SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

ANDROID-SOVELLUS SPIROMETRIALAITTEELLE

TEKIJÄ: Juha-Matti Kettunen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Sähkötekniikan koulutusohjelma	
Työn tekijä(t) Juha-Matti Kettunen	
Työn nimi Android-sovellus spirometrialaitteelle	
Päiväys 17.8.2017	Sivumäärä/Liitteet 35/3
Ohjaaja(t) yliopettaja Väinö Maksimainen	
Toimeksiantaja/Yhteistyökumppani(t) Medikro Oy	

Tiivistelmä

Opinnäytetyö tehtiin Medikro Oy:lle. Työn tavoitteena oli sekä suunnitella että kehittää Android-sovellus spirometria laitteelle. Työssä tutkittiin myös uusimpien Bluetooth-standardien erot sekä selvitettiin uusimman Bluetooth-standardin hyödyntämismahdollisuutta jatkoa varten.

Työ suoritettiin työntekijän omalla kannettavalla tietokoneella. Android-sovelluksen kehittämisen tarkoituksena oli samalla tutkia, miten haastavaa valmiista Windows-ohjelmasta porttaaminen Android-sovellukselle on. Sovelluksen kehittämistä varten tutkittiin eri kehitysympäristöjä ja valittiin parhaiten soveltuva sovelluksen toteuttamiseen. Vaatimuksena oli, että kehitysympäristöllä piti pystyä ohjelmoimaan sovellus Androidille. Tutkittavaksi valittiin Xamarin, Android Studio ja Qt, koska näillä jokaisella pystyi ohjelmoimaan sovelluksen Androidille.

Ohjelmointikieleksi valittiin C++, koska Medikronin olemassa olevassa valmiissa Windows-ohjelmassa oli käytetty samaa ohjelmointikieltä. Käyttämällä samaa ohjelmointikieltä pystyttiin hyödyntämään valmiista ohjelmasta osia Android-sovelluksen kehittämisessä. Ohjelmointikielen valinta vaikutti kehitysympäristön valitsemiseen.

Työn tuloksena saatiin yksinkertainen ja toimiva Android-sovellus. Sovellukseen portattiin kolme funktiota, joiden avulla saatiin Bluetooth Low Energy-yhteys avattua, lähetettyä Android-puhelimesta dataa spirometrialaitteelle ja katkaistua yhteys. Tämä työ loi pohjan sovelluksen jatkokehittämistä varten, jolloin sovellusta voi parantaa porttaamalla ja kehittämällä lisää funktioita.

Avainsanat
Android, Bluetooth Low Energy, BLE, spirometria

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Electrical Engineering			
Author(s) Juha-Matti Kettunen			
Title of Thesis Android Application for Spirometer			
Date	17.8.2017	Pages/Appendices	35/3
Supervisor(s) Mr. Väinö Maksimainen, Principal Lecturer			
Client Organisation /Partners Medikro Oy			

Abstract

The thesis was made for a company called Medikro Oy. The goal of the thesis was to plan and develop an Android application for a spirometry device. The thesis also included a research of the latest Bluetooth standards and how the company will benefit from it.

The thesis was carried out on the author's own laptop. The purpose of the Android application development was also to explore how challenging it is to port from the completed Windows application to the Android application. Different integrated development environments were examined for the application development. The choice of the most suitable one for the development was also made. The requirement for the integrated development environment was a possibility to develop the Android application. Xamarin, Android Studio and Qt were chosen for the examination because each of these had the possibility to develop the application for Android.

C++ was chosen as the programming language because it was already used at the Medikro's completed Windows application. By using the same programming language, it was possible to utilize parts of the Windows application in the Android application development. The choice of the programming language also effected on the choice of the integrated development environment.

The result of this thesis was a simple working Android application. Three functions were ported to the Android application. Bluetooth Low Energy connection could be opened and closed and data could be transferred between the Android phone and the spirometry device with these functions. This thesis created a foundation for further development of the Android application.

Keywords
Android, Bluetooth Low Energy, BLE, spirometer

SISÄLTÖ

LYHENTEET JA MÄÄRITELMÄT 6
1 JOHDANTO 7
2 SPIROMETRIA 8
2.1 Dynaaminen spirometria 8
2.2 Virtaus-tilavuusspirometria 9
2.3 PEF-mittaus 9
3 BLUETOOTH 10
3.1 Bluetooth SIG 10
3.2 Bluetooth Low Energy 11
3.3 Bluetooth 5 12
4 ANDROID 13
4.1 Android-käyttöjärjestelmä 13
4.2 Android:n arkkitehtuuri 14
4.2.1 Linux Kernel 14
4.2.2 Laitteiston abstraktiotaso (HAL) 14
4.2.3 Android Runtime 15
4.2.4 Natiivit C / C++ -kirjastot 15
4.2.5 Java API-kehys 15
4.2.6 Järjestelmäsovellukset 15
5 KEHITYSYMPÄRISTÖN VALINTA 16
5.1 Xamarin 16
5.1.1 Objective-C:n, Java:n, C ja C++:n yhteensopivuus 16
5.1.2 Modernisten ohjelmointikielten rakenteet 16
5.1.3 BCL 16
5.1.4 Moderni kehitysympäristö 17
5.1.5 Alustariippumattomuuden tuki 17
5.1.6 Miten Xamarin toimii? 17
5.2 Android Studio 17
5.2.1 Gradle Build-järjestelmä 18
5.3 Qt 18
5.3.1 The Qt Company 18

5.3.2 Qt Creator 19
5.3.3 Build-järjestelmä 19
5.3.4 Käyttöliittymät 19
5.4 Valittu kehitysympäristö 19
6 ANDROID-SOVELLUKSEN KEHITTÄMINEN 20
6.1 Yleistä 20
6.2 Qt Creator:lla Android-sovelluksen kehittämiseen tarvittavat työkalut 21
6.2.1 Android SDK 21
6.2.2 JDK 21
6.2.3 Android NDK 21
6.2.4 Apache Ant 22
6.3 Android-sovellus 22
6.3.1 Android-käyttöliittymä 22
6.3.2 Signals and Slots 23
6.3.3 Jaettu kirjasto 24
6.4 Ajuri 23
6.4.1 QThread-luokka 25
6.5 Bluetooth Low Energy:n rajapinnan ohjelmoiminen Qt:lla 26
6.5.1 Yhteyden muodostaminen 26
6.5.2 Palvelun etsiminen 27
6.5.3 Oheislaitteen kanssa keskusteleminen 29
7 KEHITYSIDEAT 30
LÄHTEET JA TUOTETUT AINEISTOT ..... **VIRHE. KIRJANMERKKIÄ EI OLE MÄÄRITETTY.**
LIITE 1: QT:N ASENNUS 33

LYHENTEET JA MÄÄRITELMÄT

ADT = Android – Development – Tools

AOT = Ahead – of –Time

API = Application – Programming – Interface

APK = Android – Application – Package

ART = Android – Runtime

ATT = Attribute – Protocol

BCL = Base – Class – Library

FEC_1 = Uloshengityksen sekuntikapasiteetti

FIV_1 = Sisäänhengityksen sekuntikapasiteetti

FVC = Nopea vitaalikapasiteetti

GATT = Generic – Attribute – Profile

IL = Intermediate – Language

IoT = Internet – of – Things

IO = Input – Output

JDK = Java SE Development Kit

JIT = Just – In – Time

JRE = Java – Runtime – Environment

MMEF = Ulospuhalluksen keskivaiheen virtaus

MOC = Meta-Object Compiler

NDK = Native – Development – Kit

PEF = Uloshengityksen huippuvirtaus

PIF = Sisäänhengityksen huippuvirtaus

SDK = Software – Development – Kit

SIG = Special – Interest – Group

UUID = Universally – Unique – Identifier

1 JOHDANTO

Opinnäytetyö suoritettiin Medikro Oy-yritykselle, joka valmistaa spirometreja ja muita oheislaitteita hengityksen mittaamiseen. Työn aiheena oli suunnitella ja kehittää Android-sovellus, joka kommunikoi spirometrialaitteen kanssa Bluetooth Low Energyllä. Bluetooth Low Energy valittiin langattomaksi kommunkointiprotokollaksi, koska se oli jo käytössä spirometrialaitteessa. Bluetooth Low Energyn avulla spirometrialaite käyttää vain vähän virtaa datan lähettämiseen, mikä pidentää sen toiminta-aikaa.

Työn ensimmäisessä vaiheessa tutkittiin eri kehitysympäristöjä, joilla voisi kehittää sovelluksen. Vaatimuksena oli, että kehitysympärisöllä täytyy pystyä kehittämään Android-sovellus ja kehittämiseen käytetään C++ -ohjelmointikieltä. Tämä oli tärkeää työn laajuuden kannalta, koska sovelluksen kehittämiseen käytettiin hyödyksi valmista ohjelmaa, joka oli kirjoitettu C++ -ohjelmointikielellä. Eri ohjelmointikieltä käyttämällä jouduttaisiin porttaamaan koko ohjelman toiselle ohjelmointikielelle, joka lisäisi työn määrää merkittävästi. Tutkittavaksi valittiin Xamarin-, Android Studio- ja Qt Creator-kehitysympäristöt. Näillä jokaisella pystyi ohjelmoimaan Android-sovelluksia sekä käyttämään C++ -ohjelmointikielenä.

Android-sovelluksen kehittämisen jälkeen tutkittiin uusimpien Bluetooth-standardien eroja sekä miten uusia Bluetooth-standardeja voisi jatkossa hyödyntää. Tämän tutkimuksen avulla yritys sai lisätietoa uusimmasta standardista ja sen hyödyistä. Tämä tuo myös lisää tietoa uudemman standardin käyttöönoton kannattavuudelle. Bluetooth-standardin tietoperustana käytettiin Bluetooth SIG-verkkosivua ja muita siihen liittyviä verkkosivustoja.

Työn jälkeen yrityksellä on tieto Android-sovelluksen kehittämisestä C++ -ohjelmointikielellä sekä sen haastavuudesta. Lisäksi yritykselle jää valmis sovelluspohja jatkokehitystä varten.

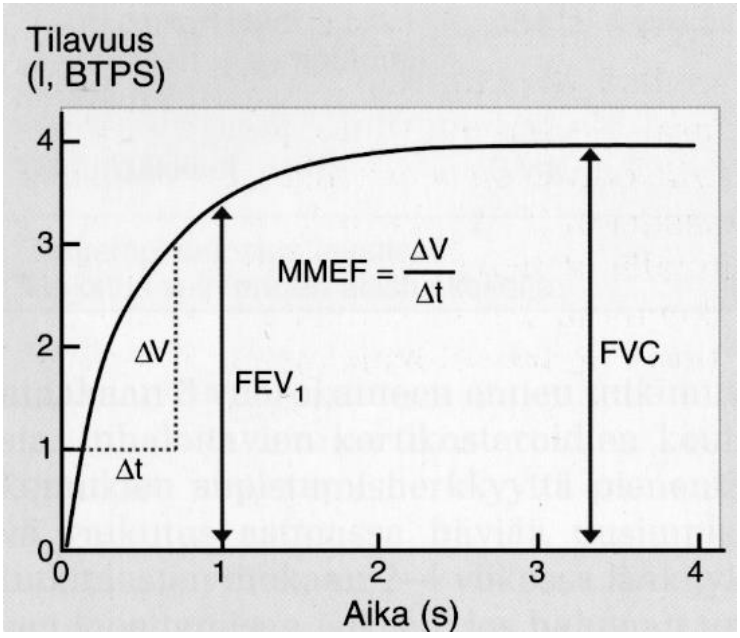
2 SPIROMETRIA

Spirometrialla mitataan keuhkoista tulevan ja menevän ilman tilavuutta ja virtausta. Mittaustuloksista voidaan päätellä mahdolliset poikkeavuudet keuhkotilaavuudessa tai virtauksissa. Tuloksista arvioidaan myös mahdollisen astmalääkityksen teho, työkyvykkyys sekä selvitetään toimenpide- ja leikkauskelpoisuus. Tuloksilla myös diagnosoidaan ja seurataan obstruktiivisia ja restriktiivisia keuhkosairauksia ja sentraalisia hengitystieahtaumia. Obstruktio tarkoittaa hengitystienahtautta ja restriktio keuhkojen tilavuuden pienenemistä. Poikkeavuuden palautuvuutta voidaan mitata keuhkoputkien avautumis- eli bronkkodilataatiokokeessa (Piirilä, 2013.)

2.1 Dynaaminen spirometria

Dynaamisessa spirometriassa uloshengitystilavuus esitetään ajan funktiona. Dynaamiseen spirometria mittaukseen on olemassa paljon erilaisia spirometreja, joista yleisin on paljespirometri. Tällä laitteella voi tutkia vain uloshengityksen dynamiikkaa ilman lisälaitteita. Sisähengityksen tutkimiseen käytetään muun muassa pneumotakografispirometria. Virtaus-tilavuusspirometrit ovat nykyään syrjäyttäneet dynaamiset spirometrit, joita käytetään pääosin mittaustulosten laadun arviointiin. (Sovijärvi;ym., 2003.)

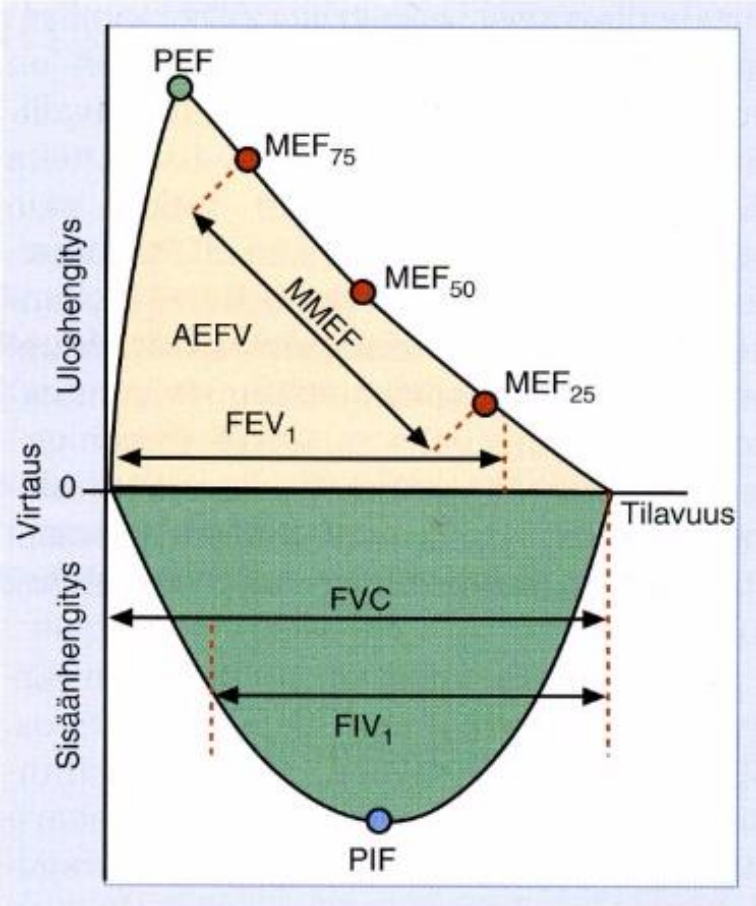
Dynaamista spirometriatutkimuksessa puhaltava henkilö istuu tukevalla tuolilla, nenä suljettuna puhalluksen ajaksi nenäpuristimella ja suukappale tiiviisti suussa. Tämän jälkeen puhaltava henkilö vetää keuhkonsa täyteen ilmaa ja puhaltaa keuhkonsa tyhjäksi mahdollisimman nopeasti. Tulosten luotettavuuden vuoksi tämä on toistettava niin pitkään, kunnes saadaan kolme yhteneväistä mittaustulosta. Onnistuneiden puhallusten jälkeen mitataan FEV_1 ja FVC. Suurimman ja toiseksi suurimman välinen mittaustulos saa poiketa korkeintaan 4% toisistaan, jotta tulos olisi luotettavalla tasolla. (Sovijärvi;ym., 2003.)



KUVA 1. Uloshengitystilavuuden esittäminen ajan funktiona (Sovijärvi;ym., 2003.)

2.2 Virtaus-tilavuusspirometria

Virtaus-tilavuusspirometria on korvannut dynaamisen spirometrian. Virtaus-tilavuusspirometria suoritetaan spirometreilla, jotka ovat dynamiikaltaan herkkiä ja sisältävät virtausanturin tai elektroniikka- ja tietojenkäsittelyjärjestelmän virtaus- ja tilavuusmittausta varten. Erona dynaamiseen spirometriaan virtaus-tilavuusspirometriassa voidaan mitata myös sisäänhengitystä ja mittaustulokset esitetään virtaus-tilavuuskoordinaatistossa. (Sovijärvi;ym., 2003.)



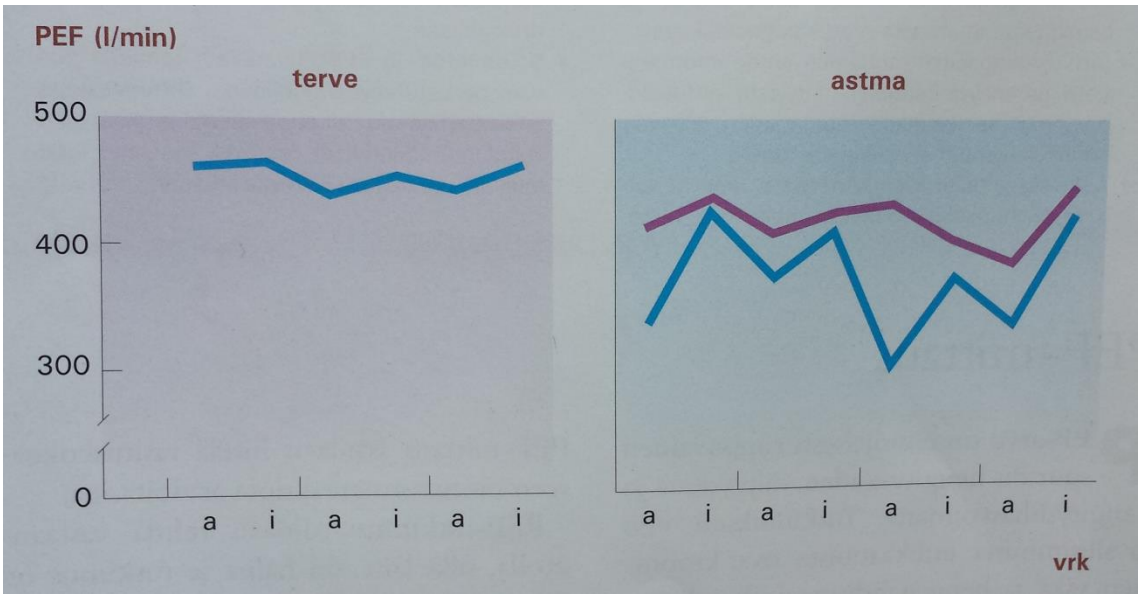
KUVA 2. Virtaus-tilavuuskoordinaatisto (Sovijärvi;ym., 2003.)

Virtaus-tilavuusspirometrian mittauksessa soveltuvat saman säännöt kuin dynaamisessa spirometria mittauksessa. Puhaltava henkilö vetää keuhkot täyteen ilmaa ja puhaltaa keuhkot tyhjiksi niin kovaa kuin jaksaa. Onnistuneet puhallukset esitetään näyttöpäätteellä tai piirturilla, jossa puhalluksen mittaustulokset on esitetty päällekkäin. Suurimmista virta-arvoista muodostetaan verhokäyrä, josta tehdään päätelmät. Jos on tarvetta mitata sisäänhengitystä, mitataan se erikseen. (Sovijärvi;ym., 2003.)

2.3 PEF-mittaus

PEF on uloshengityksen huippuvirtamittaus, jossa mitataan vähintään 10 millisekuntia kestävä virtauspiikki. PEF-mittaukseen riittää lyhyt ja voimakas puhallus, koska huippuvirtaus saadaan aina puhalluksen alkuvaiheessa. PEF-arvoihin vaikuttaa ensisijaisesti suurten hengitysteiden väljyys ja hengityslihasvoima. PEF-mittausta tehdään, jos henkilöllä esiintyy kroonista yskää ja hengenahdistusta,

joista voidaan epäillä astmaa tai keuhkoahtaumatautia. Mittausta suoritetaan myös astmapotilaan ja lääkehoidon vaikutuksen seuraamisessa. Mittauksen voi tehdä vastaanotolla, mutta sen merkitys korostuu vasta potilaan seurannassa vertailemalla aamun ja illan mittaustuloksia sekä vertailemalla ärsykkeen ja lääkkeen vaikutusta. (Kinnula;ym., 2005.)



KUVA 3. Terveen ja astmaatikon PEF-seuranta. ”a” tarkoittaa aamua ja ”i” iltaa (Kinnula;ym., 2005.)

Kuvassa 3 nähdään terveen ja astmaatikon PEF-mittausten erot. Sinisellä viivalla on merkitty ennen beetasympatomimeetti-inhalaatiota ja punaisella 15 minuuttia beetasympatomimeetti-inhalaation jälkeen. Kuvasta huomataan inhalaation toimivuus ja vuorokausivaihtelun korostuminen astmaatikolla. Pienet PEF-arvot aamuisin on tyypillistä astmaatikolla. (Kinnula;ym., 2005.)

3 BLUETOOTH

Bluetooth on langattoman tiedonsiirtoteknologian standardi lähietäisyydellä kommunikoimiseen. Bluetooth-laite käyttää johtojen ja kaapeleiden sijaan radioaaltoja esimerkiksi puhelimen tai tietokoneen yhdistämiseen. Bluetooth-laite kuten kuulokkeet tai rannekello sisältävät pienen mikrosirun, jossa on Bluetooth-radiolähetin ja yhdistämiseen tarvittava ohjelma. Kun kaksi Bluetooth laitetta haluavat keskustella keskenään, niiden täytyy muodostaa pari. Kommunikointi Bluetooth-laitteiden välillä tapahtuu lyhyen kantaman, piconet-tietoverkon kautta. Piconet on laitteiden tietoverkko, joka käyttää Bluetooth-teknologiaa. Kun tietoverkko on muodostettu, yksi laite ottaa master eli isäntä roolin ja loput laitteista toimivat slavena eli orjina. Piconet muodostuu dynaamisesti ja automaattisesti samalla, kun Bluetooth-laitteet liittyvät ja poistuvat radiokantaman alueelta. Suosituimpia Bluetooth-sovelluksia on muun muassa langaton äänentoisto. (Bluetooth SIG, 2017.)

3.1 Bluetooth SIG

Bluetooth SIG on jäsenorganisaatioista muodostunut verkosto, joka omistaa, valvoo ja suojelee Bluetooth-tavaramerkkiä. Se myös samalla jatkuvasti kehittää ja päivittää Bluetooth standardia kehittyvää teknologiaa ja markkinoita varten. SIG-järjestö perustettiin vuonna 1998 ja siihen kuuluu yli

30 000 jäsenyritystä. Bluetooth SIG itsessään ei valmista tai myy Bluetooth-laitteita. SIGin pääkonttori sijaitsee Kirkland-kaupungissa Washingtonin osavaltiossa. (Bluetooth SIG, 2017.)

Voidakseen käyttää Bluetooth-teknologiaa tuotteissaan yrityksen on oltava Bluetooth SIGin jäsen. Liittyminen on tehty yrityksille helpoksi, sillä Bluetooth SIG tarjoaa kahden tasoista jäsenyyttä, joista toinen on maksuton. Ilmaista jäsenyyttä kutsutaan Adopter-jäsenyydeksi. Tällä jäsenyydellä yritys saa lisenssin tuottaa Bluetooth-teknologiaa käyttäviä tuotteita ja käyttää Bluetooth-tuotemerkkiä pätevissä tuotteissa Bluetooth-tavaramerkkilisenssisopimuksen mukaisesti. Jäsen saa käyttöönsä työkaluja kuten Profile Tuning Suite ja voi verkostoitua ja tehdä yhteistyötä muiden Bluetooth SIG-jäsenien kanssa. Maksullista jäsenyyttä kutsutaan Associate-jäsenyydeksi. Associate-jäsenyydellä yritys voi osallistua Bluetooth SIG-työryhmiin ja Bluetooth standardin kehittämiseen. Yritys pääse tämän lisäksi myös käsiksi muun muassa markkinoiden tiedusteluraportteihin. Jotta yritys voi hakea Associate-jäsenyyttä, täytyy sillä olla Adopter-jäsenyys. (Bluetooth SIG, 2017.)

3.2 Bluetooth Low Energy

Bluetooth Low Energy on osa Bluetooth SIGin julkaisemaa Bluetooth Core Specification version 4.0-standardia, mikä virallisesti julkaistiin vuonna 2011. Se käyttää samaa 2,4 GHz-taajuusaluetta kuten Classic Bluetooth. Suurin ero Classic Bluetoothiin on nimestä pääteltävissä matalavirrankulutus. Se mahdollistaa Bluetooth Low Energy-laitteiden toiminnan kuukausiksi tai jopa vuosiksi pelkällä nappiparistolla. Laitteita, jotka tukevat tätä ominaisuutta, kutsutaan Bluetooth Smart Ready-laitteiksi. Tämän teknologia avainominaisuudet ovat kolme eri virrankulutus tasoa, kyky toimia jopa vuosia nappiparistoilla, edullinen hinta, monen toimittajan kanssa yhteentoimiva ja parannettu kantama. (The Qt Company, 2017.)

Bluetooth Low Energy käyttää asiakas-palvelinarkkitehtuuria. Palvelin tarjoaa palveluita kuten lämpötilaa tai sykettä ja mainostaa niitä. Asiakas yhdistää palvelimeen ja lukee palvelimen mainostamat tiedot. Esimerkkinä asunto, jossa on Bluetooth Smart Ready-antureita kuten termostaatti. Termostaatti on slave eli orja, joka mainostaa asunnon lämpötila-arvoja. Termostaattiin voidaan yhdistää älypuhelimella tai tietokoneella, jotka lukevat tiedot ja esittävät ne käyttäjälle. (The Qt Company, 2017.)

Bluetooth Low Energy perustuu kahteen protokollaan: ATT ja GATT. ATT:n perusrakenne on attribuutti. Jokainen attribuutti koostuu kolmesta elementistä, jotka ovat arvo, UUID ja 16-bittinen kahva. Arvo sisältää halutun tiedon, UUID kertoo attribuutin tyypin ja kahva on attribuutin ainutlaatuinen tunniste. Palvelin varastoi attribuutit ja asiakas käyttää ATT-protokollaa palvelimen lukemiseen ja palvelimelle kirjoittamiseen. (The Qt Company, 2017.)

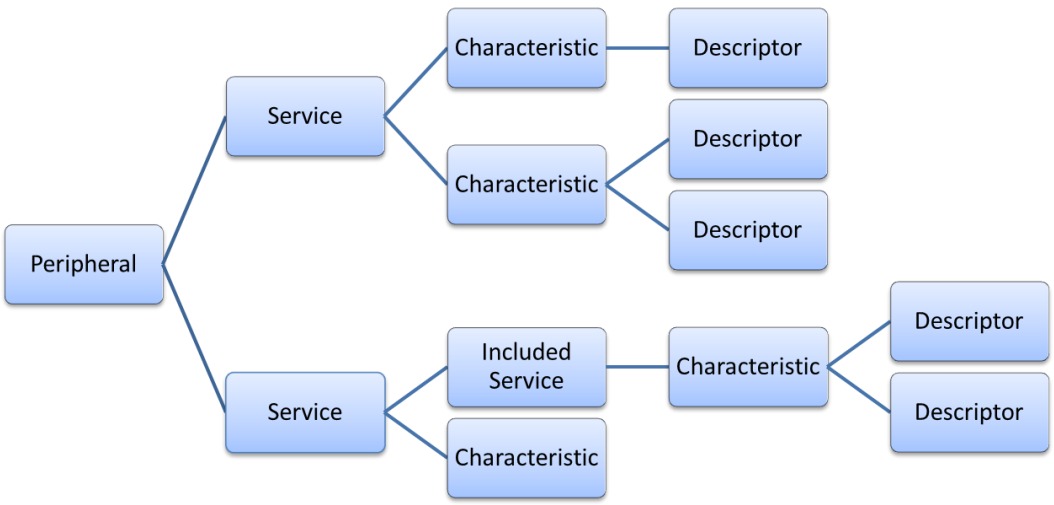
GATT määrittelee attribuuttien ryhmittäytymisen käyttämällä esimääriteltyjen UUID:n merkitystä. Taulukko 1 näyttää esimerkin sykkeen palvelusta. Todelliset arvot on varastoitu kahden karakterin sisälle. (The Qt Company, 2017.)

TAULUKKO 1. Esimerkki sykkeen palvelusta

Handle	UUID	Value	Description
0x0001	0x2800	UUID 0x180D	Begin Heart Rate service
0x0002	0x2803	UUID 0x2A37, Value handle: 0x0003	Characteristic of type *Heart Rate Measurement (HRM)*
0x0003	0x2A37	65 bpm	Heart rate value
0x0004	0x2803	UUID 0x2A08, Value handle: 0x0006	Characteristic of type Date Time
0x0005	0x2A08	18/08/2014 11:00	Date and Time of the measurement
0x0006	0x2800	UUID xxxxxx	Begin next service
...	...	...	...

Taulukossa 1 GATT määrittelee palvelun aloituksen paikasta UUID 0x2800. Jokainen attribuutti 0x2800 jälkeen on osa palvelua seuraavaan 0x2800 paikkaa, jos loppua ei tule ennen sitä. Yleisesti tunnettu UUID 0x2803 karakteri näyttää, että karakterit voidaan löytää ja jokaisella karakterin arvolla on tyyppi, joka määrittää sen luonteen. Esimerkkinä taulukossa 1 UUID:t 0x2A08, joka esittää arvot päivämäärinä ja 0x2A37, joka esittää arvot lyönteinä per minuutti. Jokaisen taulukossa 1 olevan karakterin UUID on määritellyt Bluetooth SIG. (The Qt Company, 2017.)

Yleisesti jokainen palvelu sisältää yhden tai useamman karakterin. Karakteri sisältää dataa ja se voidaan esittää deskriptionien avulla. Tämän avulla saadaan lisää tietoa tai keinoja karakterin käsittelemiseen. Kaikki palvelut, karakterit ja deskriptionit tunnistetaan niiden 128-bittisellä UUID:llä. On myös mahdollista sisällyttää palveluita palvelun sisälle. (The Qt Company, 2017.)



KUVA 4. Slaven eli orjan kaavio (The Qt Company, 2017.)

3.3 Bluetooth 5

Bluetooth 5 on uusin Bluetooth-teknologian standardi. Se kehitettiin globaaliseksi, langattomaksi standardiksi vastaamaan teollisuuden tarpeita yksinkertaisuudellaan ja turvallisuudellaan. Bluetooth

5:ssa on neljä kertaa pidempi kantama, kaksi kertaa nopeampi tiedonsiirtonopeus ja kahdeksan kertaa suurempi mainostettavan viestin kapasiteetti verrattuna edelliseen Bluetooth standardiin. Nämä parannukset on luotu lisäämään Bluetoothin toiminnallisuutta IoT:ä varten. (Bluetooth SIG, 2017.)

Bluetooth on vaikuttanut ihmisten käsitykseen IoT:stä. Bluetooth 5 tarjoaa luotettavia IoT-yhteyksiä ja helpottaa beaconien käyttöönottoa. Näiden takia yhteyksien esteet vähenevät ja mahdollistaa paremman IoT-toimivuuden. Bluetooth 5 tarjoaa joustavuutta IoT-ratkaisujen rakentamiseen, jotka perustuvat teollisuuden tarpeisiin. Nämä ominaisuudet ovat parannettu matka, nopeus ja turvallisuus. Näillä ominaisuuksilla voidaan parantaa erilaisia ympäristöjä ja lopputuotteita. Bluetooth 5:n lisääntynyt nopeus luo perustan seuraavan sukupolven Bluetooth-äänentoistolle ja laajennettu kantama tuo luotettavia IoT-yhteyksiä. Laajennettu kantama mahdollistaa myös Bluetoothin käytön kotitalouksissa, rakentamisessa ja ulkoilmatöissä. (Bluetooth SIG, 2017.)

Bluetooth on erittäin yleisesti käytetty tekniikka. Mitään muuta langatonta tekniikkaa ei ole asennettu niin moneen laitteeseen kuin Bluetoothia, jota on asennettu yli 10 miljardiin laitteeseen. Bluetooth 5 sisältää päivityksiä, jotka vähentävät potentiaalisia häiriöitä muiden langattomien teknologien kanssa. Näillä päivityksillä varmistetaan Bluetooth-laitteiden toimivuus jatkuvasti kasvavassa, kompleksisessa ja globaalisessa IoT-ympäristössä. Bluetooth 5 säilyttää myös samalla sen matalavirtaisen toiminnallisuuden ja joustavuuden vastaamaan laite- tai sovelluskehittäjien tarpeita. (Bluetooth SIG, 2017.)

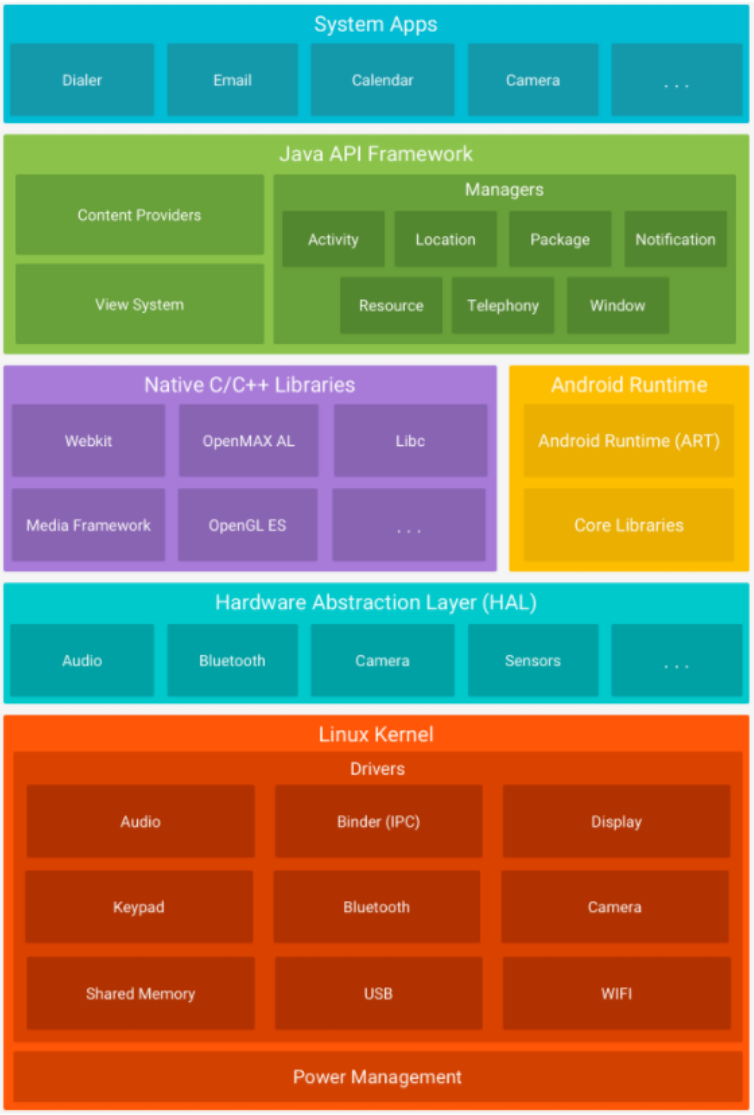
4 ANDROID

4.1 Android-käyttöjärjestelmä

Android-käyttöjärjestelmä on Linuxiin perustuva käyttöjärjestelmä, jonka kehittivät Open Handset Alliance. Androidin loivat alun perin Android Inc., jonka Google osti vuonna 2005. Google liittyi yhteen muiden yritysten kanssa muodostaakseen Open Ha ndset Alliancen. OHA vastaa Android-käyttöjärjestelmän jatkuvasta kehityksestä. (Techopedia.)

Androidin taustalla toimiva ydin perustuu Linuxiin, mutta se on räätälöity Googlen ohjeiden mukaan. Muun muassa siihen ei ole tukea GNU-kirjastoille eikä sisällä natiivia X Windows-järjestelmää. Linux kernelin sisällä on ajurit näytölle, kameralle, flash-muistille, näppäimistölle, WiFi:lle ja äänelle. Linux kerneli toimii abstraktiona laitteiston ja puhelimen ohjelmiston välillä. Se myös ylläpitää ydinjärjestelmän palveluita kuten turvallisuus, muistinhallinta, prosessinhallinta ja tietoverkko. (Techopedia.)

4.2 Androidin arkkitehtuuri



KUVA 5. Androidin arkkitehtuuri (Android Developer.)

4.2.1 Linux Kernel

Android perustuu Linux-kerneliin. Esimerkkinä ART käyttää Linux-kerneliä alatason toimintoihin kuten matalan tason muistinhallintaan. Linux-kerneliä käyttämällä Android saa käyttöönsä tärkeitä turvallisuusominaisuuksia ja sallii laitevalmistajien kehittää laitteisto-ohjelmia hyvin tunnetulle kernelille. (Android Developer.)

4.2.2 Laitteiston abstraktiotaso (HAL)

HAL mahdollistaa standardin käyttöliittymän, joka tuo esille laitteiden laitteisto-ominaisuudet korkeamman tason Java API-kehykselle. HAL koostuu monista kirjastomoduuleista, joista jokainen toteuttaa liitännän tietylle laitteistokomponentille kuten kameralle tai Bluetooth-moduulille. Kun API-kehys pyytää pääsyä laitteistoon, Android-järjestelmä lataa pyydetyn laitteistokomponentin kirjastomoduulin. (Android Developer.)

4.2.3 Android Runtime

Laitteiden, joissa on Android 5.0-versio tai uudempi, jokaisella sovelluksella on oma prosessi ja ART-instanssi. ART on suunniteltu ajamaan useita virtuaalikoneita vähän muistia omaaville laitteille suorittamalla DEX-tiedostoja. DEX on tavukoodi formaatti, joka on erityisesti suunniteltu Androidille, joka on optimoitu minimaaliseen muistin käyttöön. Build toolchainit, kuten Jack, kääntää Javan lähteet DEX-tavukoodiksi, jota voidaan ajaa Android-alustalla. (Android Developer.)

ART:n merkittävimmät ominaisuudet ovat:

- AOT ja JIT- kokoelma
- Optimoitu garbage collection
- Hyvä debugger-tuki

4.2.4 Natiivit C / C++ -kirjastot

Monet Android-järjestelmän komponentit ja palvelut, kuten ART ja HAL, on kehitetty natiivikoodista, joka vaatii C / C++ -ohjelmointikielellä koodattuja natiivikirjastoja. Android-alusta mahdollistaa sen, että Java-kehysten ohjelmointirajapinnat tuovat esille natiivikirjastojen toiminnallisuuksia sovellusten käytettäviksi. Esimerkkinä OpenGL ES:ään päästään käsiksi Android-kehyksen Java OpenGL API:n avulla, jonka avulla voidaan lisätä sovellukseen 2D- ja 3D-grafiikoiden tuen. Jos sovellus käyttää jo valmiiksi C- tai C++ -kieltä, käyttämällä Android NDK:ta päästään käsiksi natiivikirjastoihin suoraan natiivi koodista. (Android Developer.)

4.2.5 Java API-kehys

Kaikki Android-käyttöjärjestelmän ominaisuudet ovat käytettävissä API:en kautta, jotka ovat kehitetty Java-ohjelmointikielellä. API:t ovat ohjelman rakennepalasia, joita tarvitaan Android-sovellusten luomiseen. Nämä palaset yksinkertaistavat ydinosien, moduulijärjestelmän komponenttien ja palveluiden uudelleenkäyttöä. Sovellusten kehittäjillä on pääsy samoihin API-kehyksiin kuin mitä Android-järjestelmän sovellukset käyttävät. (Android Developer.)

4.2.6 Järjestelmäsovellukset

Androidissa on sähköpostia, SMS-viestittelyä, kalentereita, internetin selaamista, yhteistietoja ja muita sovelluksia varten joukko ydinohjelmia. Alustan mukana tulleilla sovelluksilla ei ole erityisasemaa verrattuna sovelluksiin, joita käyttäjä asentaa. Tämän avulla kolmannen osapuolen sovellusta voidaan käyttää muun muassa oletusselaimena, viestivälineenä tai jopa oletusnäppäimenä. Järjestelmäsovellukset toimivat sekä käyttäjien sovelluksina että avainominaisuuksina kehittäjien omille sovelluksille. Esimerkkinä sovellus, joka tarvitsee lähettää SMS-viesti. Kehittäjän ei tarvitse erikseen kehittää sovellusta viestin lähettämiseen vaan kutsua valmiiksi asennettua SMS-sovellusta viestin lähettämiseen. (Android Developer.)

5 KEHITYSYMPÄRISTÖN VALINTA

5.1 Xamarin

Xamarin on kehitysympäristö, joka tarjoaa yhden ohjelmointikielen C#:n, luokkakirjaston ja runtime-ominaisuuden, jotka toimivat kaikilla kolmella iOS-, Android- ja Windows Phone –alustalla kääntäen samalla natiivisovelluksiksi. Windows Phonen natiivikieli on jo valmiiksi C#. Jokaisella näillä alustoilla on erilaiset ominaisuudet ja jokaisen tapa kirjoittaa natiivisovelluksia vaihtelee. Toisin sanoen käännetään sovellukset natiivikoodiksi, jolloin ne toimivat moitteettomasti aliverkon Java-osajärjestelmän kanssa. Esimerkiksi jotkin alustavat sallivat vain HTML- ja JavaScript-sovellusten kehittämisen, kun taas toiset sallivat vain C / C++ -koodilla kirjoittamisen. (Xamarin, 2017.)

Xamarinin ainutlaatuisuus on se, että se yhdistää natiivialustojen hyvät puolet ja lisää omia ominaisuuksiaan kuten:

1. SDK:ten sitominen.
2. Objective-C:n, Javan, C ja C++:n yhteensopivuus
3. Modernisten ohjelmointikielten rakenteet
4. BCL
5. Moderni kehitysympäristö
6. Tuki alustariippumattomuudelle

5.1.1 Objective-C:n, Javan, C ja C++:n yhteensopivuus

Xamarin tarjoaa mahdollisuuden kutsua Objective-C-, Java-, C- ja C++ -kirjastoja, joiden avulla voidaan käyttää laajaa ja valmiiksi kehitettyä kolmannen osapuolen koodia. Tämän avulla voidaan hyödyntää olemassa olevia iOS- ja Android-kirjastoja, jotka ovat kirjoitettu Objective-C:llä, Javalla, C- tai C++:lla. Lisäksi Xamarin tarjoaa projektien sitomisen, jonka avulla voidaan helposti sitoa natiivi Objective-C- tai Java-kirjastoja käyttämällä deklaratiivista syntaksia. (Xamarin, 2017.)

5.1.2 Modernisten ohjelmointikielten rakenteet

Xamarin-sovellukset koodataan modernilla C#-ohjelmointikielellä, joka sisältää merkittäviä parannuksia verrattuna Objective-C:hen ja Javaan kuten dynaamisten kielten ominaisuudet, funktionaalisia rakenteita kuten Lambdas, LINQ, rinnakkaisohjelmointiominaisuudet ja paljon muuta. (Xamarin, 2017.)

5.1.3 BCL

Xamarin-sovellukset käyttävät .NET BCL:tä, joka on massiivinen luokkakokoelma. BCL:llä on kattavia ja virtaviivaisia ominaisuuksia kuten XML, tietokanta, serialisointi, IO, merkkijono ja verkkotuki. Nämä ovat vain osa BCL:n ominaisuuksista. Lisäksi olemassa oleva C#-koodi voidaan kääntää käytettäväksi sovelluksissa, jotka tarjoavat pääsyn tuhansille kirjastoille, joiden avulla saa vielä lisää ominaisuuksia, joita ei BCL:ssä ole. (Xamarin, 2017.)

5.1.4 Moderni kehitysympäristö

Xamarinia käytetään Xamarin Studiolla Mac OS X-käyttöjärjestelmässä ja Visual Studiolla Windows-käyttöjärjestelmässä. Nämä molemmat ovat moderneja kehitysympäristöjä, jotka sisältävät ominaisuuksia kuten koodin ennakoivan syötön, projektin ja ratkaisun hallintajärjestelmän, kattavan projektimalli-kirjaston ja monia muita. (Xamarin, 2017.)

5.1.5 Alustariippumattomuuden tuki

Xamarin tarjoaa kehittyneen, alustariippumattoman tuen kolmelle suurimmalle mobiilialustalle iOS:lle, Androidille ja Windows Phonelle. Sovellukset voidaan kirjoittaa siten, että se jakaa jopa 90% koodistaan. Xamarin.Mobile-kirjasto tarjoaa yhtenäisen ohjelmointirajapinnan yhteisten resurssien käyttämiseksi kaikilla kolmella alustalla. Tämä vähentää merkittävästi mobiilikehittäjien, joiden kohteena on kolme suosituinta mobiilialustaa, kehityskustannuksia ja markkinoiden aikaa. (Xamarin, 2017.)

5.1.6 Miten Xamarin toimii?

Xamarin tarjoaa kaksi kaupallista tuotetta, jotka ovat Xamarin.iOS ja Xamarin.Android. Molemmat on kehitetty Monon päälle, joka on .NET Frameworkin avoimen lähdekoodin versio ja perustuu julkaistuihin .NET ECMA-standardeihin. Mono on ollut lähes yhtä kauan olemassa kuin .NET Framework ja toimii lähes kaikilla alustoilla kuten Linux, Unis, FreeBSD ja Mac OS X. iOS:lla Xamarinin AOT-kääntäjä kääntää Xamarin.iOS-sovellukset suoraan natiiviksi ARM assembly-koodiksi. Androidilla Xamarinin kääntäjä kääntää IL:ksi, josta sitten JIT-käännetään natiiviksi assemblyksi, kun sovellus käynnistetään. Molemmissa tapauksissa Xamarin-sovellukset käyttävät runtimea, joka automaattisesti hoitaa tehtävät kuten muistinvarauksen, alla olevan alustan yhteentoimivuuden ja niin edelleen. (Xamarin, 2017.)

5.2 Android Studio

Android Studio on virallinen, IntelliJ IDEAan perustuva kehitysympäristö Android-sovellusten kehittämiseen. IntellJin tehokkaan koodieditorin ja kehitystyökalujen lisäksi Android Studio tarjoaa lisää ominaisuuksia, jotka parantavat tehokkuutta Android-sovelluksen kehittämiseen kuten:

- Joustava Gradleen perustuva kehitysjärjestelmä
- Nopea ja monipuolinen emulaattori
- Yhtenäinen ympäristö, jossa voit kehittää mille tahansa Android-laitteelle
- Instant Run-toiminto, jolla saat muutokset ajettua käynnissä olevaan sovellukseen luomatta uutta APK:ta
- Koodimallit ja GitHub-integraatio auttamaan yleisten sovellusominaisuuksien luomisessa ja esimerkkikoodien tuomisessa
- Laaja testaustyökalu ja kehys- valikoima

- Lint-työkalut suorituskyvyn, käytettävyyden, versioiden yhteensopivuuden ja muiden ongelmien käsittelemiseksi
- C++ ja NDK- tuki
- Sisäänrakennettu tuki Google Cloud-alustalle helpottamaan Google Cloud Messaging ja App Enginen integroimista (Android Developer, 2017.)

5.2.1 Gradle Build-järjestelmä

Android Studio käyttää Gradle Build-järjestelmää perustana. Tämä Build-järjestelmä toimii integroituna työkaluna Android Studiossa ja komentoriviltä riippumatta. Build-järjestelmän ominaisuuksia on seuraavat:

- Mukauttaminen, konfiguroiminen ja build-prosessien laajentaminen
- Useiden APK-tiedostoja luominen eri ominaisuuksilla, mutta käyttäen samaa projektia ja moduuleita
- Koodin ja eri lähteiden resurssien uudelleen käyttäminen (Android Developer, 2017.)

Käyttämällä joustavaa Gradlea voidaan saavuttaa ylläolevan listan kaikki hyödyt ilman sovelluksen ydinlähdetiedostojen muokkaamista. Android Studion Build-tiedostot on nimetty build.gradle. Ne ovat tekstitiedostoja, jotka käyttävät Groovy-syntaksia Buildn konfiguroimiseen elementeillä. Nämä elementit tarjoavat Android-laajennuksen Gradlelle. Jokaisella projektilla on ylätason Build-tiedosto koko projektille ja erillisiä moduulitason Build-tiedostoja jokaiselle moduulille. (Android Developer, 2017.)

5.3 Qt

Qt on alustariippumaton sovelluskehityskehys tietokoneille, sulautetuille järjestelmille ja älylaitteille. Se tukee muun muassa Linuxia, OS X:ä, Windowsia, VxWorksia, QNX:ä, Androidia, iOS:a, BlackBerryä, Sailfish OS:ä.

Qt itsessään ei ole ohjelmointikieli vaan C++:lla kirjoitettu kehys. MOC-esikääntäjää käytetään laajentamaan C++ -ohjelmointikieltä lisäominaisuuksilla kuten Signals ja Slots. Ennen kääntämistä MOC jäsentelee lähdetiedostot, jotka on koodattu Qt-laajennetulla C++:lla ja luo niistä standardin mukaiset C++ -lähteet. Lisäksi kehys itse ja sitä käyttävät sovellukset ja kirjastot voidaan kääntää millä tahansa standardin mukaisella C++ -kääntäjällä kuten Clang, GCC, ICC, MingGW ja MSVC. (The Qt Company, 2017.)

5.3.1 The Qt Company

Qt:n kehityksen aloittivat norjalaiset ohjelmoijat Eirik Chambe-Eng ja Haavard Nord vuonna 1990. Heidän Trolltech yritykselle, joka myi Qt-lisenssejä ja tukipalveluita, tapahtui useita omistajan vaihtoja vuosien varrella. Nykyään Trolltechin nimi on The Qt Company, joka on kokonaan Digia Oyj:n omistama sisaryhtiö. Vaikka The Qt Company käyttää pääosin Qt:ta, sitä kehittää myös suurempi

ryhmittymä Qt Project. Qt Project koostuu monista yrityksistä ja yksityishenkilöistä ympäri maailmaa, jotka noudattavat meritokraattista hallintomallia. (The Qt Company, 2017.)

5.3.2 Qt Creator

Qt:n omaa kehitysympäristöä kutsutaan Qt Creatoriksi. Se tarjoaa alustariippumattoman, täydellisesti integroidun kehitysympäristön sovelluskehittäjille. Sillä voi kehittää monenlaisiin tietokoneisiin, sulatettuihin järjestelmiin ja mobiililaitteille kuten Android ja iOS. Se toimii Linux-, MacOS- ja Windows-käyttöjärjestelmillä. (The Qt Company, 2017.)

Qt Creator tarjoaa älykkään ennakoivan koodin syötön, syntaksisen korostuksen, integroidun ohjetoiminnon, debuggerin ja profiilin integroinnin sekä integroinnin kaikille merkittävimmille versionhallintajärjestelmille kuten Git ja Bazaar. Qt Creatorin lisäksi kehittäjät voivat käyttää Qt:n Visual Studio-laajennusta. Muita kehitysympäristöjä voidaan myös käyttää, mutta kehitysympäristön käyttäminen ei ole välttämätöntä. (The Qt Company, 2017.)

5.3.3 Build-järjestelmä

Vaikka mikä tahansa Build-järjestelmää voidaan käyttää Qt:lla, Qt tuo oman qmake-järjestelmän. Se on alustariippumaton front-end natiivi build-järjestelmä kuten GNU Make, Visual Studio ja Xcode. Cmake on myös suosittu vaihtoehto Qt-projektien buildaamiseen. (The Qt Company, 2017.)

5.3.4 Käyttöliittymät

Qt Creator tarjoaa kaksi integroitua visuaalista editoria: Qt Quick Designer ja Qt Designer. Intuitiivisien, nykyaikaisten ja fluidisten käyttöliittymän luomiseen voidaan käyttää Qt Quickia. Perinteisen käyttöliittymän, joka on selvästi jäsennelty ja panostaa alustan ulkonäköön ja tuntumaan, voidaan käyttää integroitua Qt Designeria. (The Qt Company, 2017.)

5.4 Valittu kehitysympäristö

Sovelluksen kehittämiseen valitsin Qt Creator-kehitysympäristön. Parhaimpana puolena Qt Creatorissa oli, että sen pääohjelmointikieli oli C++, joka helpottaa huomattavasti valmiin koodin hyödyntämistä. Xamarinilla pystyy ohjelmoimaan myös C++:lla, mutta en saanut sitä toimimaan kuin C#:lla. Android Studiolla oli myös mahdollisuus lisätä C++ -kirjastoja, mutta se olisi ollut paljon työläämpää Qt Creatoriin verrattuna. Toinen syy oli työnantajan toive kehittää sovellus Qt Creatorilla, koska se oli yritykselle tuttu kehitysympäristö.

6 ANDROID-SOVELLUKSEN KEHITTÄMINEN

6.1 Yleistä

Android-sovellukset ohjelmoidaan Java-ohjelmointikielellä. Android SDK-työkalu kääntää koodin lisäksi muun datan ja lähdetiedostot APK-tiedostoksi, joka on apk-loppuliitteellä oleva arkistotiedosto. Yksi APK-tiedosto sisältää Android-sovelluksen koko sisällön, jonka avulla Android-käyttöjärjestelmää käyttävät laitteet asentavat sovelluksen. (Android Developer.)

Jokainen Android-sovellus on omassa ”suojalaatikossa”, jota suojelee Androidin turvallisuusominaisuudet:

- Android-käyttöjärjestelmä on monen käyttäjän Linux-järjestelmä, jossa jokainen sovellus on erillinen käyttäjä
- Oletuksena järjestelmä määrää jokaiselle sovellukselle oman Linux-käyttäjä tunnisteen, jonka jälkeen järjestelmä asettaa käyttöoikeudet sovelluksen kaikkiin tiedostoihin, jotta vain siihen määrätyllä tunnisteella on pääsy tiedostoihin
- Jokaisella prosessilla on oma virtuaalikoneensa, joten sovellusta voidaan ajaa erillään muista sovelluksista
- Oletuksena jokaista sovellusta ajaa oma Linux-prosessi. Android-järjestelmä käynnistää prosessin, kun sovelluksen mitä tahansa komponenttia ajetaan ja sulkee prosessin, kun sitä ei enää tarvita tai järjestelmä tarvitsee lisää muistia muita sovelluksia varten (Android Developer.)

Android-järjestelmä käyttää ”vähiten etuoikeus”-periaatetta. Oletuksena jokaisella sovelluksella on pääsy komponentteihin, joita se tarvitsee vain työskentelyynsä eikä mihinkään muuhun. Tämä luo turvallisen ympäristön, jossa sovellus ei pääse käsiksi järjestelmän osiin, joihin sillä ei ole oikeutta. On olemassa kuitenkin tapoja jakaa dataa muiden sovellusten kautta ja päästä käsiksi järjestelmän palveluihin. (Android Developer.)

Ensimmäisessä tavassa asetetaan kaksi sovellusta jakamaan saman Linux-käyttäjä tunnisteen, jolloin molemmilla on pääsy toistensa tiedostoihin. Molempia sovelluksia voidaan täten ajaa myös samalla Linux-prosessilla ja käyttää samaa virtuaalikonetta järjestelmäresurssien säästämiseksi. Sovellukset täytyy myös signeerata samalla sertifikaatiolla. Toisessa tavassa sovellus voi pyytää oikeudet päästä käsiksi laitteen dataan kuten käyttäjän kontakteihin, SMS-viesteihin, ulkoiseen muistiin eli SD-korttiin, kameraan ja Bluetoothiin. Käyttäjän täytyy myöntää nämä oikeudet. (Android Developer.)

6.2 Qt Creatorilla Android-sovelluksen kehittämiseen tarvittavat työkalut

6.2.1 Android SDK

SDK mahdollistaa sovellusten kehittämisen Android-alustalle. Android SDK sisältää esimerkki projekteja lähdekoodeineen, kehitystyökaluja, emulaattorin ja tarvittavat kirjastot Android-sovellusten kehittämiseen. (Techopedia.)

Joka kerta, kun Google julkaisee uuden Android-version, samalla julkaistaan vastaava SDK. Jotta uusimman Android-version toimintoja voitaisiin käyttää, sovellusten kehittäjät täytyy ladata ja asentaa jokaisen version SDK tietylle puhelimelle. Esimerkki kehitysalustoita, jotka tukevat SDK:ta ovat Window (XP tai myöhemmät versiot), Linux (viimeisimmät julkaisut) ja MAC OS X (10.4.9 tai myöhemmät versiot). Android SDK:n komponentit voidaan ladata erikseen. Kolmannen osapuolen lisäosat ovat myös ladattavina. (Techopedia.)

Vaikka SDK:ta voidaan käyttää Android-sovellusten ohjelmoimiseen komentokehotteen avulla, yleisin tapa on käyttää kehitysympäristöä. Suositeltu kehitysympäristö on Eclipsen ADT-lisäosa. Useimmat kehitysympäristöt tarjoavat graafisen käyttöliittymän kehittäjän tehtävän suoritusten nopeuttamiseksi. Koska Android-sovellukset kirjoitetaan Java-ohjelmointikielellä, täytyy ohjelmoijalla oltava asennettuna JDK. (Techopedia.)

6.2.2 JDK

JDK on ohjelmistokehitysympäristö, jota käytetään Java-ohjelmien kehittämiseen. Se sisältää JRE:n, tulkitsijan tai lataajan, kääntäjän, arkistoijan, dokumentointigeneraattorin ja muita työkaluja Javalla kehittämiseen. (Techopedia.)

Java-kehittäjille esitettiin alun perin kahta JDK työkalua, java ja javac. Molempia ajetaan komentokehotteen kautta. Java-lähdetiedostot ovat yksinkertaisia tekstitiedostoja java-laajennuksella. Kirjoitettua ja tallennettua Java-lähdekoodi, javac-kääntäjää kutsutaan luomaan class-tiedosto. Kun class-tiedosto on luotu, java-käskyä voidaan käyttää java-ohjelman ajamiseen. (Techopedia.)

Kehittäjät, jotka haluavat työskennellä kehitysympäristöllä, voivat ladata JDK:n Oracle-nettisivulta. Nämä kehitysympäristöt nopeuttavat kehitysprosessia tarjoamalla point-and-click- ja raahaa ja pudota- toiminnon sovelluksen luomiseen. (Techopedia.)

6.2.3 Android NDK

Android NDK on kokoelma työkaluja, joiden avulla voidaan käyttää C ja C++ koodia Android-sovelluksessa. Se tarjoaa kirjastoja, joilla voi hallita natiiveja toimintoja ja pääsyn fyysisen laitteen komponentteihin kuten sensoreihin. (Android Developer.)

6.2.4 Apache Ant

Apache Ant on Java-kirjasto ja komentorivityökalu, jonka tarkoituksena on ajaa prosessitiedostoissa kuvattuja prosesseja kuten ”targets” ja ”extension points”. Suurin Ant:n käyttötarkoitus on Java-ohjelmien kehittäminen. Ant tarjoaa useita sisäänrakennettuja tehtäviä mahdollistaen Java-ohjelman kääntämisen, kokoamisen, testaamisen ja ajamisen. Ant:a voi käyttää myös tehokkaasti muihinki kuin Java-ohjelmien kehittämiseen kuten C tai C++ ohjelmiin. Yleisemmin Ant:a voidaan käyttää ohjaamaan mitä tahansa prosessia, joka voidaan kuvata ”targets” ja ”task” mukaan. (Apache Ant, 2017.)

6.3 Android-sovellus

Opinnäytetyön tuloksena syntyi Android-sovellus, jossa on kaksi kerrosta. Ylimpänä on Android-käyttöliittymä ja alimpana ajuri. Android-käyttöliittymä toimii sekä graafisena käyttöliittymänä käyttäjälle että käyttää haluttuja funktioita, jotka sijaitsevat ajurissa. Ajuri on jaettu kirjasto, joka on portattu Medikron Windows-käyttöjärjestelmälle kehitetystä ohjelmasta. Ajurissa sijaitsee kaikki halutut funktiot, joita kutsutaan Android-käyttöliittymällä. Ajurissa hoidetaan myös Bluetooth-kommunikointi spirometrialaitteen kanssa.

6.3.1 Android-käyttöliittymä

Käyttöliittymäsovellus koostuu kahdesta osasta. Käyttäjälle näkyvä ohjelma on toteutettu QML:llä ja jaetun kirjaston funktioiden kutsuminen C++:lla.

```
int main(int argc, char *argv[])
{
    QGuiApplication app(argc, argv);
    DriverThread driver;

    QQuickView *view = new QQuickView;
    view->rootContext()->setContextProperty("driverThread", &driver);
    view->setSource(QUrl("qrc:/main.qml"));
    view->setResizeMode(QQuickView::SizeRootObjectToView);

    QObject *item = dynamic_cast<QObject*>(view->rootObject());

    view->show();

    QObject::connect(&driver, SIGNAL(updateStatusText(QVariant)),
                     item, SLOT(qmlUpdateStatusText(QVariant)));
    QObject::connect(&driver, SIGNAL(updateConnectState(QVariant)),
                     item, SLOT(qmlConnectState(QVariant)));

    return app.exec();
}
```

KUVA 6. Android-käyttöliittymän main-funktio (Kettunen 6.7.2017)

QGuiApplication-luokka sisältää main event-silmukan, jossa kaikki muut eventit prosessoidaan ja lähetetään. Se myös käsittelee myös käyttöliittymäsovelluksen alustuksen ja lopetuksen. app.exec()-

funktiolla käynnistetään main event-silmukka, joka on käynnissä niin pitkään kunnes QGuiApplicationin exit()-funktiota kutsutaan. (The Qt Company, 2017.)

QQuickView-luokalla luodaan ikkuna, joka näyttää käyttöliittymän ja vastaanottaa käyttäjän toimet kuten napin painalluksen. Tähän ikkunaan asetetaan URl-osoite, jossa sijaitsee käyttöliittymän lähdetiedosto, joka on tässä työssä main.qml. Lähdetiedostossa sijaitsee kaikki tiedot käyttöliittymän visuaaliseen ulkonäköön liittyen. Lähdetiedosto on ohjelmoitu QML:llä. QML on deklaratiivinen kieli, jolla ohjelmoidaan käyttöliittymän visuaaliset komponentit sekä niiden logiikka toistensa kanssa toimimiseen. setContextProperty(”driverThread”, &driver)-funktiolla asetetaan QQuickView-luokan kontekstiksi DriverThread-luokka, joka mahdollistaa datan siirtymisen C++ ja QML:n välillä. Ajamalla setResizeMode(QQuickView::SizeRootObjectToView)-funktion saadaan koko puhelimen näyttö käyttöön. QObject-luokalla otetaan haltuun QQuickView-luokan objekti, jota käytetään hyödyksi Signals and Slots-metodissa. (The Qt Company, 2017.)

DriverThread-luokassa käsitellään jaetun kirjaston funktioiden kutsuminen. DriverThread-luokan ja main.qml välinen kommunikointi on hoidettu Signals and Slots-metodilla. DriverThread-luokalla on kaksi signaalia. Joka kerta kun DriverThread-luokka emittoi signaalin, QVariant-tyyppinen parametri välitetään QML-koodissa sijaitsevaan Slot-funktioon eli joko qmlUpdateStatusText(QVariant)- tai qmlConnectState(QVariant)-funktioon.

6.3.2 Ajuri

Ajurissa on kolme kerrosta. Ylimmässä kerroksessa on funktiot, joissa sijaitsee käskyt Bluetooth-yhteyden avaamiseen ja sulkemiseen sekä datan lähettämiseen spirometrialaitteelle. Funktioita kutsutaan Android-käyttöliittymästä. Toisessa kerroksessa on itse ajuri, joka toteuttaa funktioiden käskyt. Alimpana sijaitsee Bluetooth-rajapinta, jota ajuri käyttää yhteyden muodostamiseen ja datan välittämiseen.

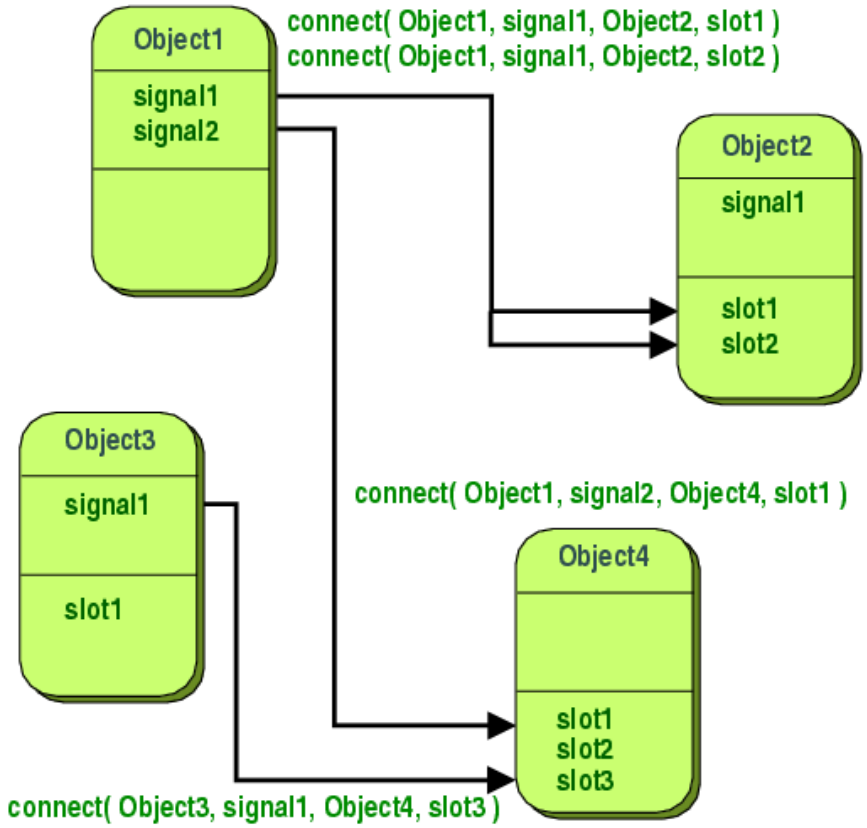
Jokaiselle yhteydelle luodaan omat säikeet datan lähettämiseen ja lukemiseen yhteyden muodostamisen aikana. Näin saadaan luotua useita yhteyksiä, jotka toimivat itsenäisesti ja eristyksissä toisistaan. Säikeen luomiseen käytetään QThread-luokkaa. Säikeet saavat käskynsä globaalista taulukosta, joita ne käyvät aktiivisesti lukemassa. Globaaliin taulukkoon käskyt kirjoitetaan ylimmässä funktiot kerroksessa.

6.3.3 Signals and Slots

Signals and Slots-metodia on Qt:n keskeisimpiä ominaisuuksia, jota käytetään objektien väliseen kommunikointiin. Tämä metodi korvaa callback-metodin. Callback-metodissa annetaan prosessoivalle funktiolle callback-funktion osoitin, jos halutaan prosessoivan funktion tekevän ilmoituksen jonkin tapahtuman jälkeen. Prosessoiva funktio kutsuu callback-funktiota osoittimen avulla. Callback-metodissa on kaksi huonoa puolta. Ensimmäinen huono puoli on, ettei ole varmaa, että prosessoiva

funktio kutsuu aina oikeilla parametreilla callback-funktiota ja toiseksi prosessoivan funktion täytyy tietää tarkalleen, mitä callback-funktiota kutsua. (The Qt Company, 2016.)

Signals and Slots-metodissa emittoidaan signaaleja, kun joku tietty event tapahtuu. Slot on funktio, jota kutsutaan tietyn signaalin emittoituessa.



KUVA 7. Signals and Slot-kaavio (The Qt Company, 2016.)

Signals and Slots-metodissa luokka, josta signaali emittoidaan ei tarvitse tietää, mikä Slot-funktio ottaa signaalin vastaan. Tarvittaessa Slot-funtio myös ignooraa ylimääräiset signaalin parametrit. Signaaleille ja Slot-funktiolle voi antaa rajattomasti parametreja. Kaikki luokat, jotka perivät QObject-luokan tai sen aliluokan, voivat sisältää signaaleja ja Slot-funktioita. Objektit emittoivat signaaleja, kun niiden tilan muutos vaikuttaa muihin objekteihin. Objektin ei tarvitse välittää vastaanottaako muu/muut objekti/objektit sen signaalit. (The Qt Company, 2016.)

6.3.4 Jaettu kirjasto

Jaettu kirjasto on osa dynaamisen kirjaston ohjelmointikonseptia. Jaetun kirjaston toimintoja käytetään vain ohjelman suorituksen aikana, mikä minimoi kokonaisuudessaan ohjelman kokoa ja parantaa ohjelman suorituskykyä pienemmällä muistin kulutuksella. Useimmissa ohjelmistoissa toimintojen jakaminen eri moduuleihin mahdollistaa niiden käyttämisen tarpeen mukaan. Jaettu kirjasto ei ole koskaan osa suoritettavaa tiedostoa tai ohjelmaa. Ajon aikana luodaan linkki jaetun kirjaston ja suoritettavan tiedoston tai ohjelman välille. (Techopedia.)

6.3.5 QThread-luokka

QThread-luokka mahdollistaa alustariippumattoman tavan hallita säikeitä. QThread-objekti hallitsee yhtä säiettä ohjelman sisällä. Heti luomisensa jälkeen QThread aloittaa ajamalla run()-funktion. Oletuksena run()-funktio aloittaa luomalla uuden säikeen kutsumalla exec()-funktiota. Tämän jälkeen Qt event-silmukkaa ajetaan säikeen sisällä. (The Qt Company, 2017.)

```
class Worker : public QObject
{
    Q_OBJECT

public slots:
    void doWork(const QString &parameter) {
        QString result;
        /* ... here is the expensive or blocking operation ... */
        emit resultReady(result);
    }

signals:
    void resultReady(const QString &result);
};
```

KUVA 8. Esimerkin Worker-luokasta (The Qt Company, 2017.)

Esimerkissä luodaan Worker-luokka, joka tulee toimimaan omassa säikeessä. Worker-luokassa on Slot-funktio doWork(), jossa käytetään saatua parametria ja lopuksi emittoidaan resultReady()-signaali, jonka parametrina on tulos.

```
class Controller : public QObject
{
    Q_OBJECT
    QThread workerThread;
public:
    Controller() {
        Worker *worker = new Worker;
        worker->moveToThread(&workerThread);
        connect(&workerThread, &QThread::finished, worker, &QObject::deleteLater);
        connect(this, &Controller::operate, worker, &Worker::doWork);
        connect(worker, &Worker::resultReady, this, &Controller::handleResults);
        workerThread.start();
    }
    ~Controller() {
        workerThread.quit();
        workerThread.wait();
    }
public slots:
    void handleResults(const QString &);
signals:
    void operate(const QString &);
};
```

KUVA 9. Esimerkin Controller-luokka (The Qt Company, 2017.)

Seuraavaksi luodaan Controller-luokka, jonka muodostimessa luodaan Worker-objekti ja yksityisenä jäsenenä QThread workerThread. Worker-objekti siirretään workerThread-säikeeseen käyttämällä QObject::moveToThread()-funktiota. Näin Worker-objekti saadaan toimimaan omassa säikeessä. Tämän jälkeen yhdistetään Controller-luokan operate()-signaali worker-objektin doWork()-funktioon ja worker-objektin resultReady()-signaali Controller-luokan handleResults()-funktioon. Lopuksi ajetaan QThread::start()-funktio workerThread-säikeen käynnistämiseksi. Nyt Controller-luokalla emittoimalla operate()-signaalin voidaan käyttää worker-objektin doWork()-funktiota, joka palauttaa tuloksen emittoimalla resultReady()-signaalin. resultReady()-signaali vastaanotetaan Controller-luokan handleResults()-funktiolla.

6.4 Bluetooth Low Energyn rajapinnan ohjelmoiminen Qt:lla

Bluetooth Low Energyn rajapinnan voi ohjelmoida asiakkaan ja/tai palvelimen puolelta. Tässä työssä ohjelmoitiin asiakkaan puolelta. Asiakkaan puolella API mahdollistaa yhteyksien luomisen oheislaitteisiin, niiden palvelujen löytämiseen ja palveluihin varastoidun datan lukemiseen ja kirjoittamiseen. Palvelimen puolella voidaan asettaa palvelut, mainostaa palveluita ja saada ilmoituksia, kun asiakas kirjoittaa karaktereihin. Esimerkissä käytetään hyödyksi koodia Heart Rate Gamesta (The Qt Company, 2017) ja Heart Rate Serverista. (The Qt Company, 2017.)

6.4.1 Yhteyden muodostaminen

Jotta Bluetooth Low Energy oheislaitteen karaktereihin voitaisiin kirjoittaa ja lukea, täytyy ensimmäisenä etsiä laite ja sen jälkeen yhdistäytyä siihen. Tämä vaatii sen, että oheislaite mainostaa olemassa oloaan ja palveluitaan. Laitteen etsimiseen käytetään apuna QBluetoothDeviceDiscoveryAgent-luokkaa. Etsintä aloitetaan QBluetoothDeviceDiscoveryAgent-luokan start-funktiolla. (The Qt Company, 2017.)

```
m_deviceDiscoveryAgent = new QBluetoothDeviceDiscoveryAgent(this);
m_deviceDiscoveryAgent->setLowEnergyDiscoveryTimeout(5000);

connect(m_deviceDiscoveryAgent, &QBluetoothDeviceDiscoveryAgent::deviceDiscovered, this,
&DeviceFinder::addDevice);
connect(m_deviceDiscoveryAgent, static_cast<void (QBluetoothDeviceDiscoveryAgent::*)
(QBluetoothDeviceDiscoveryAgent::Error)>(&QBluetoothDeviceDiscoveryAgent::error),
        this, &DeviceFinder::scanError);

connect(m_deviceDiscoveryAgent, &QBluetoothDeviceDiscoveryAgent::finished, this,
&DeviceFinder::scanFinished);
connect(m_deviceDiscoveryAgent, &QBluetoothDeviceDiscoveryAgent::canceled, this,
&DeviceFinder::scanFinished);
m_deviceDiscoveryAgent->start(QBluetoothDeviceDiscoveryAgent::LowEnergyMethod);
```

KUVA 10. QBluetoothDeviceDiscoveryAgentin signaalien yhdistäminen (The Qt Company, 2017.)

Kuvassa 8 luodaan uusi QBluetoothDeviceDiscoveryAgent-luokka ja asetetaan etsintäajaksi 5000 millisekunttia. Tämän jälkeen yhdistetään QBluetoothDeviceDiscoveryAgent tuottama signaali ”deviceDiscovered” DeviceFinder-luokan funktioon ”addDevice”. Eli kun QBluetoothDeviceDiscoveryAgent löytää oheislaitteen, se lähettää deviceDiscoverd-signaalin. Kun signaali on lähetetty, suoritetaan

funktio addDevice. Tässä esimerkissä ollaan kiinnostuneita vain Low Energy-laitteista, joten suodatamme laitteen tyypin addDevice-funktiossa. Laitteen tyyppi voidaan selvittää käyttämällä QBluetoothDeviceInfo::coreConfigurations()-lippua. (The Qt Company, 2017.)

```
void DeviceFinder::addDevice(const QBluetoothDeviceInfo &device)
{
    // If device is LowEnergy-device, add it to the list
    if (device.coreConfigurations() & QBluetoothDeviceInfo::LowEnergyCoreConfiguration) {
        m_devices.append(new DeviceInfo(device));
        setInfo(tr("Low Energy device found. Scanning more..."));
    }
    //...
}
```

KUVA 11. Low Energy-laitteiden suodatus (The Qt Company, 2017.)

Kun halutun oheislaitteen osoite on tiedossa, käytetään QLowEnergyController-luokkaa. Tämä luokka on Bluetooth Low Energyn kehittämisen kulmakivi. Luokan muodostin tarvitsee parametriksi oheislaitteen QBluetoothAddressin eli laitteen osoitteen. Lopuksi asetetaan funktiot QLowEnergyController-signaaleille ja yhdistetään suoraan oheislaitteeseen käyttämällä QLowEnergyControllerin connectToDevice-funktiota. (The Qt Company, 2017.)

```
m_control = new QLowEnergyController(m_currentDevice->getDevice(), this);
connect(m_control, &QLowEnergyController::serviceDiscovered,
        this, &DeviceHandler::serviceDiscovered);
connect(m_control, &QLowEnergyController::discoveryFinished,
        this, &DeviceHandler::serviceScanDone);

connect(m_control, static_cast<void (QLowEnergyController::*)(QLowEnergyController::Error)>
(&QLowEnergyController::error),
        this, [this](QLowEnergyController::Error error) {
    Q_UNUSED(error);
    setError("Cannot connect to remote device.");
});
connect(m_control, &QLowEnergyController::connected, this, [this]() {
    setInfo("Controller connected. Search services...");
    m_control->discoverServices();
});
connect(m_control, &QLowEnergyController::disconnected, this, [this]() {
    setError("LowEnergy controller disconnected");
});

// Connect
m_control->connectToDevice();
```

KUVA 12. Yhdistäminen oheislaitteeseen (The Qt Company, 2017.)

6.4.2 Palvelun etsiminen

Kuvan 10 näyttämä koodi aloittaa palveluiden etsimisen, kun oheislaitteeseen on muodostettu yhteys. Kun QLowEnergyController-luokan serviceDiscovered-signaali on lähetetty, suoritetaan serviceDiscovered()-funktio. Funktiossa annetaan myös ajottainen edistysraportti. Koska esimerkissä käsitellään sykkeen kuuntelu-sovellusta, joka seuraa läheisiä HeartRate-laitteita, sivuutetaan kaikki muut palvelutyypit paitsi QBluetoothUuid::HeartRate. (The Qt Company, 2017.)

```
void DeviceHandler::serviceDiscovered(const QBluetoothUuid &gatt)
{
    if (gatt == QBluetoothUuid(QBluetoothUuid::HeartRate)) {
        setInfo("Heart Rate service discovered. Waiting for service scan to be done...");
        m_foundHeartRateService = true;
    }
}
```

KUVA 13. Ajottaisen edistysraportin antaminen (The Qt Company, 2017.)

Lopulta QLowEnergyController-luokan discoveryFinished-signaali on lähetetty ilmaisemaan onnistuneesta palvelun etsinnästä. Kun HeartRate-palvelu on löydetty, luodaan QLowEnergyService-instanssi esittämään palvelua. Takaisin saadulla palvelu objektin avulla saadaan tarvittavat signaalit päivitysilmoituksia varten. Palvelun yksityiskohtien etsimiseen käytetään QLowEnergyService-luokan discoverDetails-funktiota. (The Qt Company, 2017.)

```
    // If heartRateService found, create new service
    if (m_foundHeartRateService)
        m_service = m_control->createServiceObject(QBluetoothUuid(QBluetoothUuid::HeartRate),
this);

    if (m_service) {
        connect(m_service, &QLowEnergyService::stateChanged, this,
&DeviceHandler::serviceStateChanged);
        connect(m_service, &QLowEnergyService::characteristicChanged, this,
&DeviceHandler::updateHeartRateValue);
        connect(m_service, &QLowEnergyService::descriptorWritten, this,
&DeviceHandler::confirmedDescriptorWrite);
        m_service->discoverDetails();
    } else {
        setError("Heart Rate Service not found.");
    }
```

KUVA 14. Palvelun yksityiskohtien etsintä (The Qt Company, 2017.)

Yksityiskohtien etsinnän aikana palvelun tila muuttuu DiscoveryRequired:sta DiscoveringServices:een ja lopulta ServiceDiscoverd. (The Qt Company, 2017.)

```
void DeviceHandler::serviceStateChanged(QLowEnergyService::ServiceState s)
{
    switch (s) {
    case QLowEnergyService::DiscoveringServices:
        setInfo(tr("Discovering services..."));
        break;
    case QLowEnergyService::ServiceDiscovered:
    {
        setInfo(tr("Service discovered."));

        const QLowEnergyCharacteristic hrChar = m_service-
>characteristic(QBluetoothUuid(QBluetoothUuid::HeartRateMeasurement));
        if (!hrChar.isValid()) {
            setError("HR Data not found.");
            break;
        }

        m_notificationDesc = hrChar.descriptor(QBluetoothUuid::ClientCharacteristicConfiguration);
        if (m_notificationDesc.isValid())
            m_service->writeDescriptor(m_notificationDesc, QByteArray::fromHex("0100"));

        break;
    }
    default:
        //nothing for now
        break;
    }

    emit aliveChanged();
}
```

KUVA 15. Palvelun tilan muuttumisen tutkiminen (The Qt Company, 2017.)

6.4.3 Oheislaitteen kanssa keskusteleminen

Kuvan 11 koodista nähdään, että haluttu karakterin tyyppi on HeartRateMeasurement. Koska sovellus mittaa sydämen sykkeen vaihtelua, täytyy karakterin muutosilmoitukset asettaa päälle. Huomaa, että kaikissa karaktereissa ei ole tätä toimintoa. Koska HeartRate karakteri on standartoitu, voidaan olettaa, että oheislaite lähettää ilmoituksia. Ennen kaikkea QLowEnergyCharacteristic-luokalla täytyy olla Notify-lippu asetettuna ja deskriptorin ClientCharacteristicConfiguration-tyyppi täytyy olla olemassa, jotta ilmoitukset ovat asianmukaisesti saatavilla. Lopuksi käsitellään vastaanotettu HeartRate-karakterin arvo kuvassa 14.

```
void DeviceHandler::updateHeartRateValue(const QLowEnergyCharacteristic &c, const QByteArray
&value)
{
    // ignore any other characteristic change -> shouldn't really happen though
    if (c.uuid() != QBluetoothUuid(QBluetoothUuid::HeartRateMeasurement))
        return;

    const quint8 *data = reinterpret_cast<const quint8 *>(value.constData());
    quint8 flags = data[0];

    //Heart Rate
    int hrvalue = 0;
    if (flags & 0x1) // HR 16 bit? otherwise 8 bit
        hrvalue = (int)qFromLittleEndian<quint16>(data[1]);
    else
        hrvalue = (int)data[1];

    addMeasurement(hrvalue);
}
```

KUVA 16. Vastaanotetun arvon käsittely (The Qt Company, 2017.)

Kuvassa 14 nähdään, miten luetaan standardoidun HeartRate-karakterin arvo. Yleisesti karakterin arvo on sarja tavuja. Tavujen tulkinta riippuu karakterin tyypistä ja arvon rakenteesta. Bluetooth SIG on standardisoinnut merkittävän määrän karakterin tyyppejä ja niiden arvojen rakenteita. (The Qt Company, 2017.)

7 KEHITYSIDEAT

Android-puhelimen ja spirometrialaitteen yhdistämiseen kuluva aika riippuu, onko spirometrialaite päällä ennen vai jälkeen skannauksen. Jos spirometrialaite on päällä ennen skannauksen aloittamista, laitteen yhdistämiseen kuluu reilu 10 sekuntia. Jos spirometrialaite käynnistetään skannaamisen kanssa yhtä aikaa tai sen jälkeen, yhdistämiseen kulunut aika on jopa reilut 20 sekuntia. Tästä voidaan päätellä, että yhdistämisen nopeuttamiseksi voidaan spirometrialaite kytkeä päällä ennen kuin aloittaa skannauksen tai nopeuttaa spirometrialaitteen käynnistystä.

Uusimmassa Bluetooth 5-standardissa on neljä kertaa pidempi kantama, kaksi kertaa nopeampi tiedonsiirto ja kahdeksan kertaa suurempi mainostettavan viestin kapasiteetti kuin vanhemmassa Bluetooth standardissa. Pidennetyllä kantamalla ei välttämättä ole niin paljon hyötyä, koska spirometrialaite ja Android-puhelin tulevat todennäköisesti muutenkin olemaan käytössä lähietäisyydellä toisistaan. Nopeamman tiedonsiirron ansiosta saadaan pienempi vasteaika Android-puhelimen ja spirometrialaitteen välille. Tämä mahdollistaa myös sen, että voidaan siirtää enemmän dataa spirometrialaitteelta puhelimelle ja takaisin spirometrialaitteelle ilman, että lähetysaika pidentyisi. Suuremman mainostettavan viestin kapasiteetin avulla spirometrialaite voisi välittää dataa Android-puhelimelle ilman, että muodostettaisiin yhteyttä näiden välille. Lähetettävän datan voisi sisällyttää mainostettavaan viestiin. Tällä tavalla saataisiin dataa vastaanotettua paljon nopeammin. Jotta Bluetooth 5-standardin ominaisuuksia voitaisiin hyödyntää sekä spirometrialaitteessa että Android-puhelimessa, täytyy olla Bluetooth 5-standardi käytössä. Tämän lisäksi uusin Nougat Android-käyttöjärjestelmä ei tue Bluetooth 5-standardia, mutta tulossa oleva Android O-käyttöjärjestelmä tukee.

8 LÄHTEET

Android Developer. Application Fundamentals. *Android Developer.* [Online] [Viitattu: 27. 6 2017.] https://developer.android.com/guide/components/fundamentals.html.

—. Getting Started with the NDK. *Android.* [Online] [Viitattu: 16. 6 2017.] https://developer.android.com/ndk/guides/index.html.

—. 2017. Meet Android Studio. *Android Developer.* [Online] 2017. [Viitattu: 19. 6 2017.] https://developer.android.com/studio/intro/index.html.

—. Platform Architecture. *Developer Android.* [Online] [Viitattu: 27. 6 2017.] https://developer.android.com/guide/platform/index.html.

Apache Ant. 2017. Apache Ant. *Apache Ant.* [Online] 2017. [Viitattu: 16. 6 2017.] http://ant.apache.org/.

Bluetooth SIG. 2017. Bluetooth 5: What it's all about. *Bluetooth.* [Online] 2017. [Viitattu: 13. 6 2017.] https://www.bluetooth.com/specifications/bluetooth-core-specification/bluetooth5.

—. 2017. How it Works. *Bluetooth.* [Online] 2017. [Viitattu: 13. 6 2017.] https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works.

—. 2017. membership & working groups. *Bluetooth.* [Online] 2017. [Viitattu: 13. 6 2017.] https://www.bluetooth.com/membership-working-groups.

—. 2017. Our History. *Bluetooth.* [Online] 2017. [Viitattu: 13. 6 2017.] https://www.bluetooth.com/about-us/our-history.

Kinnula, Vuokko ja Sovijärvi, Anssi. 2005. Diagnostiset tutkimukset. *Keuhkosairaudet.* Helsinki : Kustannus Oy Duodecim, 2005.

Piirilä, Päivi. 2013. Keuhkojen toiminnan tutkiminen. *Keuhkosairaudet.* Helsinki : Kustannus Oy Duodecim, 2013, ss. 22-30.

Sovijärvi, Anssi ja Piirilä, Päivi. 2003. Ventilaatiokyvy ja kehkotilavuuksien mittaukset. *Kliininen fysiologia ja isotooppilääketiede.* Helsinki : Kustannus Oy Duodecim, 2003.

Techopedia. Android Operating System. *Techopedia.* [Online] [Viitattu: 17. 7 2017.] https://www.techopedia.com/definition/25106/android-operating-system.

—. Android SDK. *Techopedia.* [Online] [Viitattu: 16. 6 2017.] https://www.techopedia.com/definition/4220/android-sdk.

—. Dynamic library. *Techopedia.* [Online] [Viitattu: 16. 6 2017.] https://www.techopedia.com/definition/27133/dynamic-library.

—. Java Development Kit (JDK). *Techopedia.* [Online] [Viitattu: 16. 6 2017.] https://www.techopedia.com/definition/5594/java-development-kit-jdk.

The Qt Company. 2017. About Qt. *Qt Wiki.* [Online] 14. 2 2017. [Viitattu: 20. 6 2017.] http://wiki.qt.io/About_Qt.

—. 2017. Bluetooth Low Energy Heart Rate Game. *Qt.* [Online] 2017. [Viitattu: 14. 6 2017.] http://doc.qt.io/qt-5/qtbluetooth-heartrate-game-example.html.

—. 2017. Bluetooth Low Energy Heart Rate Server Example. *Qt.* [Online] 2017. [Viitattu: 14. 6 2017.] http://doc.qt.io/qt-5/qtbluetooth-heartrate-server-example.html.

—. 2017. Bluetooth Low Energy Overview. *Qt.* [Online] 2017. [Viitattu: 14. 6 2017.] http://doc.qt.io/qt-5/qtbluetooth-le-overview.html.

—. 2017. Qt Creator Manual. *Doc Qt.* [Online] 2017. [Viitattu: 20. 6 2017.] http://doc.qt.io/qtcreator/index.html.

—. 2017. Qt Doc. *IDE Overview.* [Online] 2017. [Viitattu: 20. 6 2017.] http://doc.qt.io/qtcreator/creator-overview.html.

—. 2017. Qt Documentation. *Qt.* [Online] 2017. [Viitattu: 5. 7 2017.] http://doc.qt.io/.

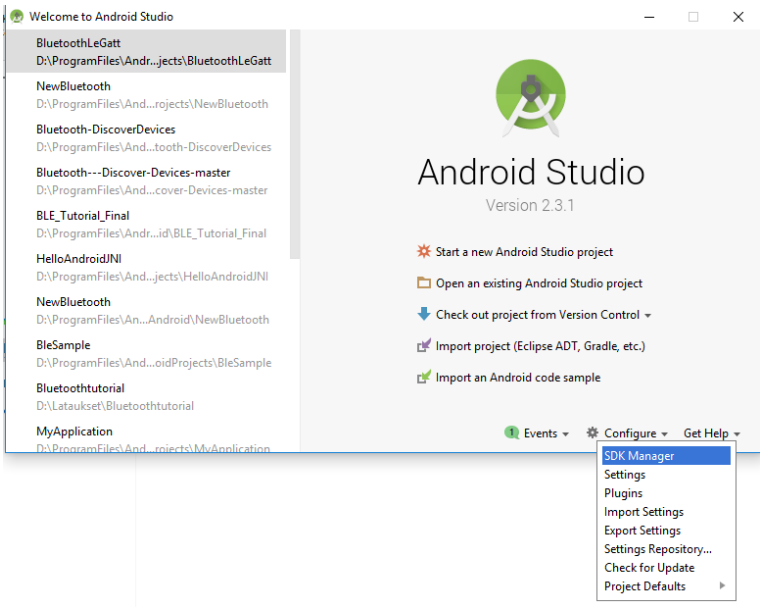
—. 2016. Qt Documentation. *Signals & Slots.* [Online] 2016. [Viitattu: 6. 7 2017.] http://doc.qt.io/qt-4.8/signalsandslots.html.

—. 2017. QThread Class. *Qt Documentation.* [Online] 2017. [Viitattu: 10. 7 2017.] http://doc.qt.io/qt-5/qthread.html.

Xamarin . 2017. Introduction to Mobile Development. *Developer Xamarin.* [Online] 2017. [Viitattu: 19. 6 2017.] https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/#Introduction_to_Xamarin.

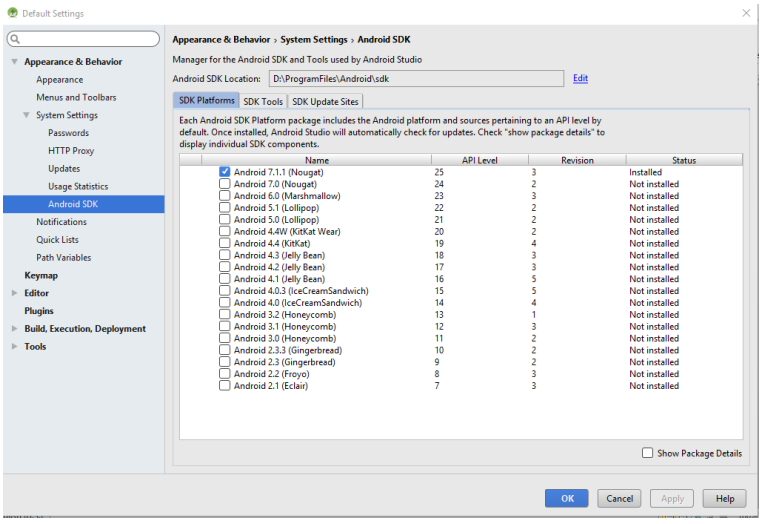
LIITE 1: QT:N ASENNUS

Nämä asennusohjeet toimivat Windows 7:lla ja 10:llä. Komponentit, joita tarvitaan Qt:lla Android-sovelluksen kehittämiseen ovat Android Studio, Qt Creator, Apache Ant, JDK ja NDK. Ensimmäisenä ladataan ja asennetaan JDK. Sen jälkeen uusin Android Studio-kehitysympäristö. Asenna ja aukaise Android Studio.



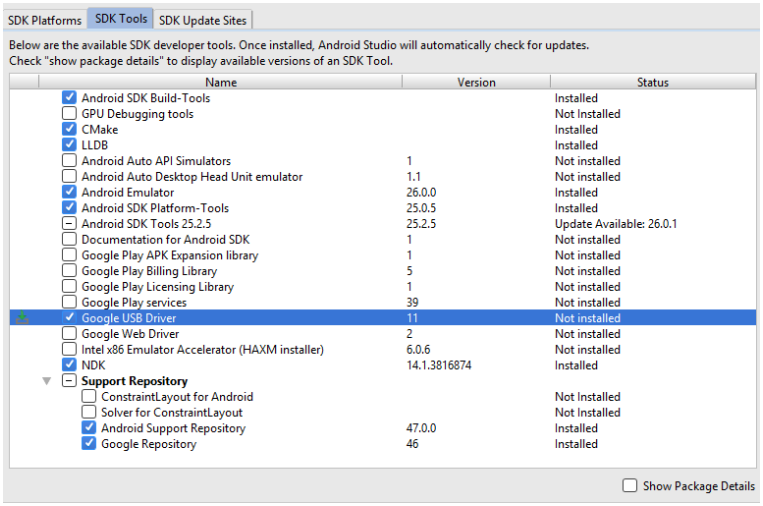
KUVA 17. Android Studio:n aloitusnäkymä

Android Studio:ssa on sulautettuna SDK Manager, jonka avulla saadaan ladattua NDK ja API. Valitse alaoikeanurkasta Configure -> SDK Manager.



KUVA 18. API:n valinta SDK Managerilla

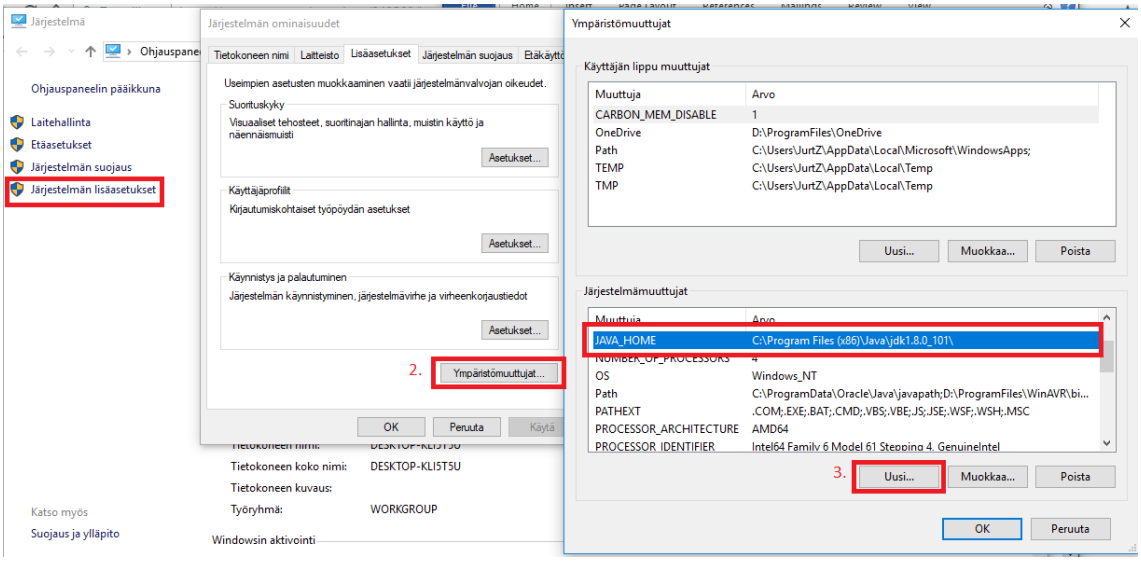
Valitse haluamasi API. Tässä työssä asennettiin API 25. Seuraavana valitse välilehdistä SDK Tools ja valitse ja asenna Android SDK Build-Tools, CMake, LLDB, Android SDK Platform-Tools, Google USB Driver, NDK, Android Support Repository ja Google Repostiory.



KUVA 19. SDK Tools-välilehti

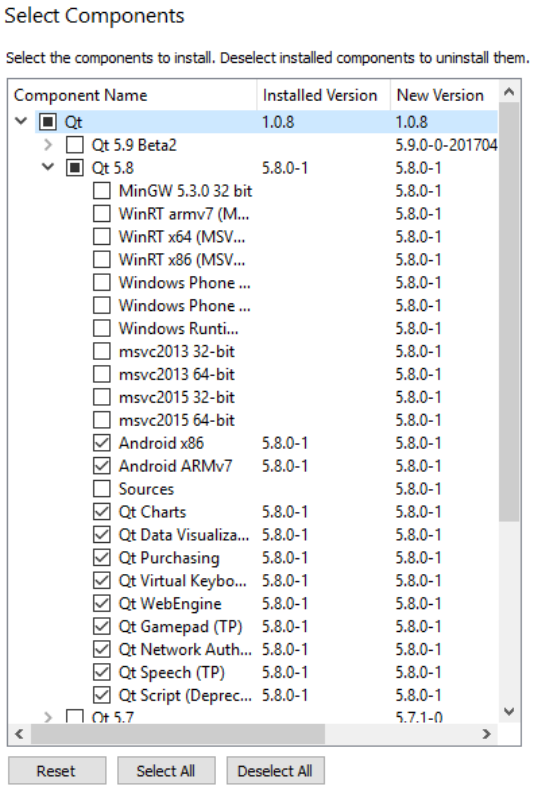
Asentamisen jälkeen korvataan NDK vanhemmalla versiolla, koska uusimmassa versiossa on virhe, joka estää yhteentoimivuuden Qt:n kanssa. Lataa vanhempi versio NDK r10e, kopioi lataamasi tools-tiedosto, mene asentamaasi SDK-tiedostoon ja korvaa SDK-tiedoston sisällä oleva tools-tiedosto.

Seuraavana asetetaan ympäristömuuttuja. Paina näppäimistöltä Windows- ja Break-painiketta. Windows-painikkeessa on Windows:n logo ja Break-painike ovat samassa Pause-painikkeen kanssa.



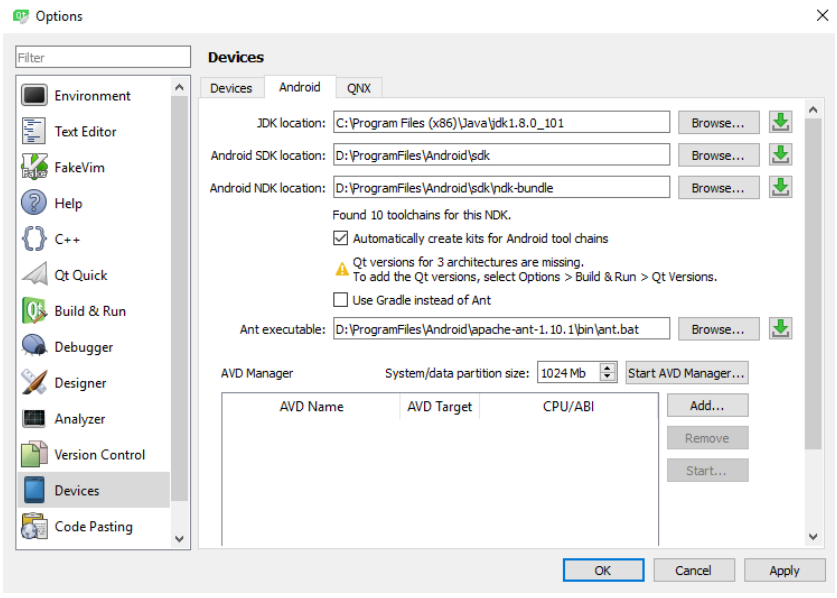
KUVA 20. Ympäristömuuttujan asettaminen

Valitse Järjestelmän lisäasetukset -> Ympäristömuuttujat -> Uusi. Kirjoita Muuttuja-kohtaan JAVA_HOME ja poluksi anna asentamasi JDK-polku. Lopuksi valitse OK. Lataa Apache Ant ja Qt Creator. Asenna ja avaa Qt Creator. Valitse ja asenna uusimmat komponentit, jotka on merkattu kuvaan 5.



KUVA 21. Qt:n komponenttien asennus näkymä

Asentamisen jälkeen mene Qt Creator:n aloitusnäkymältä mene ylätyökalupalkista Tools -> Options -> Devices. Aseta JDK:lle, SDK:lle, NDK:lle ja Ant:lle polut, minne olet ne asentanut.



KUVA 22. Polkujen asettaminen

Valitse Apply ja käynnistä Qt Creator uudelleen ottaaksesi asetetut polut käyttöön. Nyt Qt on käyttövalmis Android-sovelluksen kehittämiseen.