Joni Lampio

EKG-anturin datan välitys iPhonelle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

24.5.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Joni Lampio EKG-anturin datan välitys iPhonelle 47 sivua 24.5.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	sulautettu tietotekniikka
Ohjaajat	projektiasiantuntija Dmitri Vorobiev koulutuspäällikkö Anssi Ikonen

Insinöörityön tavoitteena oli kehittää Nordic Semiconductorin nRF51822-piiriin pohjautuva järjestelmä, joka selvittää EKG-anturilta saadusta datasta sydämen lyöntitiheyden ja lähettää sen Bluetooth low energyn välityksellä iPhonelle, jossa se esitetään tarkoitusta varten luodulla sovelluksella. Insinöörityö toteutettiin osana Metropolia Ammattikorkeakoulun tutkimus- ja kehitysyksikkö Electrian ja Tampereen teknillisen yliopiston HealthSens-projektia.

Työhön valittiin käytettäväksi nRF51822-piirin sisältävä nRF51822 development kit -moduuli sekä nRF6310-kehitysalusta, joiden valintaan päädyttiin niiden ominaisuuksien perusteella. nRF51822-piirin ohjelmistokehitykseen käytettiin avoimen lähdekoodin työkaluja, joiden valintaan vaikuttivat projektiin osallistuneiden henkilöiden aikaisemmat positiiviset kokemukset kyseisistä työkaluista.

Ennen varsinaisen järjestelmän kehittämistä tutustuttiin työssä käytetyn piirin ominaisuuksiin sekä muihin työn kannalta olennaisiin aiheisiin kuten Bluetooth low energyyn ja elektrokardiografiaan, joka on sydämen sähköistä toimintaa selvittävä tutkimus.

nRF51822-piirin ohjelmiston kehityksessä käytettiin hyväksi Nordic Semiconductorin laitekirjastoja sekä EP Limitedin QRS-kompleksin tunnistinta, joiden avulla EKG-anturin datasta saatiin selvitettyä QRS-kompleksien paikat, ja niiden avulla laskemaan sydämen lyöntitiheys.

iPhonelle kehitetty sovellus pohjautui Nordic Semiconductorin nRFready iOS demo app -sovellukseen, jolle luotiin uusi yksinkertaistettu käyttöliittymä. Sovelluksen kehitys toteutettiin Applen Xcode-kehitysympäristöllä.

Työn lopputuloksena syntyi sydämen lyöntitiheyden selvittävä järjestelmä, joka esitti sydämen lyöntitiheyden iPhone-sovelluksessa.

Avainsanat	Sydänsähkökäyrä, lyöntitiheys

Author	Joni Lampio
Title	Forwarding ECG sensor data to iPhone
Number of Pages	47 pages
Date	24 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Embedded Engineering
Instructors	Dmitri Vorobiev, Project Expert Anssi Ikonen, Head of Information Technology Degree Programme

The purpose of this bachelor's thesis was to create an nRF51822-chip based system which calculates heart rate from ECG signal and sends it to iPhone 5 via Bluetooth low energy. For showing the heart rate in iPhone it was also necessary to develop an iPhone application. This bachelor's thesis was undertaken as part of a project called HealthSens carried out in cooperation between Electria, a research and development unit in the Helsinki Metropolia University of Applied Sciences, and Tampere University of Technology.

The nRF51822 development kit module and nRF6310 development platform were chosen for this project because of their features. The software for them was developed with open source tools, such as Vim and GCC.

The system developed used Nordic Semicondutor's device library and analog to digital converter for the sampling of signal and EP Limited's QRS complex detector for detecting QRS complexes. After detection of QRS complexes, heart rate was calculated from QRS complexes and sent to iPhone.

The iPhone application was based on the nRFready iOS demo application and only a new simplified graphical user interface was developed for it. Xcode IDE was used for iPhone application development.

In conclusion, the project was successful and the system and iPhone application met the expectations.

Keywords	Bluetooth low energy, Electrocardiography, Heart rate

Sisällys

1 Johdanto

Electria on Metropolia Ammattikorkeakoulun tutkimus- ja kehitysyksikkö, joka on keskittynyt erityisesti erittäin vähävirtaisiin langattomiin ratkaisuihin. Electrialla alkoi syksyllä 2012 Tampereen teknillisen yliopiston kanssa kaksivuotinen Tekesin rahoittama HealthSens-projekti, joka pyrkii helpottamaan terveydenhuoltoalan alati pahenevaa resurssipulaa.

Projektin tavoitteena on kehittää iholle kiinnitettävä useasta laastarin kaltaisesta langattomasta anturista koostuva anturijärjestelmä, joka välittää antureilta kerätyn datan langattoman Bluetooth low energy -tiedonsiirtotekniikan avulla mobiililaitteeseen, josta se välitetään internetissä sijaitsevaan pilvipalveluun terveydenhuoltohenkilökunnan hyödynnettäväksi.

HealthSens-projektiin Electria on valinnut käytettäväksi Nordic Semiconductorin nRF51822-piirin, joka sisältää Bluetooth low energy yhteensopivan radion, jolla tieto voidaan välittää mobiililaitteelle. Projektia varten Electria on myös hankittu nRF51822-piirin pohjautuvia kehitysalustoja.

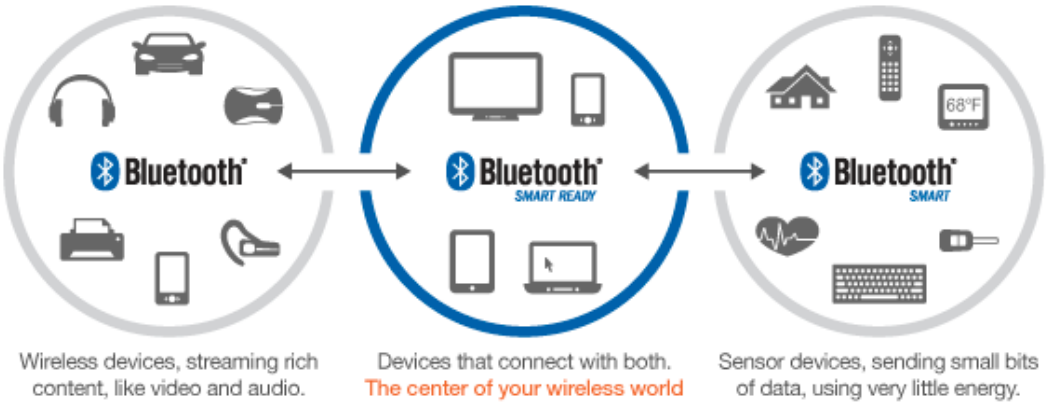
Insinöörityössä käsitellään projektin osaa, jossa analogiselta EKG-anturilta saatu signaali on tarkoitus välittää Bluetooth low energyn välityksellä mobiililaitteelle. Insinöörityön aikana kehitettävässä järjestelmässä EKG-anturilta saatu signaali on tarkoitus esikäsitellä nRF51822-piirillä siten, että signaalista saadaan selville sydämen lyöntitiheys. Esikäsittelyn jälkeen tieto on tarkoitus välittää Bluetooth Low energyn avulla iPhone 5:lle, jossa se esitetään sitä varten luodulla sovelluksella.

Insinöörityöraportissa perehdytään insinöörityön toteuttamisen kannalta olennaisiin aiheisiin, kuten Bluetooth low energyyn sekä elektrokardiografiaan. Teoriapohjan tarjoamisen jälkeen insinöörityöraportissa esitellään käytössä oleva laitteisto sekä ohjelmistot, jonka jälkeen siirrytään mikrokontrollerin ohjelmiston sekä iPhonen sovellusohjelman käsittelyyn. Raportti päätetään järjestelmän testaukseen sekä yhteenvetoon.

2 Bluetooth low energy

Bluetooth low energy on Bluetooth 4.0 standardissa esitelty uusi aikaisempaa vähävirtaisempi langaton tekniikka, jota käyttävät laitteet voivat toimia jopa vuosia yhdellä nappiparistolla. Se mahdollistaa Bluetoothin leviämisen sellaisiin laitteisiin, joihin aikaisemmat Bluetooth-teknologiat eivät ennen soveltuneet suuren virrankulutuksensa vuoksi. Tällaisia laitteita ovat esimerkiksi erilaiset langattomat mittausjärjestelmät ja kaukosäätimet, joiden ei tarvitse lähettää suuria määriä dataa. [1.]

Tekniikan vähävirtaisuus perustuu radion pieneen pulssisuhteeseen (duty cycle), joka on seurausta siitä, että pieni määrä tietoa lähetetään mahdollisimman pienellä viiveellä nopeudella 1 Mb/s, jolloin radio ehtii olemaan lepotilassa pidemmän ajan ennen seuraavaa tiedonsiirtoa [2, s. 16–18]. Esimerkkinä, jos tiedonsiirto tapahtuu kerran sekunnissa ja se vie 3 ms, radio ehtii nukkumaan ajasta jopa 99.7 % [3]. Suuremmilla tietomäärillä radio ei ehdi olemaan lepotilassa yhtä pitkää aikaa, eli pulssisuhde kasvaa, jolloin myös virrankulutus kasvaa merkittävästi. Tästä syystä vanhemmat niin sanotut klassiset Bluetooth-tekniikat soveltuvat paremmin suurien tietomäärien siirtoon. [2, s. 20.]



Kuva 1. Erilaiset Bluetooth-tunnukset sekä yhteensopivuus eri tekniikoiden kanssa [4].

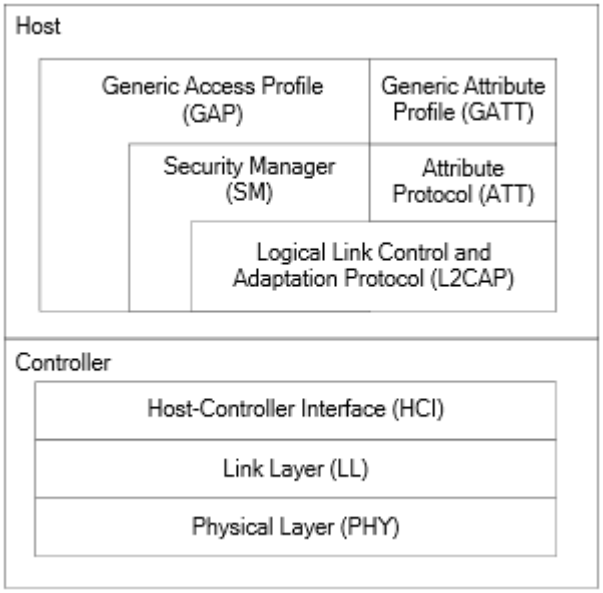
Bluetooth low energy kanssa yhteensopivat laitteet tunnistaa kuvassa 1 näkyvistä Bluetooth Smart- sekä Bluetooth Smart Ready -tunnuksista. Bluetooth Smart Ready tarkoittaa niin sanottua dual mode -laitetta, joka on yhteensopiva sekä klassisten Bluetooth-laitteiden, että Bluetooth low energy -laitteiden kanssa. Bluetooth Smart sen sijaan tarkoittaa niin sanottua single mode -laitetta, joka on yhteensopiva vain Bluetooth low energya tukevien laitteiden kanssa. Bluetooth Smart Ready -laitteita voivat olla esimer-

kiksi tietokoneet sekä matkapuhelimet ja Bluetooth Smart -laitteita syke- ja lämpömittari sekä kaukosäädin. [4.]

Bluetooth low energyn kehitys alkoi jo vuonna 2001, kun Nokia Research Centerin tutkijat huomasivat, että olemassa olevat langattomat tekniikat eivät soveltuneet kaikkiin käyttötarkoituksiin. Ratkaistakseen ongelman Nokian tutkijat aloittivat uuden entistä vähävirtaisemman ja halvemman langattoman tekniikan kehittämisen Bluetooth-standardin pohjalta.

Kehitystyön ensimmäiset tulokset julkaistiin vuonna 2004 nimellä Bluetooth low end extension. Kehitystyötä jatkettiin ja jatkokehityksen seurauksena tekniikka julkaistiin lokakuussa 2006 nimellä Wibree. Kesäkuussa 2007 Bluetoothia hallinnoivan Bluetooth SIGin (special interest group) kanssa käytyjen neuvotteluiden pohjalta sovittiin, että Wibree sisällytetään tulevaan Bluetooth-standardiin. [5, s. 6.] Yhtenäistämisprosessin jälkeen SIG julkaisi kesäkuussa 2010 Bluetooth 4.0 standardin, josta Wibree löytyy nimellä Bluetooth low energy [1; 6].

Bluetooth low energylla ja klassisella Bluetoothilla on monia yhtäläisyyksiä. Ne molemmat toimivat avoimella 2,4 GHz:n ISM (Industrial, scientific and medical) -taajuusalueella ja käyttävät tiedonsiirrossa GFSK-modulaatiota (Gaussian Frequency Shift Keying) sekä FHSS (Frequency-hopping spread spectrum) eli taajuushyppelyhajaspektritekniikkaa [7, s. 2; 8, s. 6]. Näiden yhtäläisyyksien ansiosta Bluetooth low energy voi hyödyntää dual mode -toteutuksissa osittain Bluetooth-radiota sekä protokollapinoa [2, s. 15, 21].



Kuva 2. Bluetooth low energy -protokollapino [9, s. 5].

Bluetooth low energy -protokollapino, joka on nähtävissä kuvassa 2, koostuu isäntä- ja ohjainlohkoista. Ohjainlohko pitää sisällään kolme tasoa, jotka ovat alhaalta ylöspäin lueteltuna fyysinen kerros, linkkikerros sekä HCI-rajapinta (Host-Controller interface). [9 s. 5–6.]

Fyysinen kerros

Kuten jo luvun alussa mainittiin, Bluetooth Low energy toimii fyysisellä kerroksella 2,4 GHz:n ISM-taajuusalueella. Bluetooth low energy -tekniikassa taajuusalue on jaettu neljäänkymmeneen 2 MHz:n levyiseen kanavaan. Kanavien keskitaajuudet voidaan laskea seuraavalla kaavalla

> `2402 + k x 2 MHz, jossa k on kanavan numero väliltä 0–39. [10, s. 2180.]`

Neljästäkymmenestä kanavasta kolme on niin sanottuja advertising-kanavia, joita laitteet käyttävät

- itsensä mainostamiseen muille laitteille
- muiden laitteiden etsimiseen
- yhteyden muodostamiseen laitteiden välille.

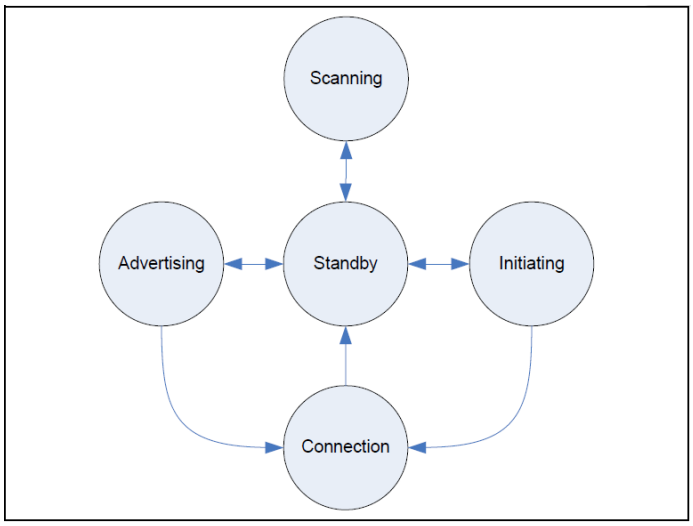
Advertising-kanavat on sijoitettu taajuuksille 2402, 2426 ja 2480 MHz, jolloin ne sijoittuvat IEEE 802.11 -standardin langattomien verkkojen kanavien 1, 6 ja 11 väliin minimoiden niiden mahdollisesti aiheuttamat häiriöt. [2, s. 16; 11, s. 3–4.]

Loppuja 37 kanavaa käytetään laitteiden väliseen tiedonsiirtoon. Muiden laitteiden taajuusalueelle aiheuttamien häiriöiden vaikutukset on pyritty minimoimaan FHSS-tekniikalla, jossa tiedonsiirtoon käytettyjen kanavien välillä hypitään ennalta sovitun mallin mukaisesti. [11, s. 3, 16.] Virrankulutuksen minimoimiseksi Bluetooth low energy -tekniikan ennalta sovitussa mallissa yhdellä kanavalla viivytään klassista Bluetooth-tekniikkaa kauemmin, jolloin kanavien vaihtojen aiheuttamat viiveet tiedonsiirrossa vähenevät [12].

Bluetooth low energy käyttää kanavien kanssa GFSK-modulaatiota, joka eroaa klassisen Bluetoothin käyttämästä GFSK-modulaatiosta modulaatioindeksin osalta [2, s. 19]. Klassisessa Bluetoothissa modulaatioindeksi on 0,28–0,35 ja Bluetooth low energyssä 0,45–0,55 [13 s. 12; 10 s. 2121]. Suuremman modulaatioindeksin etuina ovat pienempi virrankulutus, kohonnut tiedonsiirtonopeus sekä advertising-kanavien tarpeen väheneminen [2, s. 21].

Linkkikerros

Linkkikerroksen tehtävänä on pitää huolta radioyhteyden tilasta. Yhteyden tila voi vaihdella viiden tilan välillä, jotka ovat standby, advertising, scanning, initiating sekä connected. Kuvassa 3 nähtävä linkkikerroksen tilakoneen tilakaavio selventää eri tilojen välisiä suhteita. [10 s. 2194.]



Kuva 3. Linkkikerroksen tilakoneen tilakaavio [10, s. 2195].

Advertising-tila tarkoittaa tilaa, jossa laite mainostaa itseään advertising-kanavilla muille laitteille, jotka ovat scanning-tilassa eli etsivät muita laitteita. Jos joku scanning-tilassa olleista laitteista haluaa muodostaa yhteyden mainostettuun laitteeseen, se siirtyy initiating-tilaan. Initiating-tilassa se lähettää yhteyspyynnön asetuksineen laitteelle, joka mainosti itseään, ja jos tämä laite hyväksyy pyynnön, molemmat laitteet siirtyvät connected-tilaan. Conneted-tilassa yhteyttä pyytänyt laite saa muodostetun yhteyden isännän (master) roolin ja yhteyden hyväksynyt orjan (slave) roolin. Isännän roolin saanut laite voi olla yhteydessä useampaan orjaan, kun taas orja voi olla yhteydessä vain isäntäänsä. Standby-tilassa radioyhteys ei ole käytössä. [10, s. 2194–2196.]

HCI-rajapinta

HCI-rajapinta tarjoaa isännän ja kontrollerin välille standardoidun rajapinnan, joka mahdollistaa lohkojen välisen kommunikaation. Rajapinta voi olla toteutettu esimerkiksi APIn (application programming interface) eli ohjelmointirajapinnan tai SPIn (serial peripheral inteface) avulla. [9, s. 6.]

Isäntälohko

Protokollapinon isäntälohko koostuu useista eri protokollista, joilla on omat tehtävänsä. Näitä protokollia ovat

- L2CAP (Logical link control and adaption protocol)

- ATT (Attribute protocol)
- SM (Security Manager)
- GATT (Generic attribute profile)
- GAP (Generic access profile).

L2CAP-protokollan tehtävänä on kapseloida protokollapinon ylemmiltä tasoilta tulevat tieto, jotta se voidaan lähettää HCI-rajapinnan kautta ohjainlohkolle. Security managerin tehtävänä on huolehtia yhteyden turvallisesta muodostamisesta. GAPin tehtävänä on toimia rajapintana protokollapinon yläpuolella sijaitseville ohjelmaprofiileille sekä käsitellä linkkikerrokselta tulevien uusien laitteiden löydöt sekä niihin yhdistäminen.

ATT-protokollan avulla laite voi paljastaa itsestään attribuutteja toiselle laitteelle, näitä attribuutteja voi lukea tai niihin voi kirjoittaa. Tällaisia attribuutteja voivat olla esimerkiksi lämpötila-anturin lämpötila sekä mahdolliset lämpötilarajat, joiden ylittyessä tai alittuessa tulisi tehdä hälytys. GATT-taso mahdollistaa ATT-protokollan käytön määrittelemällä sille palvelukehyksen (service framework), jota protokollapinon yläpuolellä olevat ohjelmaprofiilit voivat käyttää. [9, s. 6.]

3 Elektrokardiografia

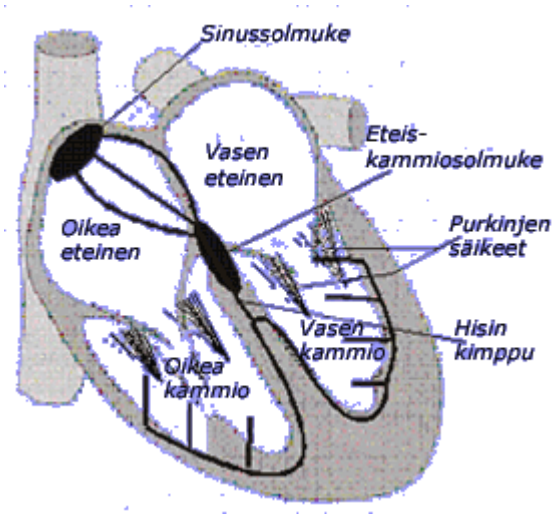
Elektrokardiografia eli EKG on yksi yleisimmistä fysiologisista tutkimuksista [14, s.6] ja samalla tärkein sydämen koneellinen tutkimus [15, s. 5]. Sillä tutkitaan sydämen sähköistä toimintaa ihon pinnalla tapahtuvien sydämen aiheuttamien sähköisten muutosten perusteella [16, s. 8].

Sydämen sähköisen toiminnan havaitsi ensimmäisen kerran englantilainen Augustus Desiré Waller 1800-luvun lopulla. Hän raportoi ensimmäisistä sydämen pinnalta mitatuista sähköpotentiaaleista vuonna 1887. Hänen työtään jatkoi alankomaalainen Willem Einthoven, joka vuonna 1902 kehitti ensimmäisen käyttökelpoisen EKG-laitteen, jolla pystyttiin mittaamaan sydämen aiheuttamia sähköisiä muutoksia kehon ulkopuolelta. Einthoven sai tekemästään työstä Lääketieteen Nobel-palkinnon vuonna 1924. Nykyisin käytössä olevan 12-kytkentäisen EKG-menetelmän kehitti Frank N. Wilson vuonna 1933. [14, s. 8.]

Elektrokardiografiassa käytetty EKG-laite, joka suodattaa ja vahvistaa signaalin, tuottaa sähköisten muutosten perusteella elektrokardiogrammin eli sydänsähkökäyrän, joka kuvaa sydämessä tapahtuvia jännitteen muutoksia ajan funktiona [17, s. 5–10]. Sydänsähkökäyrästä voidaan tulkita muun muassa sydämen lyöntitiheys, rytmin säännöllisyys, hapen- ja ravinnonsaanti sekä tunnusomaisten piirteiden perusteella mahdolliset sydänsairaudet [15, s. 5].

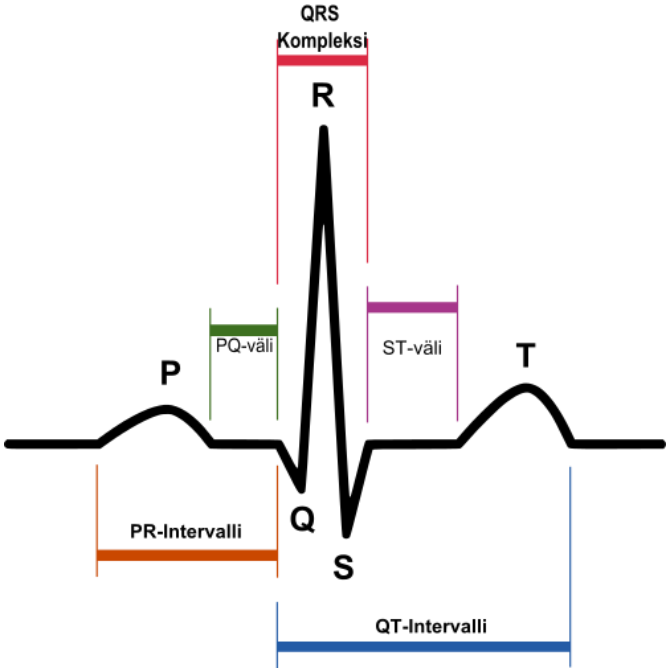
Sairaaloissa käytössä oleva 12-kytkentäinen EKG-menetelmä koostuu EKG-laitteesta ja siihen liitetyistä kahdestatoista elektrodista, jotka on kytketty potilaan iholle. Elektrodeista kuusi on kytketty rintakehälle ja loput kuusi raajoihin, jolloin niistä muodostuu kuusi paria, joiden välisiä potentiaalieroja mitataan. Tällöin tulokseksi saatu sydänsähkökäyrä koostuu kuudesta eri kuvaajasta. Rintakehän elektrodeista saatavat kolme kuvaajaa kuvaavat sydämen sähköistä toimintaa sydämen eri puolilta horisontaalitasossa ja raajoihin kytketyistä elektrodeista saatavat kolme kuvaajaa frontaalitasossa. [15 s. 7–10.]

Luotettavien ja vertailukelpoisten mittaustulosten takaamiseksi mittausprosessi sekä elektrodien paikat on standardoitu tarkasti. Ennen elektrodien kytkemistä potilaan iho tulee käsitellä resistanssin minimoimiseksi siten, että ensimmäiseksi elektrodien sijoituspaikoista tulee poistaa ihokarvat, tämän jälkeen iho tulee käsitellä alkoholilla ihoa suojaavan rasvakerroksen ja mahdollisen lian poistamiseksi. Viimeiseksi iholta tulee poistaa kuollut ihosolukko hankaamalla ihoa potilaskäyttöön kehitetyllä karhennusteipillä. [17, s. 14–15.]



Kuva 4. Sydämen johtoratajärjestelmä [18].

Sydänlihaksen ollessa lepotilassa sen sydänlihassolujen ulko- ja sisäpuolen välillä vallitsee potentiaaliero. Sydänsähkökäyrässä näkyvät niin sanotut heilahdukset ovat sydänlihassolujen de- ja repolarisaatioita eli sydänlihassolujen ulko- ja sisäpuolen potentiaalierojen häviämisiä ja palautumisia. Depolarisaatiot aiheuttavat sydänlihassolujen supistumisen, ja ne saavat alkunsa sinussolmukkeen aktivaatiosta, joka aiheuttaa sydämen eri osiin kulkeutuvan sähköisen impulssin. Impulssi etenee sydänlihaksen eri osiin kuvassa 4 näkyvää johtoratajärjestelmää pitkin, joka koostuu sinussolmukkeesta, eteiskammiosolmukkeesta, Hisin kimpusta, johtoradoista sekä Purkinjen säikeistä.[14 s. 9; 15 s. 6.]



Kuva 5. Sydänsähkökäyrä [19].

Sinussolmukkeen aktivaation aiheuttama sähköinen impulssi kulkeutuu ensin eteisiin aiheuttaen eteisten depolarisaation, joka näkyy sydänsähkökäyrällä (kuvassa 5) P-aaltona. Tämän jälkeen impulssi siirtyy johtoratajärjestelmää pitkin kammioille aiheuttaen kammioiden depolarisaation, joka näkyy sydänsähkökäyrällä (kuvassa 5) kolmena heilahduksena, jotka tunnetaan QRS-kompleksina. Kompleksin heilahduksista Q- ja S-aalto ovat negatiivisia ja R-aalto positiivinen. Kammioiden lihasmassan ollessa paljon suurempi kuin eteisten lihasmassan, on myös QRS-kompleksin amplitudi merkittävästi P-aaltoa suurempi. Normaalisti QRS-kompleksin jälkeen sydänsähkökäyrässä (kuvassa 5) näkyy enää vain yksi heilahdus, jota kutsutaan T-aalloksi. T-aalto johtuu kammi-

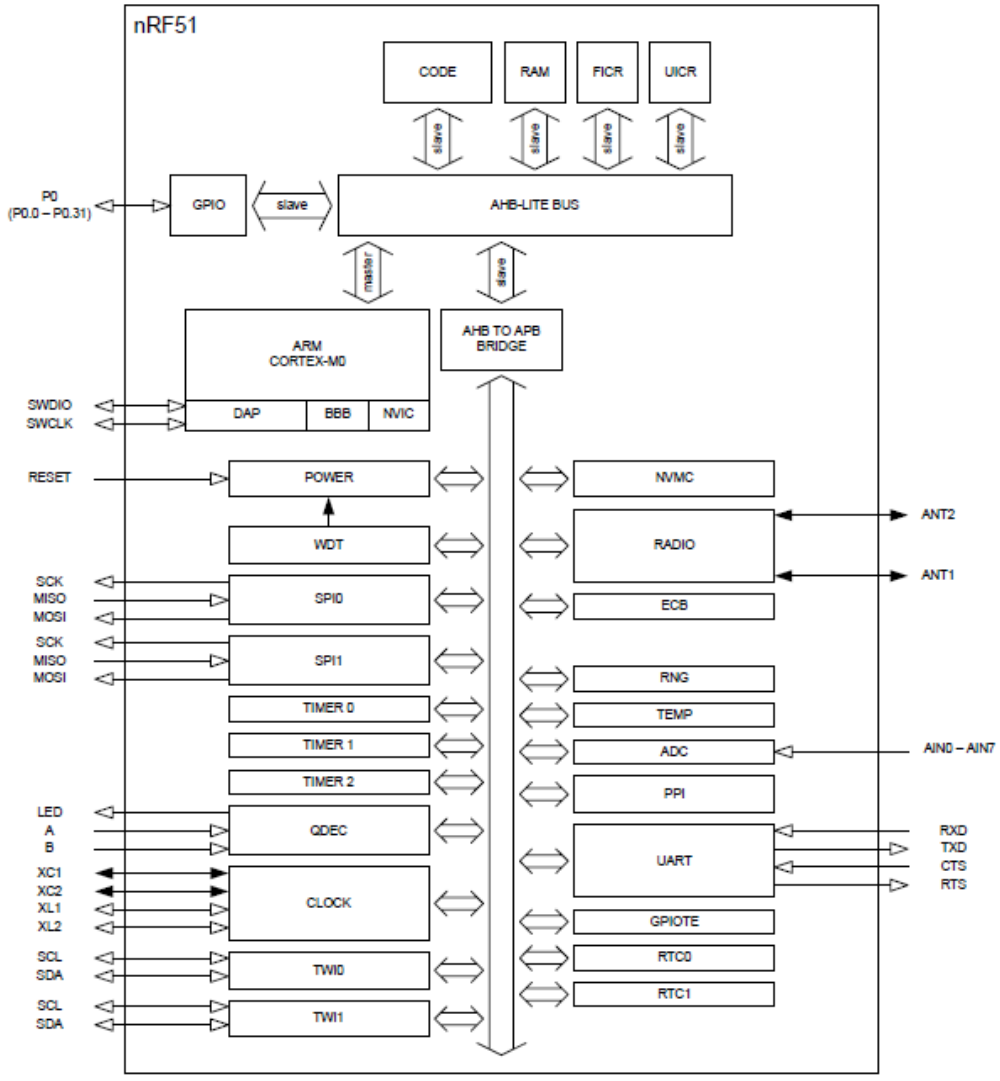
oiden repolarisaatiosta. Eteisten repolarisaatio ei sydänsähkökäyrässä normaalisti näy, koska kammioiden depolarisaatio peittää sen alleen. [16 s. 8–9.]

4 Järjestelmän komponentit

Kuten luvussa 1 mainittiin, insinöörityön tavoitteena oli kehittää järjestelmä, joka selvittää EKG-anturista luetusta signaalista sydämen lyöntitiheyden ja välittää sen Bluetooth low energyn avulla iPhonelle, jossa se oli tarkoitus esittää. Järjestelmä koostuu nRF51822-piirin sisältävästä kehitysalustasta sekä iPhone 5-älypuhelimesta, jotka esitellään seuraavaksi.

4.1 nRF51822-piiri

nRF51822 on Nordic Semiconductorin syksyllä 2012 julkaisema erittäin vähävirtainen SoC (system on chip) -piiri, joka toimii 1,8–3,6 V:n käyttöjännitteellä. Valmistaja lupaa sen kuluttavan toimettomana ollessaan 3 V käyttöjännitteellä vain 2,5 µA virtaa. Piiri on saatavissa 6 x 6 mm 48-nastaisessa QFN (Quad-flat no-leads) -kotelossa. [20.]



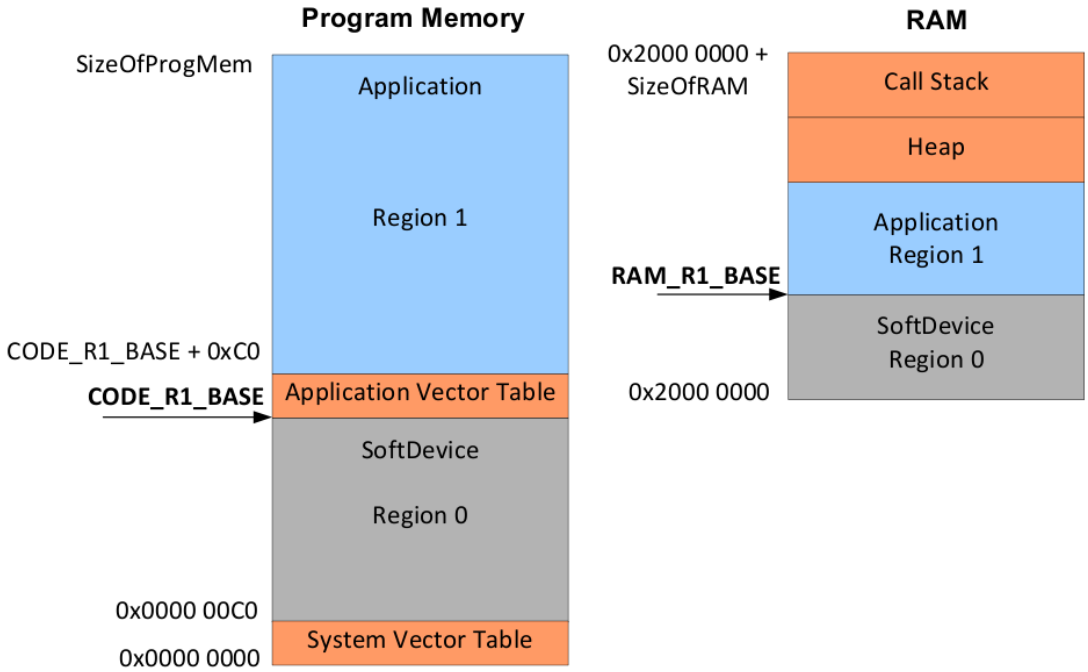
Kuva 6. nRF51822-piirin lohkokaavio [21, s. 7].

nRF51822 koostuu muun muassa 16 MHz:n kellotaajuudella toimivasta 32-bittisestä Cortex-M0-mikrokontrollerista, 256 kt:n flash-ohjelmamuistista, 16 kt:n RAM-muistista sekä 2,4 GHz:n lähetinvastaanottimesta eli radiosta [17]. Lohkokaaviosta, joka on esitetty kuvassa 6, selviää, että piiristä löytyy myös 32 GPIO (general purpose input/output) -nastaa sekä esimerkiksi I²C-, SPI-, UART- ja A/D-muunninlohkot, jotka mahdollistavat useiden lisälaitteiden liittämisen piiriin. [21, s. 7.]

Monia piirin ominaisuuksia on mahdollisuus muuttaa ohjelmallisesti. Esimerkiksi A/D-muuntimen asetuksista muokattavissa on muun muassa A/D-muuntimen resoluutio, sisääntulonasta sekä sisääntulon- ja referenssijännitteen vaimennuskertoimet [21, s.161–165].

Piirin sisältämä DAP-lohko (Debug Access Port) mahdollistaa JTAG/SWD-emulaattorin liittämisen piiriin SWD-liitännän välityksellä, joka on 2-nastainen vaihtoehtoinen liitäntä perinteiselle 5-nastaiselle JTAG-liittännälle. JTAG/SWD-emulaattorin avulla SWD-liitännän kautta on mahdollista esimerkiksi ohjelmoida piiri sekä etsiä virheitä mikrokontrollerin ohjelmakoodista sen suorittamisen aikana. [22.]

nRF51822-piirin sisältämä radio toimii 2,4GHz:n taajuudella, jolla se tukee Nordic Semiconductorin omia Gazell- sekä Enhanced Shockburst-radioprotokollia. Näiden lisäksi se tukee Bluetooth low energyä erillisen S110 SoftDevice -nimisen protokollapinon avulla, joka on ladattavissa valmiiksi käännettynä ja linkitettynä Nordic Semiconductorin kotisivulta. [20.]



Kuva 7. S110 SoftDevice-protokollapinon sijainti muistissa [23, s. 14].

S110 SoftDevice-protokollapino ohjelmoidaan 256 kt:n ohjelmamuistin alkuun, josta se kuluttaa 80 kt muistia, jolloin itse sovellukselle jää ohjelmamuistiin tilaa 176 kt. Ohjelmamuistin lisäksi protokollapino kuluttaa myös RAM-muistia, jossa myös käskypino sekä keko (heap) sijaitsevat (kuva 7). Kulutetun RAM-muistin määrä riippuu siitä, onko protokollapino otettu käyttöön ohjelmakoodissa vai ei. Kulutetun muistin määrä eri tilanteissa selviää taulukosta 1.

Taulukko 1. S110 SoftDevice-protokollapinon kuluttamat muistiresurssit.

	S110 käytössä	S110 ei käytössä
Ohjelmamuisti	80 kt	80 kt
RAM	8 kt	4 kt
Kutsupino	1,5 kt	0 kt
Keko (heap)	0 kt	0 kt

Protokollapinon ollessa käytössä ohjelmakoodista ei voida viitata osaan keskeytysvektoreista eikä järjestelmän lohkoista, koska protokollapino tarvitsee niitä. Tällaisia lohkoja ovat esimerkiksi radio ja yksi ajastimista. Osaa lohkoista taas voidaan käyttää normaalisti tai rajoitetusti protokollapinon ohjelmointirajapinnan välityksellä. [23, s. 15–16.]

S110 SoftDevice-protokollapinoa käytetään Nordic Semiconductorin SDKn sisältämän ohjelmointirajapinnan välityksellä. Itse ohjelmointirajapinta kutsuu protokollapinoa ARM-mikrokontrollerien tukemilla SVCall (SuperVisor Call) -järjestelmäkutsuilla samaan tapaan, kuin jos järjestelmäkutsuilla kutsuttaisiin käyttöjärjestelmää. Käytetyn mallin etuna on se, että protokollapinoa ei tarvitse linkittää ohjelmakoodin kanssa. [23, s.1.]

4.2 Kehitysalusta

Mikropiirivalmistajat tarjoavat usein piiriensä ominaisuuksien vertailua ja testaamista varten erilaisia kehitysalustoja. Kehitysalustojen mukana toimitetaan yleensä myös niiden tarvitsemat kaapelit, kirjastot sekä esimerkkisovellusten lähdekoodit. Ominaisuuksien vertailun ja testaamisen lisäksi kehitysalustat mahdollistavat myös kehitystyön nopean aloittamisen ilman, että yrityksen tulisi heti luoda piirille omaa prototyyppiä.

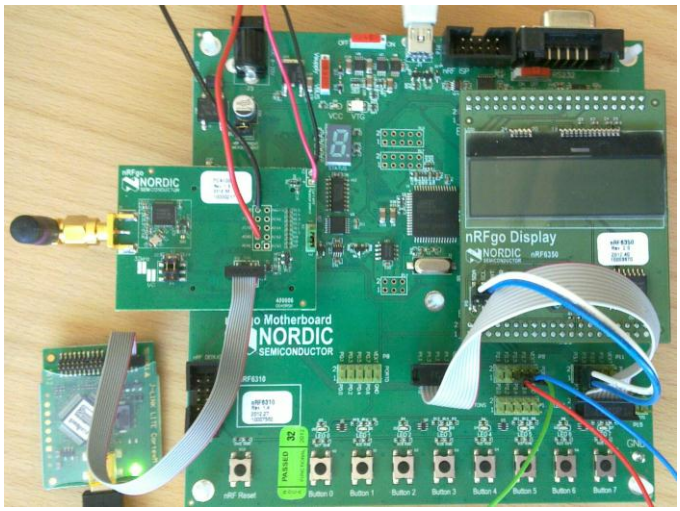
Nordic Semiconductorilta on saatavilla kaksi nRF51822-piirin sisältämää kehitysalustaa: nRF51822 Evaluation kit sekä nRF6310-kehitysalusta, johon on saatavilla nRF51822 development kit-moduuli. [20.]

Kuva 8. nRF51822 Evaluation kit.

nRF51822 Evaluation kit (kuvassa 8) sisältää yhdellä piirilevyllä kaiken tarvittavan nRF51822-piirin ominaisuuksien testaamiseen, arviointiin sekä pienimuotoiseen kehitystyöhön. Piirilevy sisältää

- pidikkeen nappiparistolle
- kaksi painokytkintä
- kaksi LEDiä
- Segger J-link OB JTAG/SWD-emulaattoripiirin
- piirilevyantennin.

Piirilevylle integroitu J-Link OB JTAG/SWD-emulaattoripiiri mahdollistaa nRF51822-piirin ohjelmoimisen sekä virheenetsinnän. [24.]



Kuva 9. nRF51822 Development kit-moduuli + nRF6310-kehitysalusta sekä J-Link Lite CortexM-9 JTAG/SWD-emulaattori.

nRF51822 Development kit (kuvassa 9) sisältää ainoastaan kaksi moduulia, jotka sisältävät nRF51822-piirin ja sen tarvitsemat komponentit. Moduulit ovat muuten toisiaan vastaavia, mutta toisessa niistä on piirilevyantenni ja toisessa SMA-liitin mukana toimitetun antennin liittämistä varten. Moduulit ovat tarkoitettu liitettäväksi nRF6310-kehitysalustaan (kuvassa 9), joka on Nordic Semiconductorin erittäin vähävirtaisten piirien kehitysalusta. nRF6310-alustaan liitettynä Development kit sisältää

- liitttimen AA-paristoille
- kahdeksan painokytkintä
- kahdeksan LEDiä
- 9-nastaisen sarjaportin
- LCD-näytön
- DC-virtaliittimen
- Segger J-Link Lite CortexM-9 JTAG/SWD-emulaattorin. [25.]

Molemmissa kehitysalustoissa on myös USB-liitin, DC/DC-muunnin, piikkirimaliittimet GPIO-nastoille sekä 32,768 kHz:n ja 16 MHz:n kiteet nRF51822-piirin sisäänrakennettujen RC-oskillaattorien lisäksi. Kehitysalustojen mukana tulee myös koodi, joka mahdollistaa piirin liittyvien dokumenttien ja software development kitin eli SDK:n lataamisen valmistajan kotisivuilta. SDK sisältää nRF51822-piirin käyttöä helpottavat kirjastot,

esimerkkiohjelmien lähdekoodeja sekä Keil MDK IDEä (integrated development enviroment) varten konfiguraatiotiedostot. [24; 25.]

Electrialle oli tilattu projektia varten molempia kehitysalustoja. Niiden kokeilun jälkeen insinöörityössä päädyttiin käyttämään nRF51822 Development kitiä yhdessä nRF6310-kehitysalustan kanssa, koska se tarjoaa laajemmat mahdollisuudet kehitystyölle. Tämän lisäksi SDK:n sisältämien esimerkkiohjelmien lähdekoodeista useammat on tarkoitettu Development kitille, joka myös helpotti kehitystyön nopeaa aloittamista.

4.3 iPhone 5

iPhone 5 on Applen syyskuussa 2012 julkaisema kosketusnäytöllinen älypuhelin. Puhelin sisältää Applen uuden A6 SoC-piirin, jonka Apple sanoo olevan kaksi kertaa edeltäjäänsä nopeampi. A6-piiri sisältää Applen suunnitteleman kaksiytimisen ARMv7-pohjaisen prosessorin, joka toimii 1,3 GHz:n kellotaajuudella. Prosessorin lisäksi A6-piiristä löytyy kolmeytiminen PowerVR SGX 543MP3 näytönohjain sekä yhden gigatavun verran 1066 Hz:n LPDDR2-keskusmuistia. [26; 27.]

Suomessa myytävä malli iPhone 5:stä tukee toisen sukupolven GSM-matkapuhelinverkkoja neljällä eri taajuudella, tuetut taajuudet ovat 850, 900, 1800 ja 1900 MHz. Kolmannen sukupolven UMTS-matkapuhelinverkkoja malli tukee niin ikään neljällä eri taajuudella, jotka ovat 850, 900, 1900, 2100 MHz. LTE-verkon taajuuksista tuettuja ovat 850, 1800 ja 2100 MHz.

Muista tiedonsiirtotekniikoista iPhone 5 tukee Bluetooth-standardin versiota 4.0 sekä langattoman lähiverkon IEEE 802.11 -standardeja a, b, g ja n. 802.11n-standardia se tukee taajuuksilla 2,4 ja 5 GHz. [27.]

Projektin kannalta tärkeintä oli tuki Bluetooth 4.0-standardille, koska kyseisessä standardissa esiteltiin tuki Bluetooth low energy -tekniikalle. Tekniikka toimii projektissa tärkeässä roolissa, koska sen avulla EKG-anturista luettu tieto siirretään mobiililaitteelle eli insinöörityön tapauksessa iPhone 5:lle. Muitakin edellä mainittuja tekniikoita tullaan projektin edetessä hyödyntämään, kun tietoa ryhdytään siirtämään pilvipalveluun, mutta projektin laajuuden vuoksi insinöörityössä ei tätä osuutta tarkemmin käsitellä.

iPhone 5 käyttää muiden iPhone-puhelinten tapaan Applen iOS-käyttöjärjestelmää. iPhonejen lisäksi iOSia käyttävät myös Applen iPod touch -soittimet, Apple TV-mediatoistin sekä iPad-taulutietokoneet. iPhone 5 käyttää käyttöjärjestelmän versiota 6, joka on kirjoitushetkellä myös uusin versio iOSista. iOSille on saatavissa Applen App storen kautta yli 800 000 erilaista sovellusta, joita on ladattu yhteensä yli 40 miljardia kertaa. [28; 29.]

iOS-sovellukset kirjoitetaan pääasiallisesti Objective-C-ohjelmointikielellä, joka on C-ohjelmointikielen oliolaajennus. Objective-C:n kehityksestä ovat vastanneet pääasiallisesti Brad Cox ja Tom Love, jotka kehittivät kielen 1980-luvun alussa. Nykyään kieltä käytetään lähinnä Applen Mac OS X- sekä iOS-käyttöjärjestelmissä. [30.]

iPhone 5:n hankintaan päädyttiin, koska projektin alkaessa se oli edeltäjänsä iPhone 4S:n kanssa ainoa älypuhelin, jossa oli yhtenäinen Bluetooth low energy tuki. Projektin alkaessa markkinoilta löytyi iPhonejen lisäksi muun muassa Samsungin Android-laitteita, joille Bluetooth-piirin valmistaja oli kehittänyt Bluetooth Low energy tuen. Näiden laitteiden ongelmaksi muodostui kuitenkin se, että Androidille ei vielä kirjoitushetkelläkään ole virallisesti Bluetooh low energy tukea.

Android-laitteen valinta olisi johtanut siihen, että kirjoitettava mobiilisovellus olisi toiminut Bluetooth low energyn osalta hyvässäkin tapauksessa vain saman valmistajan Bluetooth-piirin sisältävissä laitteissa. iPhonen valinnan myötä kirjoitettava mobiilisovellus tulee Bluetooth low energyn osalta toimimaan kaikissa iOSia käyttävissä laitteissa, joissa on tarvittava laitteisto. [31.]

5 Mikrokontrollerin kehitysympäristö

Kehitysympäristöllä tarkoitetaan ohjelmaa tai ohjelmia, joita ohjelmistokehittäjä käyttää kehittäessään ohjelmistoa. Yksinkertaisimmillaan tämä voi tarkoittaa tekstieditoria ja erillistä komentorivikääntäjää. Samaan pakettiin integroidusta ohjelmistoympäristöstä käytetään nimitystä IDE (Integrated development environment). [32.]

Nordic Semiconductor tukee nRF51822-piirin ohjelmistokehityksessä Keilin MDK-ARM IDEä tarjoamalla sille tarvittavat konfiguraatiotiedostot. MDK-ARM IDEstä on saatavissa Microsoftin Windows-käyttöjärjestelmälle useita versioita, joista yksi on ilmainen.

Ilmaisessa versiossa on kuitenkin useita rajoituksia, joista merkittävimpänä käännettävän ja linkitettävän koodin 32 kt:n kokorajoitus. [33.]

Projektissa päädyttiin piirivalmistajan suositusten sijaan käyttämään avoimen lähdekoodin työkaluja. Avoimen lähdekoodin työkaluihin päädyttiin, koska projektiin osallistujilla oli laajaa aikaisempaa kokemusta valituista työkaluista. Valittu kehitysympäristö koostui VIM-tekstieditorista, GNU make-työkalusta sekä Sourcery CodeBench Lite Edition-ristiinkääntötyökaluketjusta, joka sisälsi GNU-projektin käännöstyökalut ristiinkääntöön konfiguroituna. Ristiinkäännölla tarkoitetaan toiselta prosessoriarkkitehtuurilta toiselle tapahtuvaa käännösoperaatiota ja sitä tarvittiin, jotta lähdekoodi saatiin käännettyä työasemalla mikrokontrollerille sopivaan muotoon [34].

5.1 Vim-tekstieditori

Vim on usealle eri käyttöjärjestelmälle saatavissa oleva monipuolinen avoimen lähdekoodin komentoriviltä käytettävä tekstieditori, joka on suosittu erityisesti ohjelmoijien keskuudessa. Bram Moolenaar kirjoitti Vimin korvaamaan Vi-tekstieditorin, jota ei ollut saatavilla Amiga-tietokoneelle. Vim sisältää tätä nykyä myös paljon ominaisuuksia, joita Vi-tekstieditorissa ei ole. Tästä kertoo myös Vimin nimi, joka on lyhenne sanoista Vi improved. [35.]

Vimin suunnittelussa on pyritty siihen, että käyttäjän ei koskaan tarvitsisi irrottaa käsiään näppäimistöltä valitakseen esimerkiksi tekstiä. Tästä syystä Vim sisältää kolme eri tilaa: komentotilan, kirjoitustilan sekä visuaalisen tilan. Vimin käynnistyessä tilana on aina komentotila, jossa kirjoitettu teksti tulkitaan komennoksi. Kirjoitustilassa Vim toimii normaalin tekstieditorin tapaan kirjoittaen tekstiä muokattavaan tiedostoon. Visuaalisessa tilassa nuolinäppäimillä voidaan valita tekstiä ja muilla näppäimillä suorittaa komentoja. Komentotilasta pääsee kirjoitustilaan painamalla i-näppäintä ja visuaaliseen tilaan painamalla v-näppäintä. Komentotilaan pääsee aina painamalla esc-näppäintä.

Vimin ominaisuuksia ovat muun muassa

- tuki yli 200 ohjelmointikielen syntaksin korostukselle
- mahdollisuus avata useita tiedostoja jaetulle ruudulle

- kursorin alla olevan luvun kasvattaminen tai vähentäminen
- hakutoiminto, jolla voidaan etsiä tai korvata merkkijonoja
- historia edellisille komennoille ja hauille
- istunnon palauttaminen
- mahdollisuus suorittaa sovelluksia poistumatta Vimistä.

Vimin käyttö yhdessä muiden projektissa käytettyjen työkalujen mahdollisti jopa nRF51822-piirin ohjelmoinnin sekä virheenetsinnän suoraan Vimistä käsin, jolloin Vim toimi integroidun kehitysympäristön kaltaisesti. [36.]

5.2 Make-työkalu

Make on tiedostojen luomisen hallintaan ja automatisointiin tarkoitettu työkalu. Sitä käytetään usein ohjelmistokehityksessä, kun halutaan kääntää ja linkittää useasta lähdekooditiedostosta koostuva sovellus.

Make tarvitsee sovelluksen kääntämistä varten niin sanotun makefile-tiedoston, jossa on kerrottu miten sovellus käännetään ja mistä tiedostoista se on riippuvainen. Riippuvuudella tarkoitetaan niitä tiedostoja, joita muutettaessa sovellus tulee kääntää uudestaan. Niiden perusteella Make päättelee jokaisen käännöksen tarpeellisuuden, jos lähdekooditiedostoja ei ole muokattu viime käännöksen jälkeen on käännös turha, koska sovelluksesta tulee täysin edellistä käännöstä vastaava. [37.]

5.3 GCC-kääntäjä

GCC eli GNU Compiler Collection on GNU-projektin luoma kääntäjäkokoelma, joka mahdollistaa seuraavilla ohjelmointikielillä kirjoitettujen ohjelmien kääntämisen objektitiedostoiksi:

- C
- C++
- Objective-C

- Fortran
- Java
- Ada
- Go.

Se sisältää myös joitakin kirjastoja edellä mainituille ohjelmointikielille, näistä esimerkkinä C++-kielen libstdc++-kirjasto. [38 s. 3, 675.]

GCC kirjoitettiin alun perin GNU-käyttöjärjestelmän kääntäjäksi, mutta nykyään se on saatavilla myös monelle muulle käyttöjärjestelmälle, kuten esimerkiksi Linuxille, Windowsille sekä BSD-käyttöjärjestelmille. GCC on saatavilla myös useille eri prosessoriarkkitehtuureille. [39.]

Projektin kannalta olennaista oli se, että GCC tuki ristiinkääntämistä Intelin x86-arkkitehtuurilta ARM-arkkitehtuurille. Tämä mahdollisti ohjelmien kehityksen sekä kääntämisen mikrokontrollerille normaalilla x86-prosessorin sisältävällä työasemalla.

Kirjoitushetkellä GCCn uusin versio on 4.8, joka on julkaistu maaliskuussa 2013. Projektissa käytetty Sourcery CodeBench Lite Editionin sisältää GCC version 4.5.2. [40.]

5.4 GNU linker -linkitin

GNU linker on GNU-projektin luoma linkitin, joka tunnetaan myös nimellä GNU ld. Sen tarkoitus on luoda ajettava tiedosto tai vaihtoehtoisesti kirjasto kääntäjän lähdekoodista kääntämille objektitiedostoille. Linkitin käyttää linkitystä tehdessään linkitinskriptiä, jonka tarkoitus on kertoa linkittimelle, mihin kohtaan muistiavaruutta minkäkin objektitiedostoissa määritellyn muistialueen tulee sijoittua ajettavassa tiedostossa. Tämän lisäksi linkitinskriptissä voidaan myös määritellä muuttujien kaltaisia symboleita, joihin voidaan viitata ohjelmakoodissa. [41; 42.]

5.5 GDB-virheidenetsintätyökalu

GDB on GNU-projektin virheidenetsintätyökalu, joka on saatavilla muun muassa Linuxille sekä Windowsille. GDB:llä on mahdollista etsiä ohjelmointivirheitä muun muassa

C, C++ sekä Objective-C-kielellä kirjoitetuista ohjelmista, joita suoritetaan joko samassa laitteessa kuin GDB:tä tai etänä toisessa laitteessa. Toisessa laitteessa suoritettavan ohjelman virheidenetsintää varten tarvitaan GDB:n lisäksi GDB server, joka mahdollistaa yhteyden muodostuksen laitteiden välille. [43; 44.]

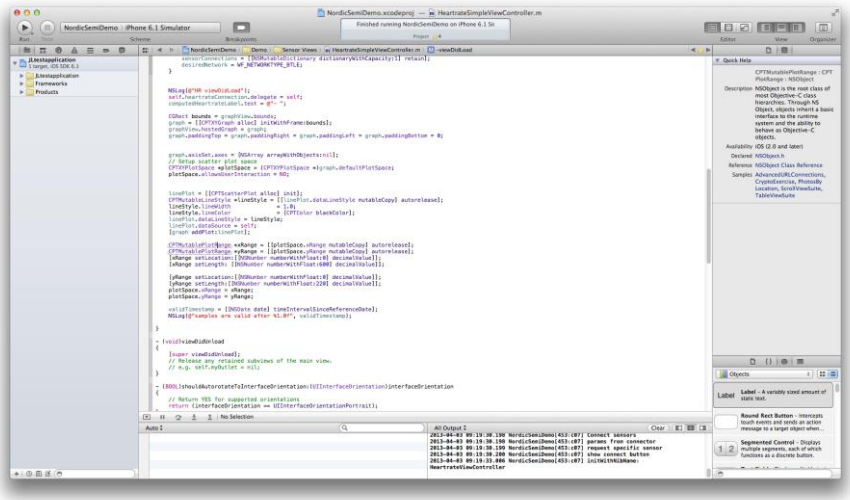
GDB-virheidenetsintätyökalu mahdollistaa esimerkiksi muuttujien arvojen seuraamisen sekä muuttamisen ajon aikana, ohjelman etenemisen seuraamisen rivi riviltä, ohjelman pysäyttämisen halutussa paikassa (breakpoint) tai ennalta määrätyn asian toteutuessa (watchpoint). GDB sisältää myös sisäänrakennetun purkajan (disassembler), jolla konekielinen koodi voidaan muuntaa assembly-kielelle, ja täten löytää esimerkiksi mahdolliset kääntäjävirheet. Virheiden etsinnän lisäksi GDB:n avulla on myös mahdollista ohjelmoida sulautettuja laitteita. [45.]

Projektissa GDB:tä käytettiin pääasiassa nRF51822-piirin ohjelmointiin. Sitä varten tuli ladata Seggerin sivuilta J-Link GDB server. J-Link GDB server mahdollistaa nRF51822 Development Kitin mukana tulleen Segger J-Link Lite CortexM-9 JTAG/SWD-emulaattorin käytön GDB:n välityksellä.

6 iOS-kehitysympäristö

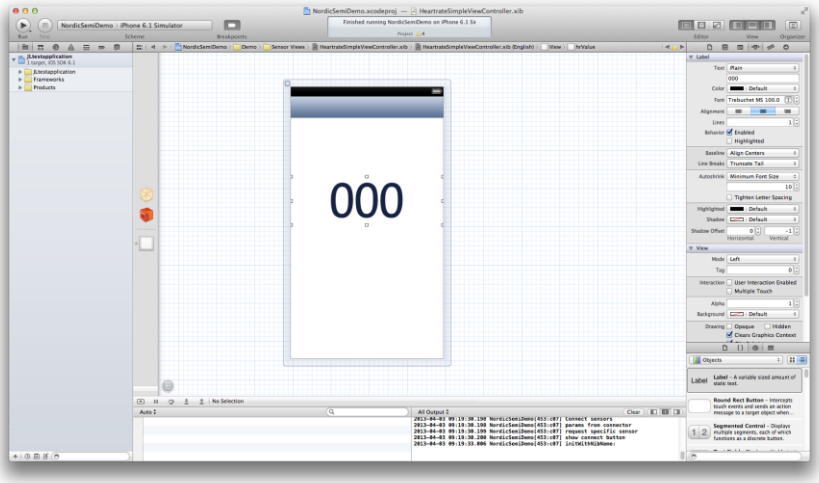
Apple on tehnyt hyvin vaikeaksi avointen työkalujen käytön iOS-sovelluskehityksessä; avoimilla työkaluilla kehitettyjä sovelluksia ei ole esimerkiksi mahdollista testata iPhonessa ilman niin sanottua jailbreakia, joka mahdollistaa kolmannen osapuolen sovellusta suorittamisen iOS-käyttöjärjestelmää hyödyntävissä laitteissa. Jailbreakin käyttö aiheuttaa laitteen takuun raukeamisen, joka ei ole toivottavaa. Tästä syystä projektissa päädyttiin käyttämään Applen Xcode-kehitysympäristöä.

Xcode on Applen integroitu kehitysympäristö (IDE), jonka ensimmäinen versio julkaistiin vuonna 2003. Se mahdollistaa sovelluskehityksen Applen OS X- ja iOS-käyttöjärjestelmille. Xcode on alusta asti ollut saatavilla vain Applen OS X -käyttöjärjestelmille ja sen kirjoitushetkellä uusin versio (4.6) tukee vain kahta viimeisintä OS X -käyttöjärjestelmää, eli Mac OS X Lionia ja OS X Mountain lionia. Kehittäjät voivat ladata Xcoden Applen sovelluskauppa Mac app storesta. [46; 47.]



Kuva 10. Xcoden käyttöliittymä.

Xcoden käyttöliittymä, joka on nähtävissä kuvassa 10, on laajasti muokattavissa. Vasemmalla olevan navigointipalkin eri välilehdistä on mahdollista muun muassa selata avatun projektin tiedostoja, etsiä tietoa avoinna olevasta projektista sekä seurata korjaamattomia ongelmia. Keskeltä löytyvän päänäkymän sisältö riippuu navigointipalkista valitusta tietueesta, jos valittuna on lähdekooditiedosto, päänäkymässä näkyy lähdekoodieditori, ja, jos taas valittuna on esimerkiksi käyttöliittymätiedosto, niin näkyvissä on interface builder, jota käytetään graafisen käyttöliittymän suunnitteluun. Päänäkymän alapuolelta löytyy lokipalkki, josta voi seurata tapahtumahistoriaa. Oikealta löytyy objektipalkki, josta voi lisätä käyttöliittymäelementtejä interface builderiin sekä tarkastella ja muokata valitun käyttöliittymäelementin ominaisuuksia.



Kuva 11. Xcoden interface builder-työkalu.

Graafisen käyttöliittymän suunnitteluun tarkoitettu Interface builder, joka on nähtävissä kuvassa 11, mahdollistaa sovelluksen graafisen käyttöliittymän suunnittelun. Suunnittelu tapahtuu raahaamalla käyttöliittymäelementtejä objektipalkin oikeasta alareunasta löytyvästä valikosta päänäkymässä näkyvään käyttöliittymään (view controller). Raahattavat käyttöliittymäelementit voivat olla vaikkapa tekstikenttiä, painonappeja tai taulukoita. Käyttöliittymään lisättyjen käyttöliittymäelementtien ominaisuuksia voi muuttaa objektipalkin yläosasta löytyvästä osiosta.

Xcode mahdollistaa iOS-sovellusten testaamisen joko Applen kehittämässä iPhone/iPad-simulaattorissa tai oikeassa laitteessa. Sovelluksen testaaminen oikeassa laitteessa edellyttää kuitenkin maksullisen Apple Developer -lisenssin hankkimista. Simulaattori mahdollistaa ohjelman suorittamisen ja virheidenetsinnän tietokoneella, mutta sillä ei ole mahdollista simuloida esimerkiksi Bluetooth-piirin toimintaa. Tästä syystä Bluetooth low energy ominaisuuksien testaamiseen jouduttiin hankkimaan Apple Developer-jäsenyys, joka mahdollisti sovelluksen suorittamisen iPhonessa. [47.]

Xcode käyttää ohjelmakoodin kääntämiseen LLVM-kääntäjää, joka Applen mukaan toimii kaksi kertaa nopeammin kuin ennen Xcoden versiota 4.2 käytössä ollut GCC-kääntäjä. Sen tuottama konekielinen koodi on myös nopeampaa GCC:n tuottamaan koodiin verrattuna. Ohjelmakoodin virheenetsintään Xcode käyttää LLDB-virheidenetsintätyökalua, joka mahdollistaa virheiden etsinnän sekä iPhone/iPad-simulaattorissa että oikeassa iOSia hyödyntävässä laitteessa.

Xcoden Instruments -työkalua on mahdollista hyödyntää sovelluksen pullonkaulojen selvittämiseen. Se mahdollistaa sovelluksen toiminnan monitoroinnin suorituksen aikana. Monitoroinnin avulla on mahdollista arvioida esimerkiksi sovelluksen virrankulutusta, verkon käyttöä sekä suorituskykyä. Instrumentsilla on myös mahdollista löytää muistivuotoja sekä muita suorituskykyyn vaikuttavia virheitä. [48; 49.]

7 Työn valmistelu

7.1 Laitteiden ja lisenssien hankinta

Ennen kuin insinöörityön tekeminen oli mahdollista aloittaa, projektia varten täytyi suorittaa laitehankintoja. Näitä laitehankintoja olivat iPhone 5 -älypuhelin sekä iOS-

sovelluskehitykseen käytetty Mac Mini -tietokone. Laitehankintojen lisäksi iOS-sovelluskehitystä varten täytyi hankkia, luvussa 6 mainittu, Apple Developer -lisenssi.

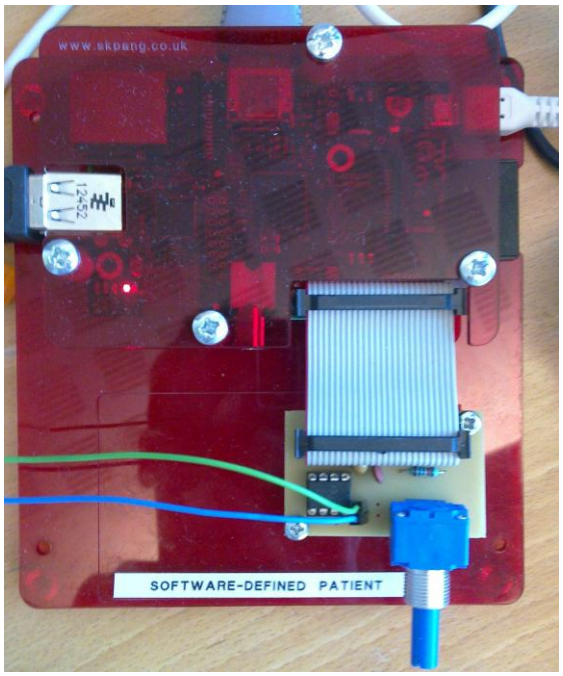
Metropolia Ammattikorkeakoulu on hankintalain mukaisesti kilpailuttanut tuotteiden, laitteiden, palveluiden sekä urakoiden hankinnan. Kilpailuttamisella on pyritty julkisten varojen mahdollisimman tehokkaaseen hyödyntämiseen eli siihen, että hankittavien tuotteiden, laitteiden palveluiden sekä urakoiden hinta-laatusuhde on mahdollisimman hyvä. [50, 51.]

Projektin kannalta kilpailuttaminen hankaloitti laitteiden hankintaa, koska kilpailuttamisen seurauksena projektia varten ei voitu hankkia laitteita suoraan esimerkiksi verkkokaupoista, vaan sen sijaan laitteet tuli hankkia keskitetysti Metropolian tietohallinnon välityksellä. Tietohallinnon kautta tilattu Iphone 5 saapui noin kuukauden sisällä tilaushetkestä, mutta erinäisistä syistä johtuen Mac Minin toimitusprosessi kesti lähes kolme kuukautta, mikä hidasti huomattavasti insinöörityön etenemistä.

Apple Developer -lisenssin hankinta onnistui sekin tietohallinnon välityksellä. Hankintaa varten tehtiin tukipyyntö helpdeskille, joka konsultoi iOS-sovelluskehityksestä vastaavaa opettajaa. Opettajan välityksellä käyttöön saatiin Apple Developer -lisenssi, joka mahdollisti sovelluksen testaamisen laitteessa, mutta ei sovelluksen lisäämistä App storeen.

7.2 Käytössä oleva laitteisto

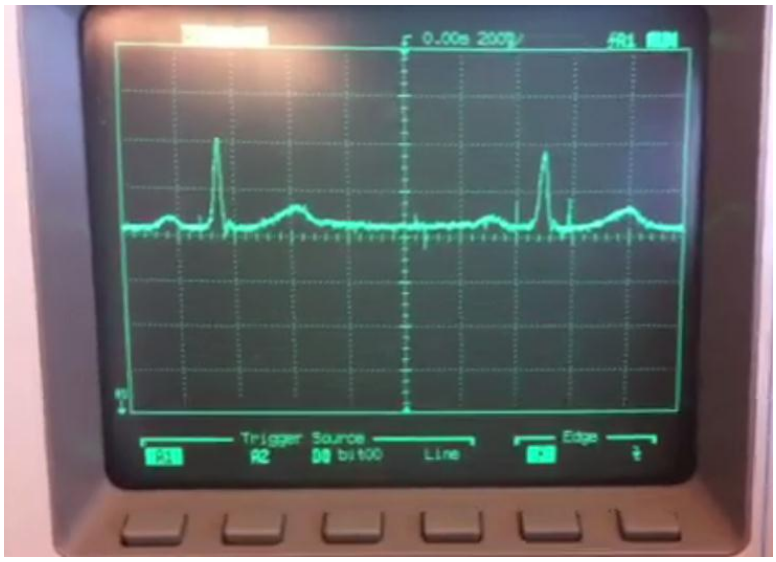
Mikrokontrollerin ohjelmistokehitystä varten käytössä oli Ubuntu 10.04 -käyttöjärjestelmällä varustettu työasema, nRF6310-kehitysalusta + nRF51822 Development kit -moduuli sekä J-Link Lite CortexM JTAG/SWD -emulaattori. Tämän lisäksi koko järjestelmän testaamista varten oli kehitetty potilassimulaattori, joka on nähtävissä kuvassa 12.



Kuva 12. Potilassimulaattori.

Potilassimulaattorin tehtävänä oli syöttää analogista EKG-signaalia nRF51822-piirille EKG-anturin tapaan. Se koostui noin luottokortin kokoisesta Debian Linux -käyttöjärjestelmällä varustetusta Raspberry PI -tietokoneesta, 12-bittisestä MCP4821 D/A-muuntimesta sekä MIT-BIH-tietokannan (Massachussetts Institute of Technology-Beth Israel Hospital) digitaalisista EKG-tallenteista.

Potilassimulaattori toimi siten, että Raspberry PI syötti D/A-muuntimelle EKG-tallenteen yksittäisiä näytteitä 360 kertaa sekunnissa, jolloin D/A-muunnin tuotti ulostuloonsa kuvan 13 kaltaista EKG-signaalia, josta on selkeästi erotettavissa sydämen lyönnin eri vaiheiden aiheuttamat aallot.



Kuva 13. Potilassimulaattorin tuottama EKG-signaali.

Potilassimulaattorin kehitykseen oli päädytty, koska projektin yhteistyökumppani ei ollut saanut kehitettyä EKG-anturiin tarvittavaa esivahvistinta insinöörityön kannalta tarpeeksi nopealla aikataululla. Tämän lisäksi potilassimulaattorin etuna EKG-anturiin verrattuna oli se, että EKG-anturin olisi pitänyt olla koko kehitysprosessin ajan kiinni ihmisessä, jotta kehitettävää laitetta olisi voitu testata. Potilassimulaattori mahdollisti myös sydänsairauksista kärsivien ihmisten EKG-tallenteiden toiston, jolloin voitiin varmistua, että insinöörityön aiheena oleva laite toimi luotettavasti myös heillä.

iOS-sovelluskehitystä varten käytössä oli projektia varten hankittu Mac Mini -työasema sekä Iphone 5 -älypuhelin iOS 6.1-käyttöjärjestelmällä.

7.3 Muut valmistelut

Koska projektissa ei käytetty Nordic Semiconductorin tukemaa Keil MDK-kehitysympäristöä, tuli Nordic Semiconductorin tarjoamat laitekirjastot saattaa GCC-kääntäjän ymmärtämään muotoon ennen kuin kirjaston kaikkia osia oli mahdollista käyttää. Tämä edellytti muun muassa jokaisen SVCall-järjestelmäkutsun muuntamisen GCC-yhteensopivaksi.

Laitekirjastoihin tehtyjen muutosten lisäksi C-standardikirjaston toteuttava Newlib-kirjasto saatettiin osittain yhteensopivaksi nRF51822-piirin kanssa. Newlib-kirjastoa

tarvittiin, jotta saatiin tuki standardi sisään- ja ulostuloille, jotka mahdollistivat esimerkiksi printf-funktion käytön siten, että se tulosti ulostulonsa UARTin välityksellä työaseman sarjaporttiin.

7.4 Mikrokontrollerin ohjelmoiminen

Kuten luvussa 5.5 mainittiin, mikrokontrollerin ohjelmoiminen tapahtuu GDB:n, J-Link GDB serverin sekä Segger J-Link Lite CortexM-9 JTAG/SWD -emulaattorin avulla. Emulaattorin ollessa liitettynä työaseman ja nRF51822-piirin väliin, siihen luodaan yhteys J-Link GDB serverillä komentorivikomennolla:

```
~$ ./JLinkGDBServer -device nRF51822 -speed 1000 -if
SWD
```

Komennossa määritellään device-parametrilla piirin tyyppi, speed-parametrilla käytettävä nopeus, joka on nRF51822-piirin tapauksessa 1000 kHz ja if-parametrilla käytössä oleva liitäntä.

Komennon suorittamisen jälkeen tulee avata toinen komentorivi-ikkuna, jossa GDB käynnistetään ja yhdistetään J-Link GDB serveriin komennoilla:

```
$ arm-none-eabi-gdb

(gdb) target remote localhost: 2331
```

Ensimmäinen komento käynnistää GDB:n ja toinen yhdistää GDB:n J-Link GDB serveriin.

Bluetooth low energyä hyödyntävän ohjelman suorittaminen nRF51822-piirillä edellyttää kahden eri tiedoston ohjelmoimista nRF51822-piirin flash-muistille. Ensimmäinen tiedosto sisältää Nordic Semiconductorin S110 SoftDevice -protokollapinon ja toinen kirjoitetun ohjelmakoodin, joka hyödyntää protokollapinoa.

Ennen kuin tiedostoja voidaan kirjoittaa flash-muistille, tulee flash-muisti tyhjentää ja tämän jälkeen asettaa kirjoitustilaan kirjoittamalla seuraavat komennot GDB:hen:

```
(gdb) set {int} 0x4001e504 = 0x02

(gdb) set {int} 0x4001e50c = 0x01

(gdb) set {int} 0x4001e504 = 0x01
```

Ensimmäinen komennoista kirjoittaa arvon 2 muistin osoitteeseen 0x4001e504, joka on NVCM (Non-volatile memory controller) -muistinohjaimen CONFIG-rekisterin osoite. Arvon 2 kirjoittaminen kyseiseen rekisteriin sallii ohjelmamuistin tyhjentämisen. Toinen komennoista kirjoittaa arvon 1 ERASEALL-rekisteriin, joka tyhjentää koko ohjelmamuistin. Kolmas komento kirjoittaa CONFIG-rekisteriin arvon 1, joka sallii ohjelmamuistiin kirjoittamisen. Kun edellä olevat komennot on suoritettu, tiedostot voidaan kirjoittaa muistiin load-komentoa hyväksikäyttäen:

```
(gdb) load S110_nRF51822_3.0.0-1.beta_softdevice.hex

(gdb) load main
```

Ensimmäinen komento ohjelmoi protokollapinon version 3 ohjelmamuistiin ja toinen komento itse ohjelman. Ohjelmoimisen jälkeen prosessori tulee resetoida joko irroittamalla ja uudelleenliittämällä virtaliitin, jolloin virheidenetsintäyhteys katkeaa ja ohjelma suoritetaan tai kirjoittamalla komento:

```
(gdb) monitor reset
```

Monitor reset-komennon jälkeen ohjelmakoodi voidaan suorittaa komennolla:

```
(gdb) continue
```

Koko ohjelmointiprosessi voidaan myös automatisoida luomalla .gdbinit-tiedosto, joka sisältää edellä mainitut komennot, samaan kansioon S110 SoftDevice -protokollapinon ja ohjelmakoodin kanssa. Tällöin GDB:tä käynnistettäessä GDB suorittaa tiedoston sisältämät komennot automaattisesti.

8 Työn toteutus

8.1 Mikrokontrollerin sovellusohjelma

Mikrokontrollerille kehitettävälle ohjelmalle oli asetettu vaatimukseksi vain se, että sen tuli selvittää analogisesta EKG-signaalista sydämen lyöntitiheys ja välittää se iPhonelle Bluetooth low energya hyväksikäyttäen. Mikrokontrollerin ohjelmiston kannalta tämä vaati neljä eri vaihetta, jotka olivat

- Bluetooth low energy -yhteyden muodostus iPhoneen
- analogisen EKG-signaalin näytteistäminen
- sydämen lyöntiheyden selvittäminen
- lyöntitiheyden lähettäminen iPhonelle.

Mikrokontrollerin ohjelmiston suunnittelussa tuli pohtia, miten nämä neljä vaihetta pystyttiin toteuttamaan mahdollisimman vähävirtaisesti. Virransäästön kannalta erityisen tärkeää oli, että laite ehti olemaan mahdollisimman pitkään lepotilassa, jolloin se kulutti vähemmän virtaa.

Ennen ohjelmiston suunnittelun aloittamista projektiin osallistuva henkilöstö kävi tutustumassa nRF51822-piiriin Nordic Semiconductorin Global tech tour 2012 -kiertueesen kuuluneessa tapahtumassa Kööpenhaminassa. Tapahtuma tarjosi uutta arvokasta tietoa nRF51822-piirin ominaisuuksista sekä siinä käytetyistä ratkaisuista.

Varsinainen ohjelmiston suunnittelu aloitettiin tutustumalla nRF51822-piirin SDK:hon, josta löytyi useita esimerkkikoodeja, jotka kuvasivat piirin eri lohkojen toimintaa. Esimerkeistä kävi ilmi, että useiden eri lohkojen toiminta perustuu tehtäviin (task) ja tapahtumiin (event). Tehtävillä ilmoitetaan lohkolle, että sen halutaan suorittavan jokin tietty tehtävä, kun lohko on suorittanut halutun tehtävän se vastaa tapahtumalla ilmoittaen, että tehtävä on suoritettu ja palaa IDLE-tilaan. Edellä mainitun kaltainen toimintamalli mahdollistaa sen, että lohkon tarvitsee olla aktiivisena vain operaation vaatiman ajan, jolloin se kuluttaa huomattavasti vähemmän virtaa verrattuna siihen, että se olisi koko ajan aktiivisena.

8.1.1 A/D-muunnoksen tekeminen

A/D-muunnoksella tarkoitetaan analogisen signaalin muuntamista digitaaliseksi näytteeksi. A/D-muunnoksen tekemistä varten nRF51822-piiri sisältää monipuolisen A/D-muuntimen, jonka asetuksia voidaan hallita A/D-muuntimen CONFIG-rekisterin kautta. CONFIG-rekisteristä voidaan muun muassa muuttaa A/D-muuntimen resoluutiota, referenssijännitteen lähdettä sekä analogisen signaalin syöttämiseen käytetty nasta.

Insinöörityössä päädyttiin käyttämään nRF51822-piirin AIN3-nastaa EKG-signaalin syöttämiseen. A/D-muunnoksen tekeminen edellytti, että A/D-muunnin asetettiin lukemaan kyseistä nastaa. nRF51822 mahdollisti myös syötettävän signaalin vaimentamisen, mutta vaimentamiselle ei ollut tarvetta, koska laitteen toiminnan testaamiseen käytetyn potilassimulaattorin ulostulojännite oli säädettävissä. Resoluutioksi valittiin 10 bittiä, jotta suoritetusta muunnoksesta saatiin mahdollisimman tarkka. Referenssijännitteeksi valittiin piiristä saatu 1,2 volttia, jotta erillistä ulkoista referenssijännitteen lähdettä ei tarvittu. Valittujen asetusten muuttaminen tehtiin nRF51822 SDK:n sisältämän laitekirjaston makroja hyödyntäen seuraavalla tavalla:

```
NRF_ADC->CONFIG = (ADC_CONFIG_RES_10bit <<
ADC_CONFIG_RES_Pos) |
(ADC_CONFIG_INPSEL_AnalogInputNoPrescaling <<
ADC_CONFIG_INPSEL_Pos) | (ADC_CONFIG_REFSEL_VBG <<
ADC_CONFIG_REFSEL_Pos) | (ADC_CONFIG_PSEL_AnalogInput3
<< ADC_CONFIG_PSEL_Pos) | (ADC_CONFIG_EXTREFSEL_None
<< ADC_CONFIG_EXTREFSEL_Pos)
```

Komento kirjoittaa A/D-muuntimen CONFIG-rekisteriin edellä mainitut asetukset sekä poistaa ulkoisen referenssijännitteen syöttönastat käytöstä. Asetusten määrittelyn jälkeen A/D-muunnoksen tekeminen onnistuu yksinkertaisimmillaan seuraavalla tavalla:

```
NRF_ADC->TASKS_START = 1;

while (!NRF_ADC->EVENTS_END);

int sample = NRF_ADC->RESULT;
```

Ensimmäisessä komennossa annetaan A/D-muuntimelle tehtäväksi uusi A/D-muunnos kirjoittamalla A/D-muuntimen TASKS_START-rekisteriin arvo yksi. A/D-muunnin ilmoittaa muunnoksen valmistumisesta tapahtumalla eli kirjoittamalla arvon 1 A/D-muuntimen EVENTS_END-rekisteriin, jota toisessa komennossa odotetaan. Kolmannessa komennossa juuri valmistunut näyte luetaan A/D-muuntimen RESULT-rekisteristä. Insinöörityössä päädyttiin kuitenkin käyttämään keskeytystä A/D-muuntimen arvon lukemiseen virrankulutuksen minimoimiseksi. Tämä vaati seuraavien komentojen käyttämisen alustuksen yhteydessä:

```
NRF_ADC->INTENSET = ADC_INTENSET_END_Msk;

nrf_nvic_SetPriority(ADC_IRQn, NRF_APP_PRIORITY_LOW);

nrf_nvic_EnableIRQ(ADC_IRQn);
```

Ensimmäinen komento asettaa A/D-muuntimen aiheuttamaan keskeytyksen, kun A/D-muunnos on valmis. Toinen komennoista määrittelee aiheutettavan keskeytyksen tärkeyden ja kolmas aktivoi keskeytyksen. Tämän jälkeen A/D-muunnoksen tekemiseksi tarvitsi käyttää vain komentoa

```
NRF_ADC->TASKS_START = 1;
```

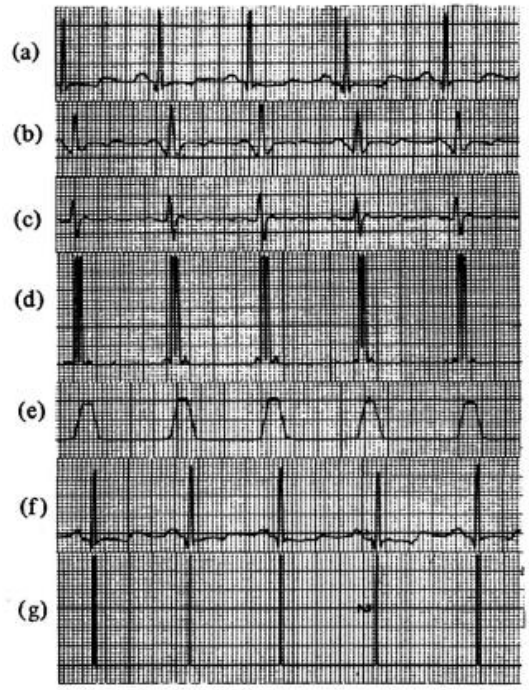
jolla käsketään A/D-muunninta aloittamaan muunnoksen suorittaminen. A/D-muunnoksen ollessa valmis A/D-muunnin aiheuttaa keskeytyksen, jolloin ohjelmakoodi siirtyy suorittamaan keskeytysfunktiota. Keskeytysfunktiossa A/D-muuntimelta voidaan lukea arvo normaalisti RESULT-rekisteristä.

8.1.2 Näytteistäminen ja sydämen lyöntitiheyden selvittäminen

Sydämen lyöntitiheyden tunnistamisessa päädyttiin käyttämään apuna EP Limitedin kehittämää avoimen lähdekoodin EKG-analyysisovellusta, jonka dokumentointi sekä lähdekoodi ovat ladattavissa EP Limitedin kotisivuilta [49]. EP Limitedin sovellus kykenee tunnistamaan EKG-signaalista yksittäisen sydämen lyönnin eri vaiheet sekä mahdolliset rytmihäiriöt. Sovelluksen sisältämää QRS-kompleksin tunnistusta on kehitetty noin viidentoista vuoden ajan. [53 s. 1–2.]

Sydämen lyöntitiheyden selvittäminen edellyttää sydämen lyöntien tunnistamista ja se onnistuu helpoiten jos sydämen lyönnistä onnistutaan tunnistamaan QRS-kompleksi. QRS-kompleksin tunnistamista varten sovelluksesta tarvittiin vain QRS-kompleksin tunnistus-funktio. Näitä sovelluksen lähdekoodi piti sisällään kolme. Yksi niistä oli PIC-mikrokontrollerille kirjoitettu tunnistin ja kaksi muuta olivat alustariippumattomia, eli ne eivät olleet sidottu mihinkään tiettyyn laitteistoon. Toinen näistä kahdesta oli dokumentoinnin mukaan paranneltu versio toisesta, ja se valittiin tämän takia käytettäväksi.

QRS-kompleksin tunnistin pohjautuu Pan-Tompkins -algoritmiin, jossa EKG-signaali muun muassa suodatetaan neljän digitaalisen suodattimen läpi, jonka jälkeen siitä päätellään QRS-kompleksien paikat. [53 s. 3–6.]



Kuva 14. Signaali Pan-Tompkins algoritmin suodatuksen eri vaiheissa. [54 s. 231]

Kuvasta 14 nähdään alkuperäisen Pan-Tompkins -algoritmin suodattimien toiminnan eri vaiheet ylhäältä alaspäin. A-kohdassa on alkuperäinen EKG-signaali ennen suodatusta. B-kohdassa signaali on suodatettu 5–12 Hz:n kaistanpäästösuodattimella, jonka jälkeen se on derivoitu (kohta c). Derivoinnin jälkeen näytteistetty signaali on korotettu toiseen potenssiin (kohta d), jonka jälkeen siitä on otettu liukuva keskiarvo 150 millisekunnin aikaikkunalla (kohta e). G-kohdassa signaali on kulkenut koko Pan-Tompkins -algoritmin läpi, pystyviivat kuvaavat signaalista löytyneitä QRS-komplekseja. G-

kohdan pystyviivoja voidaan verrata suodattamisen aiheuttamalla viiveellä viivästettyyn alkuperäiseen signaaliin (kohta f) ja todeta, että algoritmi on tunnistanut QRS-kompleksit EKG-signaalista.

EP Limitedin QRS-kompleksin tunnistimen toteutuksessa Pan-Tompkins -algoritmia on paranneltu suorituskyvyn osalta siten, että liukuvan keskiarvon aikaikkunaa on pienennetty 80 millisekuntiin. Tämän lisäksi algoritmin vahvistusherkkyyden (gain sensitivity) pienentämiseksi algoritmia on muokattu ottamalla näytteistä itseisarvo alkuperäisessä Pan-Tompkins -algoritmissa käytetyn potenssiin korottamisen sijaan. [53, s.6.]

QRS-kompleksin tunnistinta käytetään kutsumalla QRSDet-funktiota, joka saa kaksi parametria. Ensimmäinen parametri on näyte, joka halutaan tutkia ja toista käytetään funktion staattisten muuttujien alustamiseen tarvittaessa. Funktio palauttaa QRS-kompleksin tunnistettuaan, kompleksin tunnistamiseen kuluneen ajan eli viiveen, jos QRS-kompleksia ei tunnistettu se palauttaa nollan.

QRS-kompleksin tunnistin on dokumentoinnin mukaan optimoitu 200 Hz näytteenottotaajuudelle ja sen tunnistustarkkuus heikkenee, jos näytteidenottotaajuus eroaa siitä. 200 Hz näytteenottotaajuuden saavuttamiseksi näytteitä tulee ottaa 5 millisekunnin välein, tämä saavutettiin Nordic Semiconductorin laitekirjastosta löytyvällä application timer -ajastimella.

Application timerin etuna normaaliin ajastimeen verrattuna oli se, että se mahdollisti useiden RTC1-reaaliaikakelloa hyödyntävien ajastimien asettamisen. Normaalia ajastinta käytettäessä useamman ajastimen luominen vaatisi useamman ajastinlohkon käyttämistä, joka lisäisi virrankulutusta. Uusi application timer luotiin komennolla

```
app_timer_create(&s_acquisition_timer_id,
APP_TIMER_MODE_REPEATED, acquisition_timeout_handler);
```

jossa ensimmäinen parametri oli ajastimen tunnistamiseen käytetyn app_timer_id_t-tyyppisen muuttujan osoite. Toinen parametri kertoi, että ajastimen oli tarkoitus laueta toistuvasti. Kolmas parametri oli funktio-osoitin, joka osoitti funktioon, jota kutsuttiin, kun ajastin laukeaa. Luomisen jälkeen application timer-ajastin oli mahdollista käynnistää komennolla

```
app_timer_start(s_acquisition_timer_id,      ACQUISITI-
ON_INTERVAL, NULL);
```

jossa ensimmäinen parametri oli ajastimen tunnistamiseen käytetty app_timer_id_t-tyyppinen muuttuja. Toinen parametri kertoi ajastimen intervallin. Kolmantena parametrina olisi voitu käyttää yleiskäyttöistä osoitinta, jolla olisi voitu välittää tietoa ajastimen keskeytysfunktioon. Tälle ei ollut kuitenkaan tarvetta, joten funktiolle välitettiin nolla-osoitin.

Sydämen lyöntiheyden selvittämiseksi ajastin asetettiin laukeamaan 200 kertaa sekunnissa. Ajastimen keskeytysfunktiossa otettiin näyte A/D-muuntimella ja se välitettiin QRSDet-funktiolle. QRSDet-funktion paluuarvosta voitiin päätellä, löytyikö signaalista QRS-kompleksi, jos löytyi, senhetkinen application timerin arvo tallennettiin staattiseen muuttujaan. Seuraavan QRS-kompleksin löytyessä edellisellä kerralla tallennettu laskurin arvo siirrettiin toiseen staattiseen muuttujaan ja alkuperäiseen muuttujaan tallennettiin uusi application timerin laskurin arvo. Näiden kahden muuttujaan tallennetun arvon perusteella voitiin laskea sydämen lyöntitiheys seuraavalla kaavalla

```
bpm = 60 x application timerin kellotaajuus / ( nyk.
laskurin arvo – edel. laskurin arvo)
```

Kaava antaa tulokseksi sydämen lyöntien määrän minuutissa, joka voidaan tämän jälkeen lähettää iPhonelle.

8.1.3 Bluetooth low energy -yhteyden luominen ja datan lähetys

Bluetooth low energy -yhteyden luomiseen käytettiin apuna nRF51822 SDK:sta \Board\nrf6310\ble-kansiosta löytyvää esimerkkiohjelmaa, joka on nimeltään ble_app_hrs. Ohjelman tarkoituksena on demonstroida GATT-spesifikaatiossa esiteltyä HRS (Heart rate service) -palvelua, joka on tarkoitettu laitteille, jotka mittaavat sydämen sykettä.

ble_app_hrs-esimerkkisovelluksesta selvisi yhteyden muodostukseen tarvittavat vaiheet, jotka olivat

- Bluetooth low energy -protokollapinon tapahtumankäsittelijän alustaminen

- GAP-parametrien alustaminen
- mainostus (advertising) -parametrien alustaminen
- HRS-palvelun alustaminen
- yhteyden neuvottelun parametrien alustaminen
- mainostuksen aloittaminen.

Protokollapinon tapahtumankäsittelijän alustus tuli tehdä, jotta protokollapino pystyy kutsumaan tapahtumankäsittelijäksi määriteltyä funktiota aina kun protokollapinossa tapahtuu jotain ohjelman toiminnan kannalta merkittävää, kuten vaikkapa yhteyden muodostus tai katkeaminen.

Luvussa 2.2 mainittiin GAP-protokollan vastaavan muun muassa uusiin laitteisiin yhdistämisestä sekä protokollapinon yläpuolella olevista profiileista. Tästä syystä GAP-protokollalle tuli määritellä yhdistämiseen ajoitukseen liittyviä asetuksia sekä esimerkiksi laitteen nimi ja tyyppi ennen kuin laitteeseen oli mahdollista luoda yhteys.

Mainostukseen liittyvistä asetuksista tuli määritellä muun muassa laitteen tukemat bluetooth-tekniikat, eli nRF51822-piirin tapauksessa vain bluetooth low energyä. Tämän lisäksi määriteltiin, että mainostuksessa käytetään laitteen koko nimeä eikä esimerkiksi vain osaa siitä. Tämän jälkeen määritetyt asetukset enkoodattiin ja lähetettiin protokollapinolle.

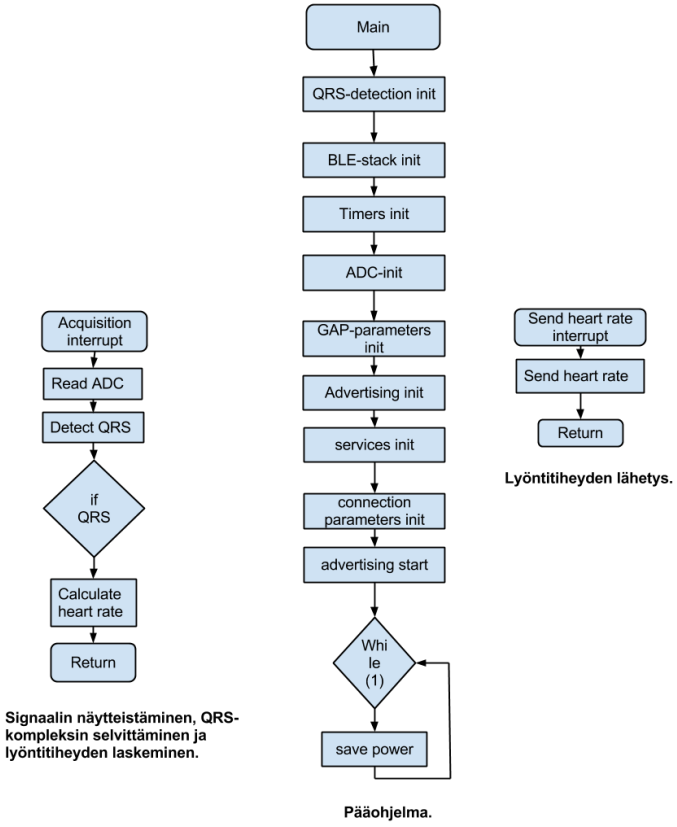
HRS-palvelun alustus tapahtui määrittelemällä EKG-anturin sijainti keholla ja sen tukemat ominaisuudet, kuten esimerkiksi anturin irtoamisen ilmoittaminen. Tämän lisäksi määriteltiin mitä tietoja palvelun kautta sai lukea.

Lopuksi ennen kuin laitetta voitiin alkaa mainostamaan, tuli määritellä yhteyden neuvottelussa käytetyt asetukset. Näissä asetuksissa määriteltiin muun muassa, että yhteyden muodostamisessa käytetyt asetukset haetaan yhdistettävältä laitteelta ja että yhteyttä ei katkaista, jos yhteydessä tapahtuu virhe.

Edellä mainittujen vaiheiden jälkeen yhteyden muodostaminen voitiin aloittaa aloittamalla laitteen mainostaminen muille laitteille. Mainostamisen aloittamisen jälkeen voitiin mennä virransäästötilaan odottamaan, että toinen laite yritti muodostaa yhteyttä. Tämän jälkeen jos yhteyden muodostus onnistui, protokollapino kutsuu sille määriteltyä

tapahtumankäsittelijää. Tapahtumankäsittelijässä yhteyden tilaksi muutettiin yhdistetty ja siinä myös käynnistettiin ohjelman application timer -ajastimet.

Yhteyden muodostuksen jälkeen ohjelmalta lähetettiin sydämen syke iPhonelle sekunnin välein. Lähetystä varten ohjelmaan luotiin uusi application timer -ajastin. Application timerin luonti tapahtui samaan tapaan kuin luvussa 8.1.2. Ainoana erona oli, että ajastin asetettiin laukeamaan sekunnin välein. Juuri luodun ajastimen keskeytysfunktion ainoa tehtävä oli lähettää sydämen lyöntitiheys ble_hrs_heart_rate_measurement_send-komennolla.



Kuva 15. Mikrokontrollerin sovelluksen vuokaavio.

Kuvassa 15 on sovelluksen vuokaavio, josta on nähtävissä ohjelman toiminnan pääpiirteet. Vuokaaviossa vasemmalla on näytteistämiseen käytetty keskeytysfunktio, joka suoritetaan 200 kertaa sekuntissa näytteistämiseen käytetyn application timerin lauetessa. Oikealla on nähtävissä sydämen lyöntitiheyden lähettämiseen käytetty keskeytysfunktio, joka suoritetaan kerran sekunnissa sydämen sykkeen lähettämiseen tarkoitetun application timerin lauetessa. Keskellä on nähtävissä ohjelman main-funktio, jon-

ka tehtävänä ei ole alustusten ja laitteen mainostamisen jälkeen tehdä mitään muuta kuin vain nukkua keskeytysten välillä.

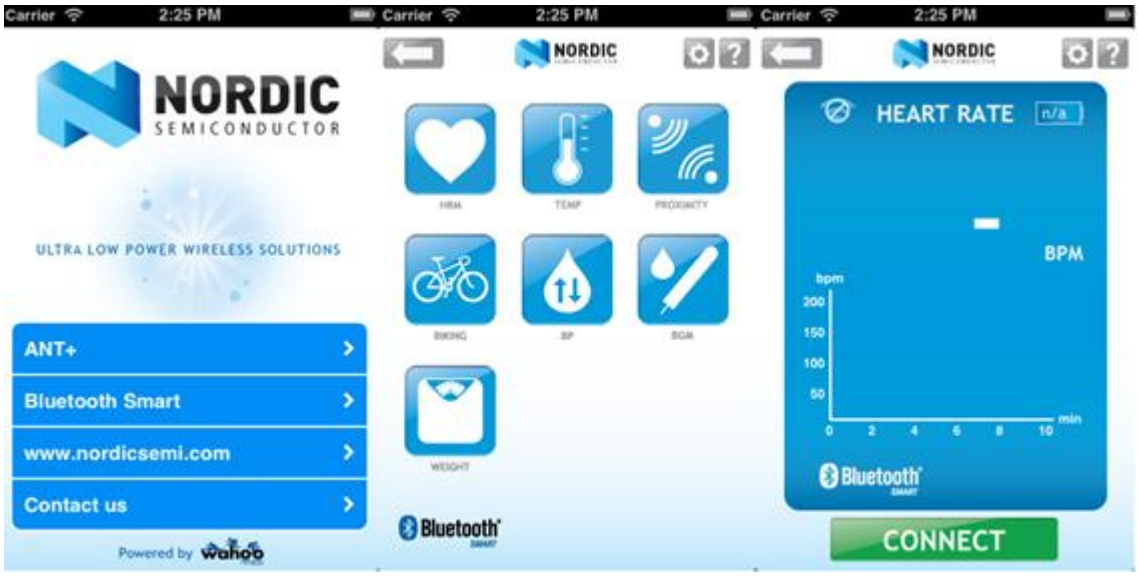
8.2 Sovellusohjelma iPhonelle

Iphone-sovellukselle oli asetettu vaatimukseksi, että sen tuli esittää nRF51822-piirin lähettämän sydämen lyöntitiheys ja piirtää siitä kuvaaja, joka kuvasi sydämen lyöntiiheyttä ajan funktiona. Tämä vaati iPhone-sovelluksen toiminnalta vähintään neljä eri vaihetta, jotka olivat

- Bluetooth low energy -yhteyden muodostus nRF51822-piiriin
- sydämen lyöntitiheyden vastaanottaminen
- sydämen lyöntitiheyden esittäminen
- kuvaajan piirtäminen.

Nordic Semiconductor tarjosi vastaavan toiminnallisuuden sisältämän nRFready iOS Demo App -esimerkkisovelluksen, josta päätettiin karsia pois turha toiminnallisuus. Sovelluksen käyttöliittymää myös yksinkertaistettiin huomattavasti. Kuten jo luvussa 6 mainittiin, sovelluskehitys tapahtui Applen Xcode IDEllä.

Sovelluksen muokkaamista varten nRFready iOS Demo App ladattiin Nordic Semiconductorin kotisivuilta. Lataamisen jälkeen se purettiin ja avattiin Xcodessa. [55.]



Kuva 16. Alkuperäinen sovellus ennen muokkausta.

Kuvasta 16 on nähtävissä muokkaamattoman sovelluksen valikkorakenne, josta on havaittavissa, että se sisälsi näkymät painon, sydämen lyöntitiheyden, lämpötilan sekä monien muiden arvojen esittämiseen. Jokaiseen näkymään siirtyminen ohjelman päänäkymästä kuitenkin vaati useamman näytön painalluksen, ja tästä haluttiin päästä eroon. Ohjelman ainoaksi näkymäksi haluttiin sydämen lyöntitiheyden osoittama näkymä, jonka takia sovelluksen MainWindow.xib-tiedosto avattiin lähdekoodieditorissa ja sieltä vaihdettiin jokainen RootViewController-viittaus osoittamaan HeartrateSimpleViewController-näkymään. Tämän jälkeen sovellus käynnistyi suoraan sydämen lyöntitiheyden esittämään näkymään.

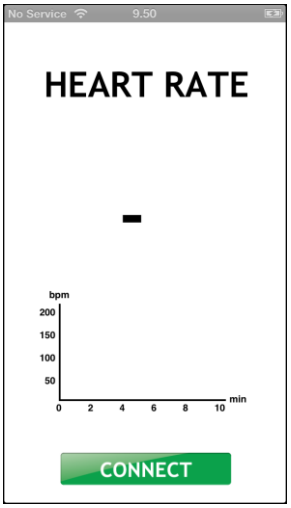
Sovellusta iPhonessa testatessa havaittiin, että vaikka sovellus käynnistyi sydämen lyöntitiheyden esittävään näkymään, se ei enää yhdistänyt itseään nRF51822-piiriin. Pitkän lähdekoodin tutkinnan jälkeen havaittiin, että tämä johtui siitä, että näkymää ei kutsuttu initWithNibName-funktiolla, jonka takia se ei alustanut tarvittuja parametreja. Tämä korjattiin siirtämällä initWithNibname-funktion sisältö näkymää ladattaessa suoritettavan viewDiDLoad-funktion alkuun.

Seuraavaksi sovelluksesta poistettiin sovelluksessa käytössä ollut mukautettu navigointipalkki, josta oli ollut mahdollista siirtyä edelliseen näkymään sekä asetuksiin. Tämä tapahtui poistamalla koodista self buildCustomNavbar-metodin kutsu. Ohjelmaa testattaessa kuitenkin huomattiin, että sovellus oli korvannut navigointipalkin iOSin

omalla navigointipalkilla. Tästä päästiin eroon kutsumalla metodia [self.navigationController setNavigationBarHidden: YES].

Alkuperäinen sovellus oli kehitetty iPhone 4s -puhelimelle, jonka takia se ei käyttänyt iPhone 5 -puhelimen isompaa näyttöä vaan jätti sovelluksen ylä- ja alapuolella mustat palkit. Tämä korjattiin interface builderissa muuttamalla sydämen lyöntitiheyden näkymän asetuksista näytön kooksi ”Retina 4 Full Screen” ja tämän jälkeen lisäämällä projektin asetussivulta 4 tuumaiselle retina-näytölle ohjelman käynnistyessä näytettävä kuva (Launch image), jolloin iPhone 5 avaa näkymän koko ruudulla.

Edellä mainittujen vaiheiden jälkeen voitiin aloittaa käyttöliittymän turhien elementtien poistaminen. Sydämen lyöntitiheyden lähettävä nRF51822-piirin ohjelmisto ei esimerkiksi lähetä akun varausta iPhonelle, joten sen näyttäminen on täysin turhaa. Elementtien poisto tapahtui interface builderista valitsemalla poistettavat elementit ja painamalla delete-näppäintä, jonka jälkeen lähdekoodista poistettiin viittaukset näihin elementteihin. Viimeisessä vaiheessa jäljelle jääneet elementit siirrettiin halutuille paikoille ja elementtien väri muutettiin mustaksi, jotta ne erottuivat valkoisesta taustasta.



Kuva 17. Sovelluksen käyttöliittymä muutosten jälkeen.

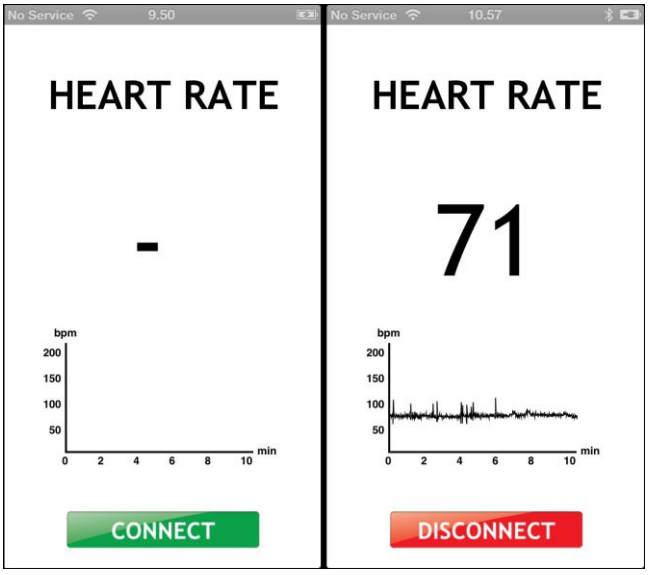
Kuvassa 17 on nähtävissä sovelluksen käyttöliittymä ylimääräisten ominaisuuksien karsimisen ja muiden muutosten jälkeen. Lopputulos on huomattavasti alkuperäistä yksinkertaisempi, eikä siinä tarvita enää ylimääräisiä näytön painalluksia näkymään pääsemiseksi.

9 Järjestelmän testaus

Järjestelmän luotettavan toiminnan varmistamiseksi järjestelmä tuli testata huolellisesti. Järjestelmän testaukseen käytetty testikokoonpano koostui luvussa 7.2 esitellystä potilassimulaattorista, työasemasta sekä itse järjestelmästä eli iPhone 5:stä sekä nRF6310-kehitysalustasta, johon oli kiinnitetty nRF51822 development kit- moduuli.

Ennen järjestelmän testaamista nRF6310-kehitysalusta tuli liittää työasemaan ja siihen liitettyyn nRF51822 development kit-moduuliin tuli ohjelmoida S110 SoftDevice-protokollapino sekä itse ohjelmakoodi luvussa 7.4 esitetyllä tavalla, jonka jälkeen potilassimulaattori voitiin liittää nRF51822-piirin AIN3- ja GND-nastojen väliin. nRF51822:lle tehtyjen valmistelujen lisäksi iPhonelle kehitetty sovellus piti siirtää Xcodella iPhonen muistiin, jonka jälkeen voitiin aloittaa järjestelmän testaaminen.

Testaus suoritettiin useita kertoja siten, että iPhone-sovelluksella muodostettiin Bluetooth low energy yhteys nRF51822-piiriin (kuvassa 18 vasemmalla), jonka jälkeen potilassimulaattorilla syötettiin nRF51822-piirin analogiseen sisääntulon AIN3-nastaan EKG-signaalia. Signaalin syötön alettua iPhone-sovelluksesta seurattiin nRF51822-piirin lähettämiä lukuarvoja ja kuvaajan piirtymistä (kuvassa 18 oikealla).



Kuva 18. iPhone-sovellus ennen yhteyden muodostamista ja yhteyden muodostamisen jälkeen, kun sydämen lyöntitiheyttä on lähetetty yli 10 minuuttia.

nRF51822-piirin lähettämän radiosignaalin kantamaa arvioitiin siirtämällä iPhonea kauemmas piiristä ja samalla seuraamalla piirin lähettämiä lukuarvoja iPhonen näytöltä, jos sovelluksen esittämässä sydämen lyöntitiheydessä ei esiintynyt pientäkään variaatiota, voitiin olettaa, että nRF51822-piirin lähettämä tieto ei ollut saavuttanut iPhonea, jonka takia sovelluksen esittämä sydämen lyöntitiheys ei päivittynyt.

Suoritettujen testien perusteella voitiin todeta tiedon lähetyksen ja vastaanoton toimivan luotettavasti. Tiedon vastaanotossa esiintyi ongelmia ainoastaan silloin kun iPhone siirrettiin yli 10 metrin päähän nRF51822-piiristä, jolloin sydämen lyöntitiheyden päivitys iPhonessa lakkasi. Ongelma kuitenkin korjautui heti kun iPhone tuotiin lähemmäs nRF51822-piiriä.

Järjestelmän selvittämän sydämen lyöntitiheyden tarkkuudesta ei voitu saada täyttä varmuutta, koska saatavilla ei ollut minkäänlaista referenssiä, johon selvitettyjä arvoja olisi voitu verrata. Kuitenkin esimerkiksi kuvasta 18 on nähtävissä, että järjestelmän selvittämä lyöntitiheys on suurimman osan ajasta noin 70 lyöntiä minuutissa, joka on sydämen normaalin lyöntitiheyden (45–80 lyöntiä minuutissa) alueella, joten voidaan olettaa, että arvo on melko tarkka [56].

10 Yhteenveto

Insinöörityön tavoitteena oli suunnitella ja toteuttaa järjestelmä, joka selvitti EKG-anturilta saadusta signaalista sydämen lyöntitiheyden ja välitti sen Bluetooth low energyn avulla iPhone 5:lle, jossa se esitettiin tarkoitusta varten luodulla sovelluksella. Insinöörityö toteutettiin osana Metropolia Ammattikorkeakoulun tutkimus- ja kehitysyksikkö Electrian sekä Tampereen teknillisen yliopiston HealthSens-projektia ja siihen oli valittu käytettäväksi Nordic Semiconductorin erittäin vähävirtainen nRF51822-piiri.

Työn toteuttaminen edellytti nRF51822-piiriin, Bluetooth low energyyn, elektrokardiografiaan, iOS-sovelluskehitykseen sekä työssä käytettyihin työkaluihin perehtymistä. Työssä hyödynnettiin Nordic Semiconductorin laitekirjastoja sekä EP Limitedin QRS-kompleksin tunnistinta.

Työn haastavimmaksi osuudeksi osoittautui iPhone-sovelluksen kehitys, johon jäi hankittujen laitteiden toimitusten viivästymisen takia huomattavasti vähemmän aikaa kuin

oli alun perin suunniteltu. Laitteiden toimitusten viivästymisen lisäksi ongelmia esiintyi järjestelmän testauksessa, koska testaamista varten ei ollut saatavilla EKG-anturia. Ongelman ratkaisemiseksi testaamista varten kehitettiin potilassimulaattori. Haasteista huolimatta insinöörityö onnistui, ja sen lopputuloksena syntyi toimiva järjestelmä, joka täytti sille asetetut vaatimukset.

Insinöörityön aikana kehitettyä järjestelmää tullaan jatkokehittämään HealthSens-projektin tavoitteiden mukaisesti. Järjestelmää varten tullaan muun muassa luomaan oma piirilevy, jossa kehitetyn sovelluksen tulisi toimia ilman muutoksia. Siihen tullaan myös liittämään EKG-signaalin mittaamiseen tarvittava esivahvistin, jonka jälkeen se on siirrettävissä taipuisalle piirilevylle ja loppuen lopuksi laastarin muotoon.

Lähteet

1 Low Energy. 2013. Verkkodokumentti. Bluetooth SIG. <http://www.bluetooth.com/Pages/low-energy.aspx> Luettu 2.4.2013.

2 Bluetooth low energy. 2013. Verkkodokumentti. CSR plc. <https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=227336> Luettu 2.4.2013.

3 Joe Decuir. 2010. Bluetooth 4.0: Low Energy. Verkkodokumentti. CSR plc. <http://chapters.comsoc.org/vancouver/BTLER3.pdf> Luettu 2.4.2013.

4 The Smart Way to Connect. 2013. Verkkodokumentti. Bluetooth SIG. <http://www.bluetooth.com/Pages/Smart-Logos.aspx> Luettu 3.4.2013.

5 Rajput Kansingh. 2008. A Seminar Report on “Wibree – Ultra Low Power Radio Technology for Small Devices”. Verkkodokumentti. Sarvajanik College of Engineering & Technology. <http://123seminarsonly.com/Seminar-Reports/039/74558261-Wibree-ULP-BT-Word.pdf> Luettu 3.4.2013.

6 Wibree. 2012. Verkkodokumentti. Nokia Developer. <http://www.developer.nokia.com/Community/Wiki/Wibree> 31.7.2012. Luettu 3.4.2013.

7 Chen, J.C. 2012. Bluetooth 4.0. Verkkodokumentti. Federal communications commission. <http://transition.fcc.gov/bureaus/oet/ea/presentations/files/apr12/3d.-Apr-12-BT_TETRA-JC.pdf> Huhtikuu 2012. Luettu 3.4.2013.

8 Bluetooth low energy: Getting started. 2011. Bluegiga. <http://www.glynstore.com/content/docs/bluegiga/BLE_getting_started.pdf> 15.5.2011. Luettu 3.4.2013.

9 Bluetooth low energy software Developer's Guide. 2013. Verkkodokumentti. Texas Instruments. <http://www.ti.com/lit/ug/swru271d/swru271d.pdf> 4.1.2013. Luettu 3.4.2013.

10 Bluetooth core specification: Core Version 4.0. 2010. Verkkodokumentti. Bluetooth SIG. <https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737> 30.6.2010. Luettu 3.4.2013.

11 Gomez C., Oller J., Paradells J. 2012. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. Verkkodokumentti. MDPI. <http://www.mdpi.com/1424-8220/12/9/11734/pdf> 29.8.2012. Luettu 3.4.2013.

12 Mikhail Galeev. 2011. Bluetooth 4.0: An introduction to Bluetooth Low Energy — Part I. Verkkodokumentti. EETimes. <http://www.eetimes.com/design/communications-design/4217866/Bluetooth-4-0--An-introduction-to-Bluetooth-Low-Energy-Part-I> 14.7.2011. Luettu 3.4.2013.

13 Lynch, Jamel Pleasant. 2002. Co-Channel Interference In Bluetooth Piconets. Verkkodokumentti. <http://scholar.lib.vt.edu/theses/available/etd-11182002-115502/unrestricted/Chapter2.pdf> 10.6.2002. Luettu 3.4.2013.

14 Eskola, Katja. 2010. 12-kanavaisen Lepo-EKG:n laadukas rekisteröintitekniikka-dvd. Verkkodokumentti. <http://urn.fi/URN:NBN:fi:amk-2010120517153> syksy 2010. Luettu 5.4.2013.

15 Sagulin, Piia. 2009. V4-rintakytkennän sijoittelun merkitys 12-kytkentäisen lepo-EKG:n rekisteröinnissä naisilla. Verkkodokumentti. <http://urn.fi/URN:NBN:fi:amk-200912107552> 8.12.2009. Luettu 5.4.2013.

16 Salmela, Niina. 2011. EKG-käyrän rekisteröinti - Hoitajien EKG-käytän rekisteröintiosaaminen. <http://urn.fi/URN:NBN:fi:amk-201105249457> 2.5.2011. Luettu 5.4.2013.

17 Kuja-Aro, S., Mantonen, V. 2012. Lepo-EKG:n itseopiskelumateriaali; Verkkokurssi Päijät-Hämeen sosiaali- ja terveysyhtymän hoitohenkilökunnalle. Verkkodokumentti. <http://urn.fi/URN:NBN:fi:amk-2012112917179> Lokakuu 2012. Luettu 6.4.2013.

18 Sydämen johtoratajärjestelmä kuva. 2013. Verkkokuva. <http://rytmihairio.net/images/johtorata.gif> Luettu 6.4.2013.

19 Sinusrytmikuva. 2013. Verkkokuva. Wikipedia. <http://upload.wikimedia.org/wikipedia/commons/thumb/9/9e/SinusRhythmLabels_fi.svg/500px-SinusRhythmLabels_fi.svg.png> Luettu 6.4.2013.

20 nRF51822. 2013. Verkkodokumentti. Nordic Semiconductor. <http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF51822> Luettu 17.3.2013.

21 nRF51 Series Reference Manual. 2012. Nordic Semiconductor.

22 Serial Wire Debug. 2013. Verkkodokumentti. ARM Ltd. <http://www.arm.com/products/system-ip/debug-trace/coresight-soc-components/serial-wire-debug.php> Luettu 16.3.2013.

23 nRF51822 S110 SoftDevice Specification. 2012. Nordic Semiconductor.

24 nRF51822 Evaluation Kit. 2013. Verkkodokumentti. Nordic Semiconductor. <http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF51822-Evaluation-Kit> Luettu 18.3.2013.

25 nRF51822 Development Kit. 2013. Verkkodokumentti. Nordic Semiconductor. <http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF51822-Development-Kit> Luettu 18.3.2013.

26 Awais, Imran. 2012. New Benchmark Shows iPhone 5 Processor Clocked At 1.3GHz, Faster Than Previously Reported. Verkkodokumentti. Redmondpie. <http://www.redmondpie.com/new-benchmark-shows-iphone-5-processor-clocked-at-1.3ghz-faster-than-previously-reported/> 27.9.2012. Luettu 15.4.2013.

27 iPhone 5 tekniset tiedot. 2013. Verkkodokumentti. Apple Inc. <https://www.apple.com/fi/iphone/specs.html> Luettu 15.4.2013.

28 iOS 6 - Maailman kehittynein mobiilikäyttöjärjestelmä. 2013. Verkkodokumentti. Apple Inc. <http://www.apple.com/fi/ios/what-is/> Luettu 15.4.2013.

29 App Store Tops 40 Billion Downloads with Almost Half in 2012. 2013. Verkkodokumentti. Apple Inc. <http://www.apple.com/pr/library/2013/01/07App-Store-Tops-40-Billion-Downloads-with-Almost-Half-in-2012.html> 7.1.2013. Luettu 15.4.2013.

30 The History of Objective-C. 2011. Verkkodokumentti. Techtopia <http://www.techotopia.com/index.php/The_History_of_Objective-C> 31.3.2011. Luettu 15.4.2013.

31 Manish Kumar Srivastava. Bluetooth Low Energy for Android Platform. 2013. Verkkodokumentti. <http://www.tekritisoftware.com/bluetooth-low-energy-android-platform> Luettu 15.4.2013.

32 C Programming - What you need before you can learn. 2013. Verkkodokumentti. <http://en.wikibooks.org/wiki/C_Programming/What_you_need_before_you_can_learn> 22.1.2013. Luettu 16.4.2013.

33 Keil Evaluation Software Limitations. 2013. Verkkodokumentti. AMR Ltd. <http://www.keil.com/demo/limits.asp> Luettu 16.4.2013.

34 Introduction to cross-compiling for Linux. 2012. Verkkodokumentti. <http://landley.net/writing/docs/cross-compiling.html> 8.11.2012. Luettu 15.4.2013.

35 Paul, Ryan. 2011. Two decades of productivity: Vim's 20th anniversary. Verkkodokumentti. Arsctechnica. <http://arstechnica.com/information-technology/2011/11/two-decades-of-productivity-vims-20th-anniversary/> 2.11.2011. Luettu 16.4.2013.

36 Guckes, Sven. 2001. Mikä on Vim? Verkkodokumentti. <http://www.vim.org/6k/features.fi.txt> 30.3.2001. Luettu 15.4.2013.

37 GNU Make. 2010. Verkkodokumentti. Free Software Foundation Inc. <http://www.gnu.org/software/make/> 3.7.2010. Luettu 14.4.2013.

38 Using the GNU Compiler Collection. 2013. Verkkodokumentti. Free Software Foundation Inc. <http://gcc.gnu.org/onlinedocs/gcc-4.8.0/gcc.pdf> Luettu 14.4.2013.

39 Host/Target specific installation notes for GCC. 2013. Verkkodokumentti. Free Software Foundation Inc. <http://gcc.gnu.org/install/specific.html> 22.3.2013. Luettu 14.4.2013.

40 GCC, the GNU Compiler Collection. 2013. Verkkodokumentti. Free Software Foundation Inc. <http://gcc.gnu.org/> 12.4.2013. Luettu 14.4.2013.

41 Chamberlain, Steve. 1994. Using ld - The GNU linker. Verkkodokumentti. <http://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_mono/ld.html> Tammikuu 1994. Luettu 14.4.2013.

42 Linker Scripts. 2012. Verkkodokumentti. <http://sourceware.org/binutils/docs/ld/Scripts.html> 13.11.2012. Luettu 14.4.2013.

43 GDB: The GNU Project Debugger. 2013. Verkkodokumentti. Free Software Foundation Inc. <http://www.gnu.org/software/gdb/> 12.3.2013. Luettu 15.4.2013.

44 Using the gdbserver program. 2002. Verkkodokumentti. Free Software Foundation Inc. <http://ftp.gnu.org/old-gnu/Manuals/gdb-5.1.1/html_node/gdb_130.html> 14.2.2002. Luettu 15.4.2013.

45 Debugging with GDB. 2013. Verkkodokumentti. Free Software Foundation Inc. <www.sourceware.org/gdb/current/onlinedocs/gdb.html> Luettu 14.4.2013.

46 Apple Introduces Xcode, the Fastest Way to Create Mac OS X Applications. 2003. Verkkodokumentti. Apple Inc. <http://www.apple.com/pr/library/2003/06/23Apple-Introduces-Xcode-the-Fastest-Way-to-Create-Mac-OS-X-Applications.html> 23.6.2003. Luettu 15.4.2013.

47 Xcode. 2013. Verkkodokumentti. Apple Inc. <https://itunes.apple.com/us/app/xcode/id497799835> 15.4.2013. Luettu 16.4.2013.

48 Developer tools. 2013. Verkkodokumentti. Apple Inc. <https://developer.apple.com/technologies/tools/> Luettu 14.4.2013.

49 What's new in Xcode 4. 2013. Verkkodokumentti. Apple Inc. <https://developer.apple.com/technologies/tools/whats-new.html> Luettu 14.4.2013.

50 Hankinnat. 2011. Verkkodokumentti. Metropolia Ammattikorkeakoulu Oy. <https://tuubi.metropolia.fi/portal/fi/group/tuubi/henkilokunnalle/talousasiat/hankinta> Muokattu 7.9.2012. Luettu 12.4.2013.

51 Sopimustoimittaja. 2010. Verkkodokumentti. Metropolia Ammattikorkeakoulu Oy. <https://tuubi.metropolia.fi/portal/fi/group/tuubi/henkilokunnalle/talousasiat/sopimustoimittaja> Muokattu 4.3.2013. Luettu 15.4.2013.

52 EP Limited: Open Source ECG Analysis Software. 2008. Verkkodokumentti. EP Limited. <http://www.eplimited.com/software.htm> 16.1.2008. Luettu 15.4.2013.

53 Hamilton P.S. 2002. Open Source ECG Analysis Software Documentation. Verkkodokumentti. EP Limited. <http://eplimited.com/osea13.pdf> Luettu 14.4.2013.

54 Pan J., Tompkins W.J. 1985. A Real-Time QRS Detection Algorithm. Verkkodokumentti. <http://mirel.xmu.edu.cn/mirel/public/Teaching/QRSdetection.pdf> Maaliskuu 1985. Luettu 15.4.2013.

55 nRFready iOS Demo App. 2013. Verkkodokumentti. Nordic Semiconductor. <http://www.nordicsemi.com/eng/Products/ANT/nRFready-iOS-Demo-App> Luettu 15.4.2013.

56 Leposyke. 2013. Verkkodokumentti. Mediweb Oy. <http://www.hoitonetti.fi/terveysarvot/leposyke/> Luettu 3.5.2013.