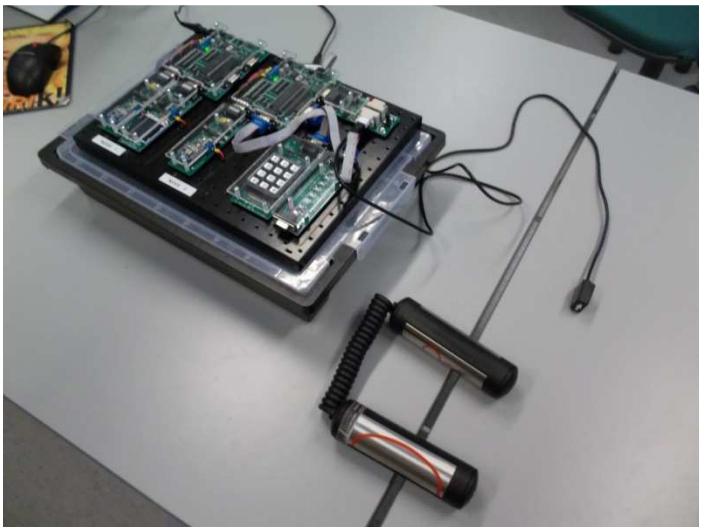
Ohjelmat siirrettiin omille nodeilleen ja kokoonpanoa testattiin. Päätelaite lähetti onnistuneesti mittausdataa koordinaattorille, joka tulosti sen näytölle (kuva 33).

KUVA 33. Happisensorin tulos koordinaattorin näytöllä

Happipitoisuus oli tietoliikennelaboratoriossa sensorin mukaan 18 %. Normaali ilman happipitoisuus on noin 21 %, joten happipitoisuus pitäisi tarkistaa luotettavalla mittarilla, jotta nähtäisiin mahdollinen virhe. Happisensorin datalehden mukaan sensorin vanhetessa näytetyt arvot pienenevät, jolloin kalibrointi olisi tarpeellinen. Koska tarkkaa happipitoisuutta ei soveltuvan mittarin puutteessa tiedetty, ei sensoria voitu kalibroida.

6.4 Sykemittari

Kun happi- ja kosteusantureiden ohjelmat oli saatu toimimaan, siirryttiin tekemään sykemittarin ohjelmaa. Sykemittarin toiminta perustuu jännitepulssin lähettämiseen jokaisen sydämenlyönnin perusteella. Ohjelman tehtäväksi muodostui siis laskea sydämen lyönnit ja verrata sitä kuluneeseen aikaan, jotta saadaan selville BPM (Beats Per Minute) eli kuinka monta kertaa sydän lyö minuutin aikana. Lopullinen tulos siirretään koordinaattorille, jossa tulos esitetään näytöllä. (Kuva 34.)

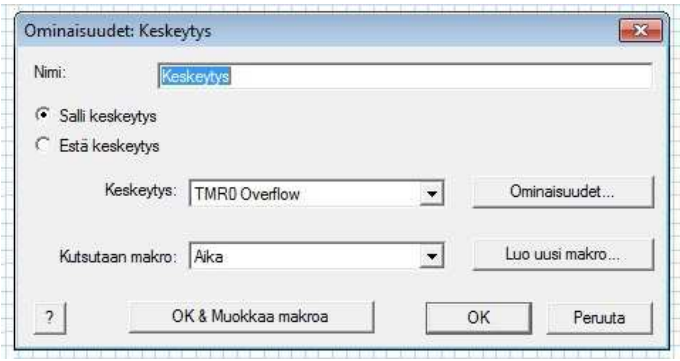


KUVA 34. Sykemittarin kokoonpano

Sykemittarin toimintaa testattiin yksinkertaisella ohjelmalla. Tehty ohjelma vilkutti päätelaitteen LED-lohkon ledejä sykkeen tahtiin. Testillä varmistettiin, että esimerkiksi sykemittarin käsiosan patterit eivät ole loppuneet ja että jännitepulssin vastaanotto toimi oikealla tavalla.

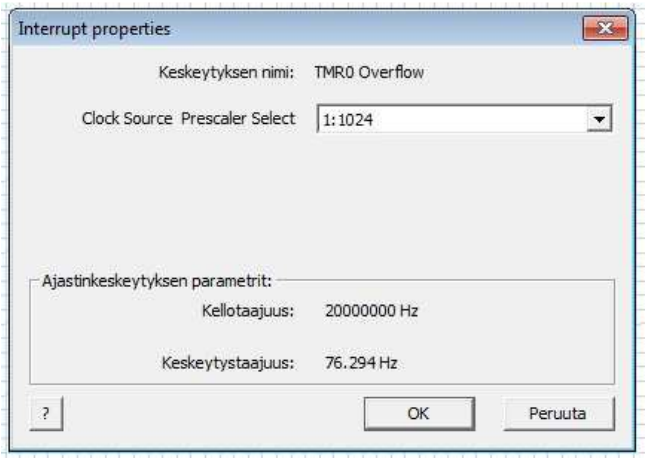
Koska ohjelmassa pitää mitata aikaa, ensimmäisenä selvitettiin kuinka flowcoden avulla voidaan käyttää AVR:n Timeriä, jonka avulla ajan mittaus on mahdollista. Timer ei vaikuta pääohjelman toimintaan, koska se toimii omana makronaan aliohjelmassa. Sen toiminta perustuu keskeytyksiin, jotka muodostetaan oskillaattorin taajuuden avulla. Oskillaattori toimii käytettävässä kokoonpanossa 20 MHz:n taajuudella. Ajan mittausta varten löytyi Matrix Multimedian sivuilta esimerkki, jota apuna käyttämällä saatiin sekunnin mittaava aliohjelma. (21.)

Ajanmittausohjelman tekeminen aloitettiin valitsemalla vuokaavion lohkoksi Keskeytys-makro, jonka liukuvalikosta valitaan keskeytyksen tyypiksi ”TMR0 overflow”. (Kuva 35.)



KUVA 35. Keskeytys-makron asetukset

Ominaisuudet-valikosta valitaan esiskaalaimen asetukseksi 1:1024, jolloin 20 MHz:n oskillaattoritaajuudella keskeytystaajuus on 76,294 Hz. (Kuva 36.)



KUVA 36. Esiskaalaimen asetukset

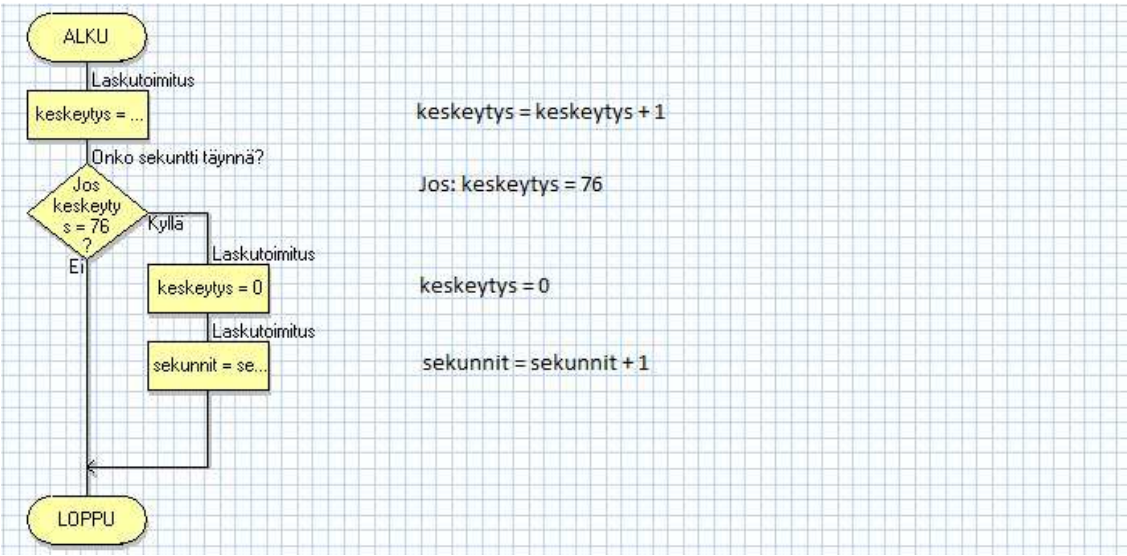
Keskeytysohjelmaa varten tulos pyöristetään, joten käytettävä lukema on 76. Pyöristyksen takia saatava mittausaika ei ole aivan tasan sekunti, vaan jää hieman tämän alle. Tarkka tulos saadaan kaavasta 3, jossa s on sekunteja.

$$s = \frac{76}{76{,}294\ Hz} \approx 0{,}99614$$

KAAVA 3. Todellinen mittausaika

Kuten nähdään, virhe on todella pieni, joten sillä ei ole ohjelman suorituksen kannalta merkitystä.

Kun tarvittavien keskeytysten määrä oli saatu laskettua, tehtiin kuvan mukainen aliohjelma nimeltä ”Aika”. Ohjelma toimii siten, että keskeytys-muuttujaan lisätään aina yksi kun keskeytys tulee. Kun keskeytyksiä on kertynyt tarvittava määrä eli 76, nollataan keskeytysten määrä ja merkitään sekunti kuluneeksi. Aliohjelma pyörii siis ikuisesti mitaten aikaa.



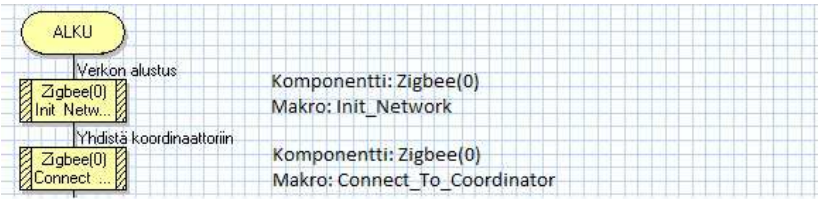
KUVA 37. Aliohjelma ”Aika” joka mittaa sekunnin

Itse pääohjelman tekeminen aloitettiin päätelaitteen ohjelmasta. Päätelaitteen tehtävänä oli mitata tulevat jännitepulssit ja muuttaa ne BPM-lukemaksi. BPM-lukeman saamiseksi tarvittiin laskutoimitus, joten seuraavana alettiinkin selvittää, minkälaista kaavaa laskutoimituksessa tulisi käyttää. Kaavaksi muodostui kaavan 4 mukainen.

$$\frac{60 * pulssit}{sekunnit} = BPM$$

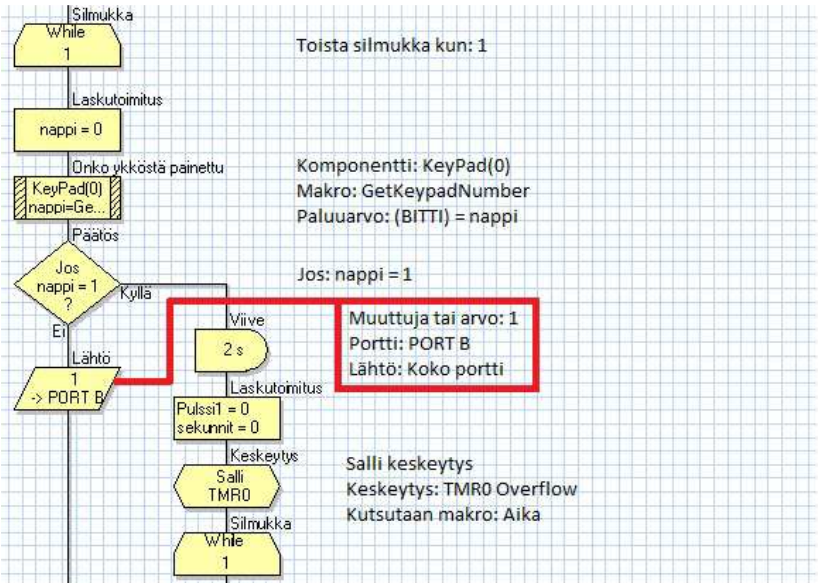
KAAVA 4. Kaava sykkeen mittaukselle

Kaavassa 4 kerrotaan sykemittarilta saadut jännitepulssit kuudellakymmenellä, ja tulos jaetaan kuluneilla sekunneilla. Vastaukseksi saadaan syke per minuutti eli BPM. Yhdellä mittauksella ei voi saada tarpeeksi tarkkaa arviota sykkeestä, koska ajan yksikkönä oli kokonainen sekunti, joten mittausajan pitää olla tarpeeksi pitkä. Mittausväleiksi päätettiin ottaa viisi sekuntia, paitsi viimeinen mittaus, jonka väli edelliseen oli neljä sekuntia. Kokonaisuudessaan sekunteja laskettiin kahteenkymmeneen asti, jonka jälkeen sekunnit nollattiin ja niiden laskeminen alkaa alusta. Jokaisen mittausvälin kohdalla lasketaan BPM kaavan 4 avulla, ja tulosta verrataan edellisen mittausvälin tulokseen laskemalla lukujen keskiarvo. Pääohjelma koostuu kokonaisuudessaan kolmesta osasta. Aluksi tehtdään ZigBee-lohkon alustus ja yhdistetään koordinaattoriin (kuva 38).



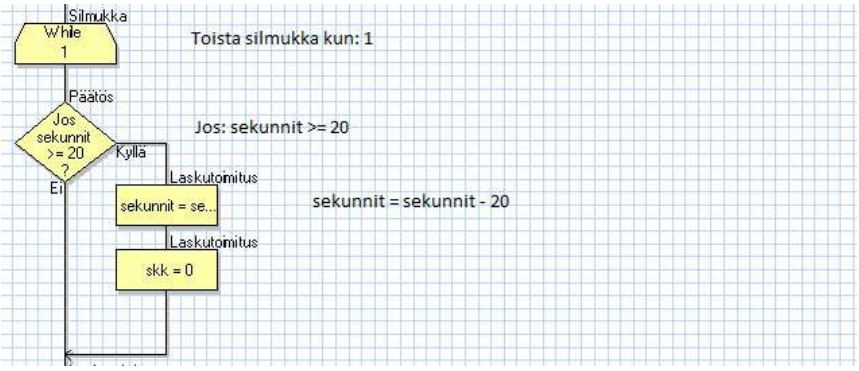
KUVA 38. Päätelaitteen pääohjelman alku

Tämän jälkeen ohjelma menee silmukkaan, jossa odotetaan näppäimistön napin ”1” painallusta. Ulospäin näkee ohjelman odottavan napin painallusta, kun yksi ledi palaa LED-lohkossa. Kun nappia ”1” painetaan, kahden sekunnin viiveen jälkeen ohjelmassa käytettäviä muuttujat nollataan. Tämän jälkeen timer käynnistetään sallimalla keskeytykset, eli aliohjelma ”aika” alkaa laskemaan sekunteja taustalla. (Kuva 39.)



KUVA 39. Pääohjelman ensimmäinen silmukka

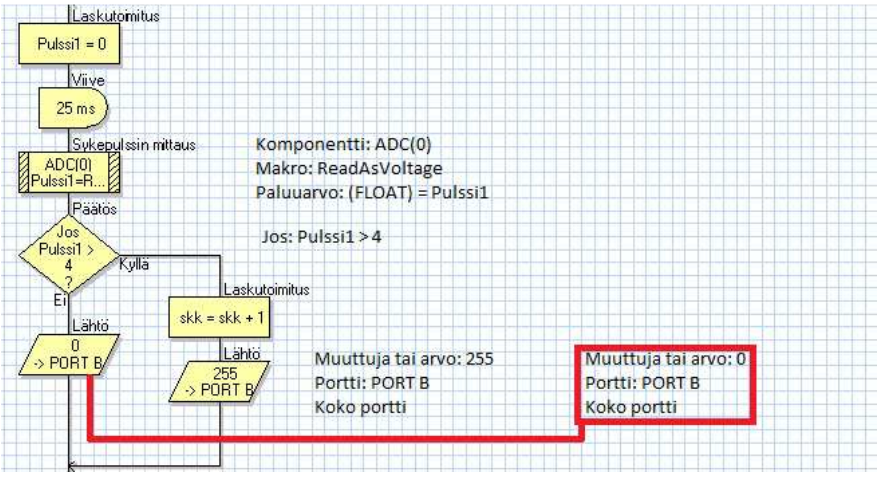
Pääohjelman sykettä mittaavan silmukan toiminta alkaa kuluneitten sekuntien mittaamisella. Kun sekunteja on kertynyt kaksikymmentä, nollataan sekunnit ja laskeminen alkaa taas alusta. Samalla nollataan myös arvo ”skk”, joka on kahdenkymmenen sekunnin aikana rekisteröityjen sydämenlyöntien lukumäärä. (Kuva 40.)



KUVA 40. Mittaussilmukan alku

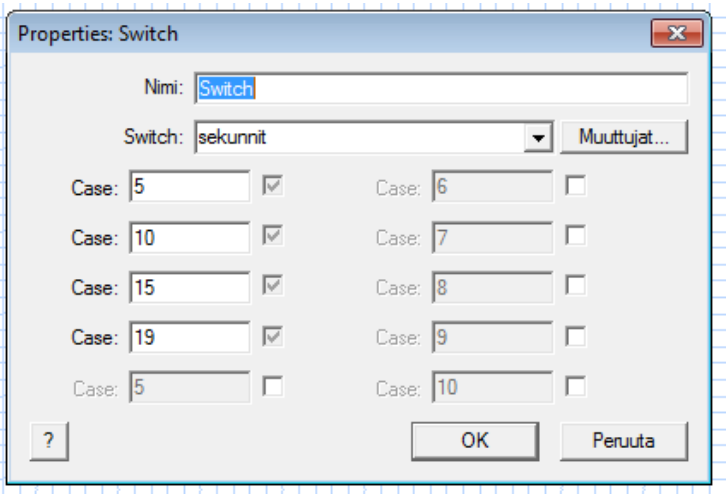
Seuraava osa silmukassa on sykkeen mittaus jännitepulssien avulla. Aluksi nollataan luettavan pulssin muuttujan arvo, jotta uusi mittaus voidaan suorittaa. Ennen itse mahdollisen tulevan jännitepulssin mittausta on 25 millisekunnin viive. Viive piti laittaa ennen mittausta, koska muuten laskuri rekisteröi ylimääräisiä pulsseja. Viiveen jälkeen tarkastettiin sensorilohkolta tulevaa

jännitettä. Jos jännitepulssi tunnistettiin, arvo ”skk” suurenee yhdellä, ja ledit sytytetään. Jos pulssia ei tule, pysyvät ledit sammuksissa. (Kuva 41.)



KUVA 41. Jännitepulssin mittaus

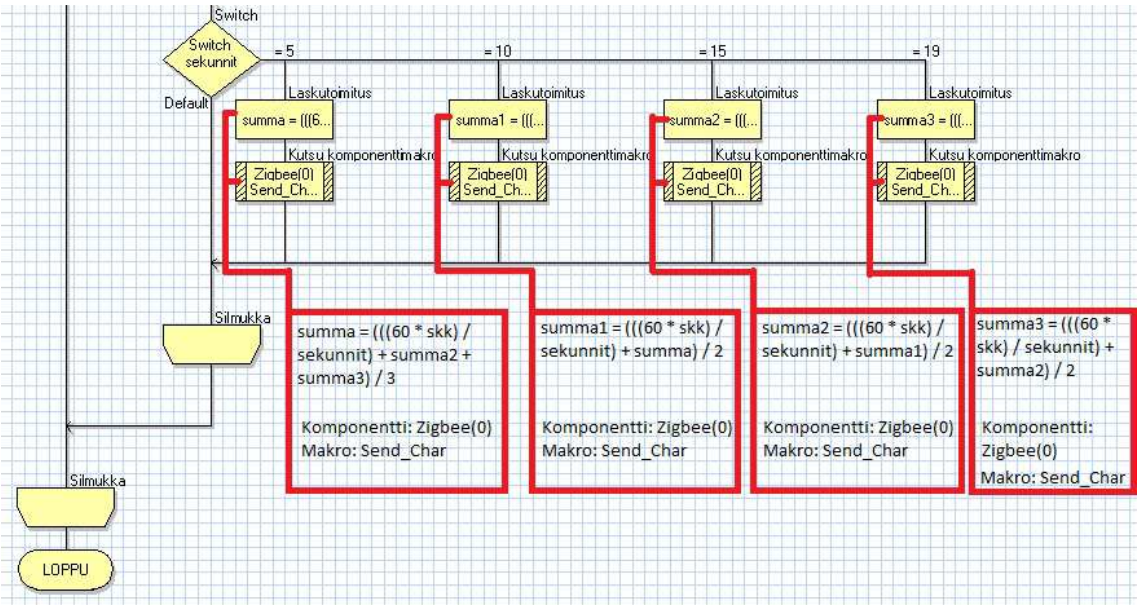
Ohjelman loppuosassa lasketaan mitattujen jännitepulssien määrää sekä aliohjelmasta saatuja sekunteja käyttämällä BPM-arvo, joka lähetetään koordinaattorille. BPM arvo lasketaan, kun sekunteja on kulunut 5, 10, 15, ja 19. Laskuhetken valinta tehdään Switch-ohjelmalohkon avulla. Switch-lohkoon voidaan määrittää arvot joiden toteutuessa sille suunniteltu toiminto toteutuu. (Kuva 42.)



KUVA 42. Switch-lohkon asetukset

Halutut BPM-arvon laskukohdat asetettiin kohtiin ”Case:”. Muuttujana oli sekunti, jonka arvon perusteella valitaan haluttu tehtävä. Jokaisella

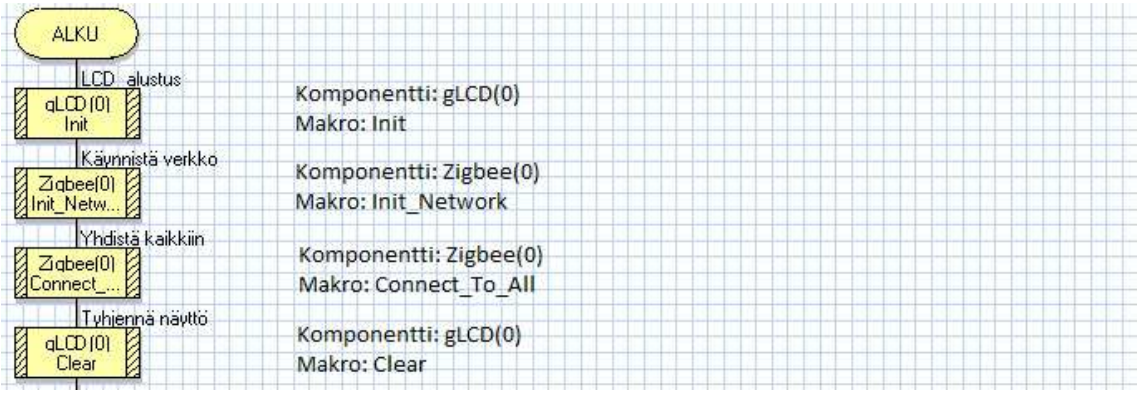
laskuhetkellä suoritetaan laskukaava, jossa kaavaa 4 käyttäen lasketaan BPM. Rekisteröidyt laskentapulssit saadaan muuttujasta ”skk”. Samalla myös tulos keskiarvoistetaan vertaamalla sitä edellisiin tuloksiin. Kun BPM on laskettu, lähetetään se koordinaattorille. (Kuva 43.)



KUVA 43. Ohjelman loppuosa

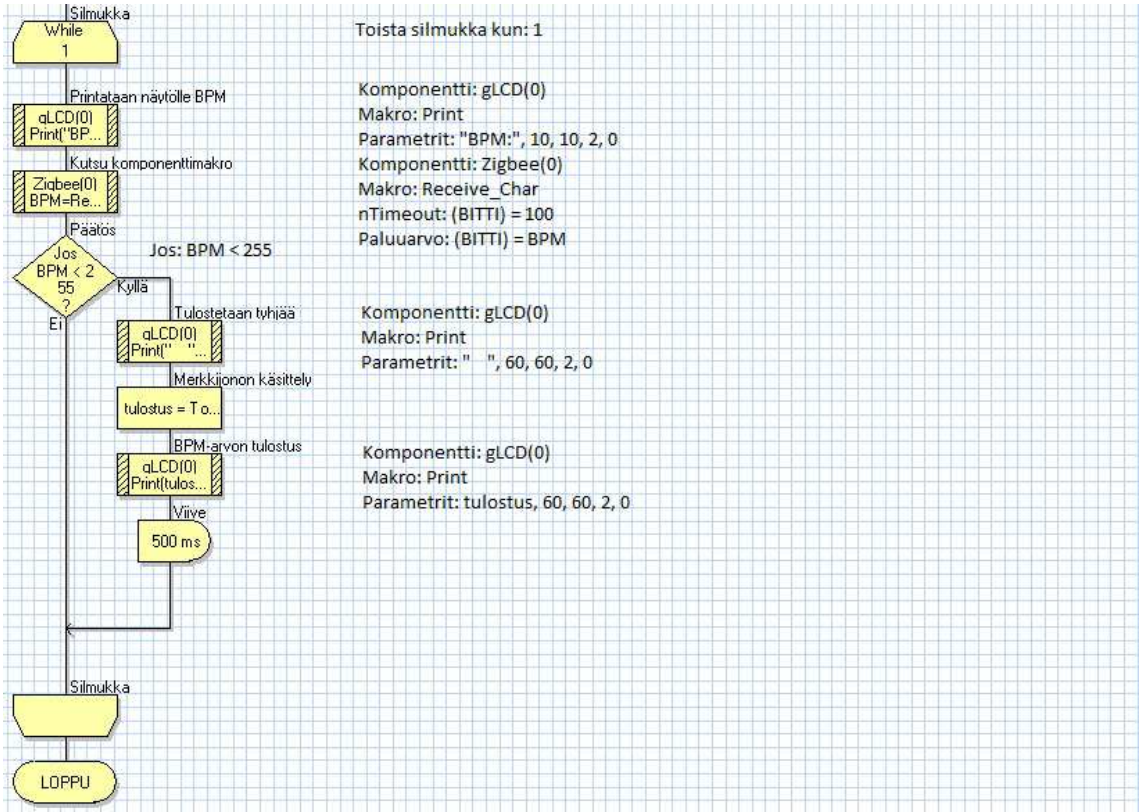
Ohjelman suoritus jatkuu viimeisessä silmukassa niin kauan kunnes päätelaite käynnistetään uudelleen tai resetoidaan.

Päätelaitteen ohjelman jälkeen suunniteltiin koordinaattorin ohjelma. Koordinaattorin ohjelman tarkoituksena oli ottaa BPM-arvo vastaan ja tulostaa se gLCD-näytölle. Ohjelma aloitettiin alkuasetusten määrittelyllä. Alussa alustetaan gLCD- ja ZigBeelohkot, otetaan yhteys päätelaitteeseen ja tyhjennetään gLCD-näyttö. (Kuva 44.)



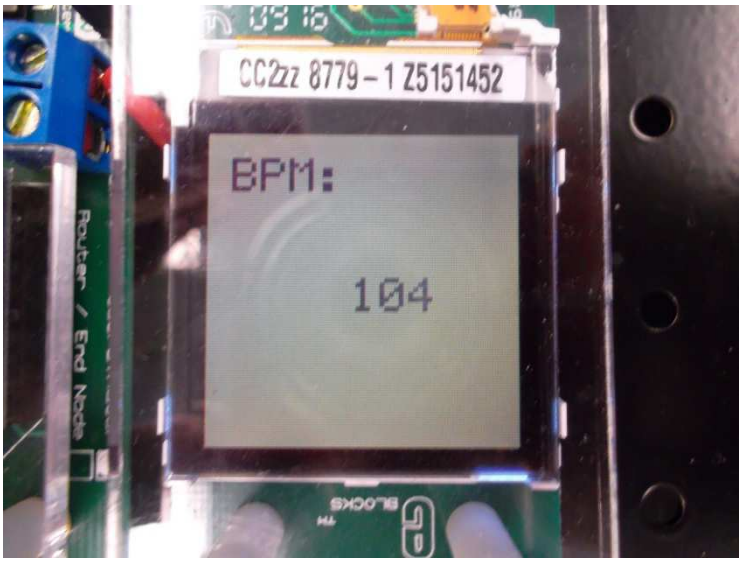
KUVA 44. Koordinaattorin ohjelman alkuosa

Kun alkuasetukset on tehty, ohjelma siirtyy silmukkaan, jossa BPM-arvoa otetaan vastaan päätelaitteelta. Silmukan alussa näytölle tulostetaan kiinteänä pysyvä merkkijono ”BPM:”. Tämän jälkeen otetaan vastaan päätelaitteelta tuleva BPM-arvo. Jos vastaanotettava arvo on saatavilla, eli vastaanotettavan tavun arvo on alle 255, siirrytään tulostukseen. Muussa tapauksessa silmukka aloitetaan alusta. Arvo tulostetaan koko mittauksen ajan kiinteänä pysyvän ”BPM:”-merkkijonon alle. (Kuva 45.)



KUVA 45. Koordinaattorin ohjelman loppuosa

Kun molemmat ohjelmat olivat saatu valmiksi, lähetettiin ne omille nodeilleen. Lopullinen ohjelma toimii odotetunlaisesti. Päätelaite lähettää syketiedon koordinaattorille, ja tieto tulostuu näytölle (kuva 46).



KUVA 46. BPM-arvo koordinaattorin näytöllä

Ohjelmalla saatiin ulostuloksi järkeviä BPM-lukemia. Lukema nousi loogisesti, jos esimerkiksi mittauksen aikana tai ennen sitä oli tehnyt jotain sykettä nostavaa, kuten hyppinyt paikallaan tai juossut ympäri luokkaa. Myös eri henkilöillä oli erilainen lepopulssi, ja kokonaisuudessaan ohjelman antamat lukemat tuntuivat järkeviltä.

7 POHDINTA

Työn tarkoituksena oli valita kolme erityyppistä sensoria ja ottaa ne käyttöön E-Blocks-ympäristössä. Sensoreiksi valikoituivat happi- ja kosteussensorit sekä sykemittari. Sensoreille suunniteltiin omat ohjelmansa, jotka lähettivät ZigBeetä käyttäen päätelaitteelta sensoritiedon koordinaattorille. Koodinaattorin tehtävänä oli tulostaa sensoritieto gLCD-näytölle. Lopputulokseksi saatiin kolme erilaista E-Blocksissa toimivaa sensorisovellusta.

Kosteussensorisovelluksessa koordinaattorille lähetetään päätelaitteelta kosteussensorin mittaama ilman suhteellinen kosteusprosentti. Happisensorisovelluksessa lähetettävänä tietona on ympäröivän ilman happipitoisuus. Viimeisenä sovelluksena on sykemittari, jossa lasketaan mittarin rekisteröimät sydämenlyönnit ja tulostetaan koordinaattorin näytölle BPM-lukema.

Työn alussa testiohjelmaa ajaessa huomattiin että lähetetty tieto ei aina saapunut koordinaattorille. Kun asiaa tutkittiin Zena-verkkoanalysaattorin avulla huomattiin, että aluksi käytetyillä kanavilla esiintyi muutakin liikennettä. ZigBeen käyttämää 2,4 GHz:n taajuutta käyttävät myös muutkin langattomat tekniikat, kuten esimerkiksi WLAN. Näin voitiin siis pitää mahdollisena, että koulun WLAN aiheutti häiriöitä yhteyteen. WLANin tukiasemia sijaitsee laboratoriosiiven käytävällä, joten niiden aiheuttamien häiriöiden mahdollisuus oli suuri.

Kosteussensorin ohjelma saatiin toimimaan ilman suurempia ongelmia. Aluksi sensorin hidas reagointi kosteuden muutoksiin ihmetytti, mutta lisäämällä viiveitä lähetykseen ja vastaanottoon saatiin reagointinopeutta kasvatettua. Jatkokehityksenä suhteellisen epätarkan mittaustuloksen (virhe ±10 %) tarkentamiseksi voidaan suorittaa kalibrointi. Koska laitehuollosta lainatun yleismittarin kalibroinnista ei voitu olla varma, ei sen antaman arvon perusteella alettu kalibrointia suorittamaan. Toinen mahdollisuus kalibrointiin olisi ollut erilaisten suolojen avulla, mutta tähän ei ollut mahdollisuutta.

Happisensorin antaman lukeman oikeellisuudelle ei saatu täyttä varmuutta. Sensori antoi lukemaksi 18 %, mutta normaalisti ilman happipitoisuus on noin 21 %. Luultavasti sensori oli jo sen verran vanhentunut, että kalibrointia olisi tarvittu. Koska ilman happipitoisuuden muutokset eivät ole kovin suuria, sensorin ohjelmaa voisi jatkokehittää tarkentamalla mitattavaa lukua yhdellä desimaalilla. Yhden desimaalin lisäys mittaustarkkuuteen aiheuttaa sen, että mittaustietoa ei enää voida lähettää pelkästään yhdellä tavulla. Lähetettävä data pitäisi jakaa kolmeksi tavuksi, joista kolmas varmistaa, että vastaanotetut tavut ovat oikeat.

Sykemittarin ohjelman kehityksessä esiintyi aluksi pieniä ongelmia. Jostain syystä sensorilohkolta tulevien jännitepulssien luku ei onnistunut oikealla tavalla, vaan laskuri rekisteröi runsaasti ylimääräisiä pulsseja. Ongelma saatiin korjattua lisäämällä 25 millisekunnin viive ennen sensorilohkon lukua. Tätä pienemmällä viiveellä ylimääräisiä pulsseja rekisteröityi. Koska viive ei kuitenkaan ole kovin suuri, voidaan olettaa, että sillä ei ole suurta merkitystä mittauksen kokonaistulokseen. Myös muita ongelmia ja puutteita sykemittarin ohjelman lopputarkastelussa huomattiin. Päivitysnopeus sykkeen muutokselle on melko hidas. Päätelaitteen ohjelmaan lisättävän viiveen avulla päivitysnopeutta voisi mahdollisesti kasvattaa, mutta ylimääräistä viivettä ei voitu laittaa, koska se olisi varmasti vaikuttanut mittaustulokseen.

Koko työn jatkokehityksenä voitaisiin tehdä sensoriverkko, jossa jokainen sensorisovellus toimii omana nodenaan ja lähettää koordinaattorille mittaustietonsa. Mittaustulokset tulostettaisiin koordinaattorin näytölle omiin sarakkeisiinsa yhtäaikaisesti.

Oppimiskokemuksena työ oli hyvin antoisa. Ohjelmointitaitoni eivät koskaan ole olleet perusteita kummemmat, joten alku tuntui tuskastuttavalta. Kuitenkin Flowcoden avulla pieniä testiohjelmia tekemällä pääsi nopeasti jyvälle siitä, mitä ominaisuuksia tarvittavat ohjelmat vaatisivat. Myös Matrix Multimedian kotisivuilla oli paljon opettavaisia esimerkkejä. Loppujen lopuksi pystyin itse melko hyvin jo päättelemään mitä lohkoja ohjelmien vuokaaviot tarvitsisivat. Työn lopussa pystyin jo miettimään vaihtoehtoisia ratkaisuja ohjelmien tekemiselle.

LÄHTEET

1. Rantala, Jyri. 2006/2007. Sensoriverkot älykkäissä ympäristöissä. Saatavissa: http://www.ele.tut.fi/teaching/ele-7100/vuosi06-07/harjoitustyot/Rantala_Sensoriverkot.pdf. Hakupäivä 15.5.2012.

2. SenseNode - Wireless Sensor Node. 2012. Genetlab Bilgi Teknolojileri San. Ve Tic. A.S. Saatavissa: http://www.defence-industries.com/contractors/army/base-camp-protection/genet/. Hakupäivä 16.5.2012.

3. Wireless sensor networks and embedded devices. 2012. VTT. Saatavissa: http://www.vtt.fi/research/technology/wireless_sensor_networks_and_embedded_devices.jsp?lang=en. Hakupäivä 16.5.2012.

4. Related technologies. 2012. Purelink Technology Inc. Saatavissa: http://www.purelink.ca/en/technologies/related-technologies.php. Hakupäivä: 16.5.2012.

5. ZigBee Technology Tutorial. 2012. Adrio Communications Ltd. Saatavissa: http://www.radio-electronics.com/info/wireless/zigbee/zigbee.php. Hakupäivä 18.4.2012.

6. ZigBee Student Notes. 2009. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/LearningCentre/EB538-82-03.pdf. Hakupäivä 18.4.2012. Hakupäivä 18.4.2012.

7. IEEE 802.15.4 Technology & Standard. 2012. Adrio Communications Ltd. Saatavissa: http://www.radio-electronics.com/info/wireless/ieee-802-15-4/wireless-standard-technology.php. Hakupäivä 19.4.2012.

8. Aaltonen, Tapio. 2009. Hajaspektritekniikat. 2009. Saatavissa: https://jop.cs.tut.fi/twiki/bin/view/TLTT/Hajaspektri. Hakupäivä 14.5.2012.

9. Basic Architecture. 2007. Jennic Ltd. Saatavissa: http://www.jennic.com/elearning/zigbee/files/html/module3/module3-2.htm. Hakupäivä 19.4.2012.

10. About. 2012. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/about.php. Hakupäivä 8.5.2012.

11. E-blocks. 2012. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/eblocks.php. Hakupäivä 8.5.2012.

12. ZigBee board datasheet. 2008. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/datasheets/EB051-30-1.pdf. Hakupäivä 9.5.2012.

13. Sensor board datasheet. 2005. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/resources/files/datasheets/EB003-30-2.pdf. Hakupäivä 9.5.2012.

14. Flowcode. 2012. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/flowcode.php. Hakupäivä 20.5.2012.

15. Atmel AVR. 2012. Wikipedia. Saatavissa: http://de.wikipedia.org/wiki/Atmel_AVR. Hakupäivä 11.5.2012.

16. Relative Humidity Sensor. 2012. Vernier. Saatavissa: http://www.vernier.com/products/sensors/rh-bta/. Hakupäivä 28.5.2012.

17. O2 Gas Sensor. 2012. Vernier. Saatavissa: http://www.vernier.com/products/sensors/o2-bta/. Hakupäivä 28.5.2012.

18. Hand-Grip Heart Rate Monitor. 2012. Vernier. Saatavissa: http://www.vernier.com/products/sensors/hgh-bta/. Hakupäivä 28.5.2012.

19. ZigBee Instructor Guide. 2009. Matrix Multimedia. Saatavissa: http://www.matrixmultimedia.com/LearningCentre/EB538-82-03.pdf. Hakupäivä 28.5.2012.

20. Desbonnet, Jon. 2011. Using the Microchip ZENA ZigBee/802.15.4 network analyzer with Linux. Saatavissa: http://jdesbonnet.blogspot.com/2011/02/using-microchip-zena-zigbee802154.html. Hakupäivä 28.5.2012.

21. Rowland, Ben. 2009. Measuring Time with Interrupts. Saatavissa: http://www.matrixmultimedia.com/resources/files/articles/MX016%20-%20Measuring%20Time%20with%20Interrupts.pdf. Hakupäivä 28.5.2012.