Opinnäytetyö (AMK)
Tietotekniikka
Sulautetut ohjelmistot
2015

Heikki Kangas

BLUETOOTH SMART ANDROID-KEHITYKSESSÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ
TURUN AMMATTIKORKEAKOULU
Tietotekniikka | Sulautetut ohjelmistot
2015 | Sivumäärä: 35
Jari-Pekka Paalassalo

Heikki Kangas

BLUETOOTH SMART ANDROID-KEHITYKSESSÄ

Tässä opinnäytetyössä tutustuttiin Bluetooth-teknologian uusimman sukupolven virtapihiin ominaisuuteen, Bluetooth Smartiin, sekä sille tarjolla oleviin kehitystyökaluihin. Tarkoituksena oli luoda yleiskatsaus Bluetooth Smartin mahdollisuuksiin Android-käyttöjärjestelmässä sovelluskehityksen näkökulmasta.

Teoriaosuudessa käytiin läpi Bluetoothin ja Bluetooth Smartin historiaa, tulevaisuutta, ominaisuuksia sekä vertailua näiden kahden välillä. Fyysisenä laitteena työssä käytettiin Texas Instrumentsin CC2541 SensorTag -kehitystyökalua, jonka avulla Smart-teknologian ominaisuuksiin tutustuminen on suhteellisen helppoa. Ohjelmoinnissa käytettiin Android Studio -kehitysympäristöä.

Työn tuloksena syntyi Bluetooth Smart -yhteensopiva Android-sovellusmalli, jota olisi jatkojalostuksella mahdollista kehittää osaksi laajempaa sovellusprojektia.

ASIASANAT:

Bluetooth, Smart, Low Energy, Android, langaton, SensorTag

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES
Information Technology | Embedded software
2015 | Total number of pages: 35
Jari-Pekka Paalassalo

Heikki Kangas

BLUETOOTH SMART IN ANDROID SOFTWARE DEVELOPMENT

The purpose of this thesis was to become familiar the newest addition to the latest Bluetooth generation, Bluetooth Smart, and the tools available for development. The goal was to obtain an overview for the possibilities of Bluetooth Smart in Android operating system software development environment.

The theoretical part of the thesis discusses and compares the history and the features of both Bluetooth and Bluetooth Smart.

In the practical part of the thesis, the development tool used for this work was Texas Instruments CC2541 SensorTag development kit which offers an easy-to-approach way to become familiar with the properties of Smart technology. The programming was carried out in Android Studio IDE.

The result of this project was a Bluetooth Smart-compatible Android software template which with some refinements could be used in any comprehensive software project.

KEYWORDS:

Bluetooth, Smart, Low Energy, Android, wireless, SensorTag

SISÄLTÖ

KUVAT

TAULUKOT

KOODIT

KÄYTETYT LYHENTEET

ADC	Analog-to-digital converter. Analogisen signaalin muuntaminen digitaaliseen muotoon.
AFH	Adaptive Frequency Hopping. Taajuushyppelyn variaatio, jolla vältetään ruuhkaisimmat lähetyskanavat.
API	Application Programming Interface. Ohjelmointirajapinta, jonka kautta ohjelmistot keskustelevat keskenään.
FHSS	Frequency Hopping Spread Spectrum. Koodijakokanavoinnissa käytettävä tekniikka, joka vaihtelee lähetystaajuutta jatkuvasti.
IDE	Integrated Development Environment. Ohjelmointiympäristö, joka yksinkertaisimmillaan koostuu tekstieditorista ja kääntäjästä.
IoT	The Internet of Things. Teollinen internet, eli yksilöitävien laitteiden joukko internetin infrastruktuurissa.
IPSP	Internet Protocol Support Profile. Ominaisuus, joka mahdollista IPv6-tuen Bluetooth Smart -laitteissa.
IPv6	Internet Protocol version 6. Laajennettu internet-pakettien välityksessä käytetty protokolla.
RSSI	Received Signal Strength Indication. Signaalin voimakkuuden suure.
SIG	(Bluetooth) Special Interest Group. Bluetoothin kehitystä valvova organisaatio.

USART	Universal asynchronous receiver/transmitter. Muuntaja digitaalisen sarja- ja rinnakkaisliikenteen välillä.
UUID	Universally Unique Identifier. 128-bittinen yksilötunniste.
WLAN	Wireless Local Area Network. Langaton lähiverkkotekniikka.
WYSIWYG	What You See Is What You Get. Ohjelmisto, jossa muokattava sisältö pyritään esittämään käyttäjälle mahdollisimman paljon lopputulosta vastaavana.

1 JOHDANTO

Älylaitteiden räjähdysmäinen kasvu 2000-luvun alusta lähtien on muuttanut pysyvästi ihmisten suhdetta teknologiaan. Yhä suurempi enemmistö ihmisistä kantaa pävittäin mukanaan laitteita, jotka ovat jatkuvassa yhteydessä internetin eksponentiaalisesti kasvavaan datamäärään. Kyse on älypuhelimista, jotka ovat suhteellisen lyhyessä ajassa muodostuneet yleishyödykkeiden sijaan lähes korvaamattomiksi.

Älypuhelinten valtava suosio on osaltaan raivannut tietä myös sellaiselle teknologialle, joka ennen oli mahdollista lähinnä tieteiskirjallisuudessa. Esimerkkinä, joskaan ei vielä ihan jokapäiväisenä ilmiönä, voidaan pitää Android Wear -käyttöjärjestelmää, joka on tarkoitettu Android-isäntälaitteiden kanssa yhdistettäviin niin sanottuihin "puettaviin" teknologioihin, kuten älykelloihin (Wareable 2015). Toinen hieman tunnetumpi esimerkki on Google Glass, tietokone älylasien muodossa, joka vähäisestä suosiostaan huolimatta on pohjustanut kiinnostusta lisättyyn todellisuuteen (Wikipedia 2015).

Tässä opinnäytetyössä tutustutaan Bluetooth Smart -teknologiaan, joka alhaisen virrankulutuksensa ansiosta tulee olemaan keskeinen osa langattomien älylaitteiden kehitystä. Tavoitteena oli luoda toimiva Android-sovellusmalli, jota olisi mahdollista soveltaa minkä tahansa Smart-laitteen vaatimusten mukaan. Työosuudessa käytettiin Texas Instrumentsin SensorTag -kehitystyökalua demonstroimaan Bluetooth Smartin ominaisuuksia.

2 BLUETOOTH

Bluetooth on langattoman tiedonsiirron standardi, jonka avulla elektroniset laitteet voivat luoda laitepareja sekä keskustella lähietäisyyksillä keskenään. Nykyään Bluetooth on laajasti käytössä monissa eri yhteyksissä viihde-elektroniikasta hyvinvointiteknologiaan, ja sitä voidaankin pitää yhtenä suurimmasta tavaramerkeistä langattomien tekniikoiden saralla yhdessä WLANin kanssa. Tavallisimpia jokapäiväisiä käyttökohteita ovat mm. tietokoneiden langattomat hiiret ja näppäimistöt, kuulokkeet, peliohjaimet sekä älypuhelinten oheislaitteet. (Bluetooth Technology Website 2015)

Bluetoothin kehittäminen alkoi vuonna 1994, jolloin ruotsalainen telekommunikaatiojärjestelmien valmistaja Ericsson halusi luoda langattoman vaihtoehdon RS-232-seriaalidatakaapeleille. Nimi Bluetooth otettiin käyttöön 900-luvulla hallinneen tanskalaisen viikinkuningas Harald Sinihampaan mukaan, joka yhdisti Tanskan heimot yhdeksi kuningaskunnaksi. Logo (kuva 1) on yhdistelmä Haraldin riimumuotoisista nimikirjaimista. (Bluetooth Technology Website 2015)

Kuva 1. Bluetooth-logo. (Flicksoftware 2014)

Vuonna 1998 Ericsson, Intel, Nokia, Toshiba ja IBM perustivat Bluetooth Special Interest Groupin, voittoa tavoittelemattoman järjestön, jonka tarkoituksena on valvoa Bluetooth-teknologian kehitystä ja lisenssien jakoa yritysten käyttöön (Radio-Electronics 2015). SIG:ssa on nykyään yli 25 000 jäsenyhtiötä (Bluetooth Technology Special Interest Group 2015).

2.1 Versiot

Bluetoothin ensimmäinen versio julkaistiin heinäkuussa 1999 ja vuoden 2014 loppuun mennessä viimeisin versio oli 4.2. Tärkeimmät versiopäivitykset esitellään taulukossa 1. (Bluetooth Technology Website 2015)

Taulukko 1. Bluetoothin versiot. (Wikipedia 2015)

Versio	Julkaistu	Datansiirtonopeus*	Erikoista
1.0	1999	< 1 Mb/s	
2.0	2004	3 Mb/s	Enhanced data rate
3.0	2009	24 Mb/s	High speed
4.0	2010	24 Mb/s / 1 Mb/s	Low energy

* teoreettinen maksimi

Ensimmäinen versio sisälsi paljon ongelmia erityisesti laitteiden yhteensopivuudessa. Vasta vuonna 2003 julkaistu versio 1.2 toi mukanaan päivityksiä, joiden jälkeen Bluetoothia saatettiin pitää vakavasti otettavana langattomana datansiirtoteknologiana. (Wikipedia 2015)

Versiossa 2.0 Bluetooth sai uuden ominaisuuden, EDR:n, jonka myötä käytännöllinen datansiirtonopeus saatiin lähes kolminkertaistettua. Tätäkin suurempi harppaus otettiin versioon 3.0, jossa datansiirto tapahtuu WLANin avulla jopa 24 Mb:n/s nopeudella. Käytännössä ainoastaan yhteyden muodostus tapahtuu Bluetoothin omilla kanavalinkeillä. (Wikipedia 2015)

Versiossa 4.0 esiteltiin vähäisestä virrankulutuksesta tunnettu Low Energy -ominaisuus, joka esitellään tarkemmin luvussa 3.

2.2 Kanavat ja taajuudet

Bluetooth-lähetin toimii 2,4 — 2,4835 GHz:n ISM-taajudella, joka on muutamaa poikkeusta lukuunottamatta vapaasti käytettävissä maailmanlaajuisesti. Esimerkiksi Espanjanssa, Ranskassa ja Japanissa ISM:n käyttöä on rajoitettu enemmän. Bluetoothin käyttämä taajuusalue on jaettu 79 eri kanavaan 1 MHz:n välein alkaen 2,402 GHz:stä ja päättyen 2,480 GHz:iin jättäen alueen molempiin pähin 2 ja 3,5 MHz:n suoja-alueet (guard band). (Radio-Electronics 2015)

Käytetyn taajuusalueen ongelma on se, että se on käytössä myös muissa langattomissa tiedonsiirtojärjestelmissä, kuten yleisimpänä esimerkkinä WLANissa, jotka usein joutuvat pakostakin toimimaan samoilla taajuusalueilla. WLAN käyttää Bluetoothista poiketen kerralla yhtä kaistanleveydeltään 22 MHz:n kanavaa. (Hewlett-Packard 2002)

Varhaiset Bluetooth-laitteet saattoivat käyttää WLANin kanssa samoja taajuuksia pitkiä aikoja aiheuttaen häiriötä ja estäen lopulta molempien laitteiden toiminnan. Aiheutetun häiriön minimointia varten SIG otti Bluetoothia varten käyttöön taajuushyppelyn, FHSS, jonka periaatteiden mukaan Bluetooth-signaali vaihtelee lähetyskanavaa 1 600 kertaa sekunnissa. WLAN käyttää samaa kanavaa yhdellä kerralla pidemmän aikaa, joten päällekkäisyyksiä tapahtuu silti. Tätä varten taajuushyppelyä on paranneltu sopeutuvalla taajuushyppelyllä, AFH, joka ennen lähetystä tarkistaa, onko valitulla kanavalla muuta aktiivista liikennettä. (Intelligent Hospital Today 2013)

3 BLUETOOTH SMART

Bluetooth Smart (toiselta nimeltään Bluetooth Low Energy) sai alkunsa vuonna 2006 Nokian kehittämänä Wibree-nimisenä langattomana tiedonsiirtotekniikkana. Vuonna 2010 se liittettiin osaksi Bluetooth 4.0 -standardia. Bluetooth Smartin tarkoituksena on tarjota Bluetoothin ominaisuudet pienemmällä virrankulutuksella laitteille, jotka eivät vaadi suuria datansiirtomääriä tai jatkuvaa datansiirtoa. Ensimmäinen Smart-teknologiaa tukeva laite oli Apple iPhone 4S, joka julkaistiin lokakuussa 2011. (Wikipedia 2015)

Bluetooth Smart -laitteet jaetaan kahteen ryhmään: Bluetooth Smart ja Bluetooth Smart Ready. Smart Ready -laitteet ovat isäntälaitteita, kuten älypuhelimia, kannettavia tietokoneita tai kämmentietokoneita. Smart-teknologian lisäksi ne tukevat myös vanhempia Bluetooth-laitteita. (Laptop Mag 2012)

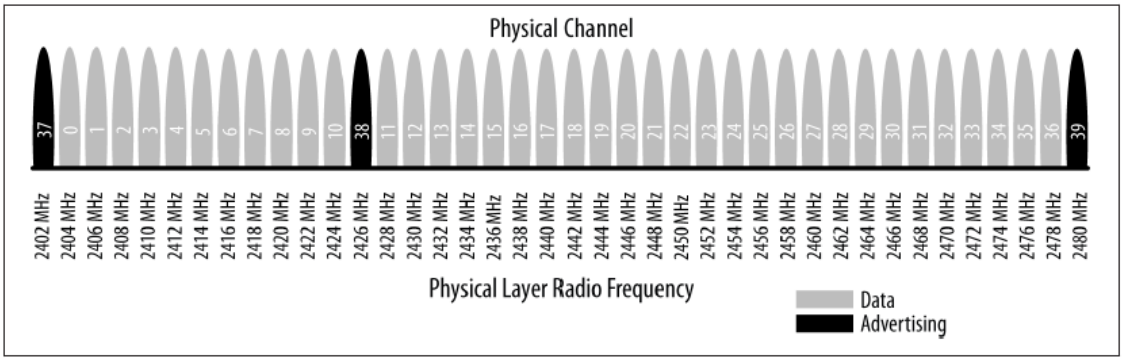
Smart-laitteet ovat lisälaitteita, jotka voidaan yhdistää ainoastaan Smart Ready -isäntälaitteisiin. Tyypillisimpiä käyttökohteita ovat esimerkiksi kuntoilun ja terveydenhallinnan lisälaitteet, kuten sydänmonitorit, kotitalouksien valaistuksenhallintajärjestelmät sekä älypuhelimiin liitettävät älykellot. (Laptop Mag 2012)

3.1 Erot Bluetoothin vanhempiin versioihin

Merkittävämpänä erona vanhaan Bluetoothin voidaan pitää alhaista virrankulutusta, joka voi parhaimmillaan olla vain kymmenesosa vanhan Bluetoothin käyttämästä virrasta. Syy tähän on se, että Smart-laitteet pysyvät suurimman osan ajasta horrostilassa ja heräävät ainoastaan datan lähettämistä tai vastaanottamista varten. Pienen virrankulutuksen vuoksi laitteet tulevat toimeen pienemmillä paristoilla, jotka koostaan huolimatta tarjoavat joissan tapauksissa jopa monen vuoden käyttöiän. (LitePoint Corporation 2012)

Pieni virrankulutus myös rajoittaa Smartin ominaisuuksia joissakin tapauksissa. Esimerkiksi datansiirtonopeus ei vielä tällä hetkellä yllä tavallisen Bluetoothin tasolle, jonka nopeus voi joissain tapauksissa olla moninkertainen Smartiin verrattuna. Tämän takia jatkuvaa yhteyttä sekä suurta kaistanleveyttä käyttävät laitteet, kuten esimerkiksi langattomat kaiuttimet ja kuulokkeet, eivät sovellu käytettäväksi Bluetooth Smart -yhteydellä. (LitePoint Corporation 2012)

Vanhemmista versioista poiketen Smart käyttää ISM-taajudella 40 kaistanleveydeltään 2 MHz:n kanavaa, joista 3 on varattu mainostusliikenteelle (kuva 2). Vanhemmat Bluetooth-versiot käyttävät mainostukseen 32 kanavaa, joiden skannaamiseen saattaa kulua jopa yli 10 kertaa enemmän aikaa kuin Smart-laitteella. (LitePoint Corporation 2012)



Kuva 2. Bluetooth-smart kanavat. (Townsend ym. 2014, 17)

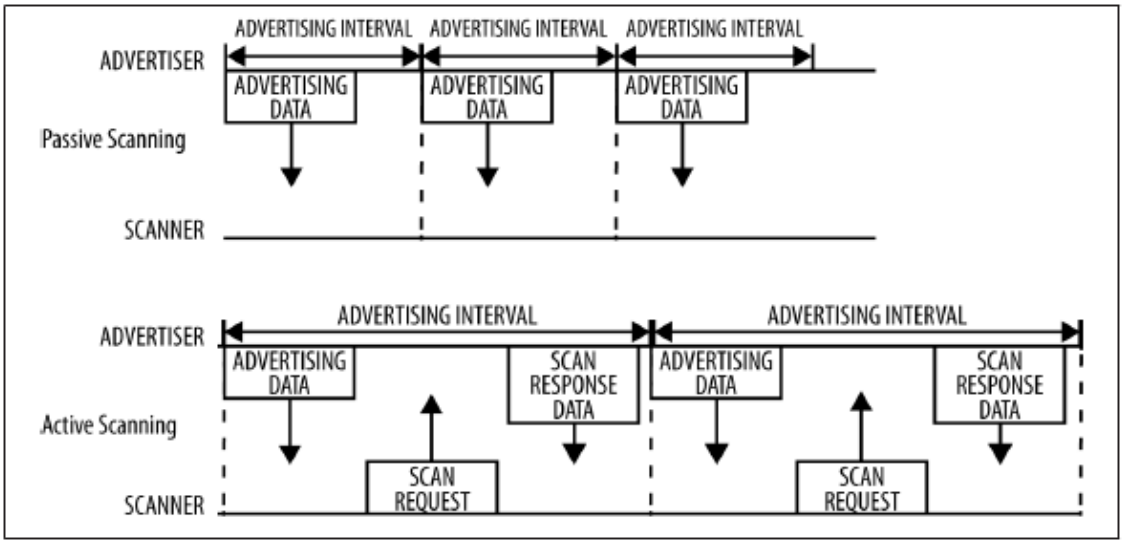
3.2 GAP

Generic Access Profile (GAP) on Bluetooth Smartin kulmakivi, joka mahdollistaa laitteiden välisen toiminnan. Se määrittelee tavat, joiden mukaan laitteet mainostavat itseään, löytävät toisensa ja lähettävät dataa, sekä tarpeen vaatiessa rajoittaa laitteen näkyvyyttä. (Adafruit Learning System 2015)

Laitteen mainostaminen voi tapahtua kahdella tavalla: joko suoralla mainostuksella tai vastauksella mainostuspyyntöön. Molemmat mainostustavat sisältävät pakollisen 31 tavun paketin, joka sisältää vähimmäismäärän tietoa

mainostavasta laitteesta. Mainostusintervalli voidaan asettaa välille 20 ms - 10,24 s. (Townsend ym. 2014, 19)

OIlessaan passiviisessa skannausmoodissa skannaava laite yksinkertaisesti vain kuuntelee saapuvia mainospaketteja ilman, että mainostava laite saa minkäänlaista tietoa sitä kuuntelevista laitteista. Toinen tapa, vastaus mainostuspyyntöön, mahdollistaa suuremman datamäärän lähetyksen. Tällöin pakettiin voidaan lisätä esimerkiksi laitteen tekstimuotoinen nimi. Aktiivisessa skannausmoodissa oleva laite voi tarvittaessa pyytää tätä pakettia mainostavalta laitteelta. Kuva 3 esittelee molemmat mainostustavat. (Townsend ym. 2014, 19–20)



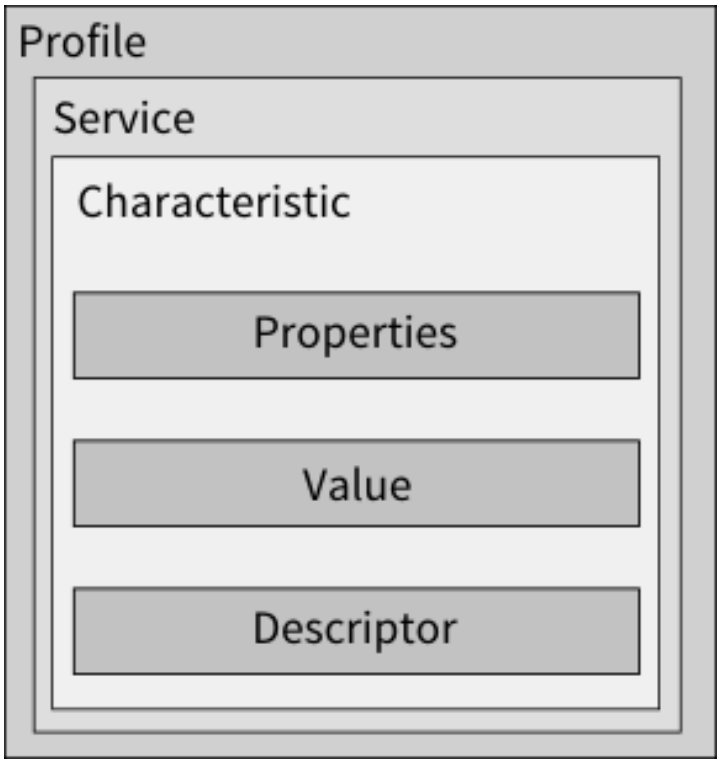
Kuva 3. Bluetooth Smartin mainostustavat. (Townsend ym. 2014, 21)

3.3 GATT

Generic Attribute Profile (GATT) on standardi, joka määrittelee kahden Bluetooth Smart -laitteen välisen liikenteen sen jälkeen, kun ne on yhdistetty toisiinsa. Se käyttää viestien lähettämiseen ATT-protokollaa (Attribute Protocol), jota käytetään laitteiden käyttämän datan säilyttämiseen ja hakemiseen. (Adafruit Learning System 2015)

GATT jakaa Bluetooth Smart -laitteet kahteen ryhmään: palvelimiin (server) ja asiakkaisiin (client). Käytännössä palvelin on aina oheislaite, kuten esimerkiksi sydänmonitori. Asiakas, kuten esimerkiksi tietokone tai älypuhelin, lähettää palvelimelle komentoja, joilla voidaan lukea tai kirjoittaa palvelimella olevaa dataa. Palvelin vastaa kutsuihin ja lähettää tarpeen vaatiessa ilmoituksen, kun datassa tapahtuu muutoksia. (Bluetooth Developer Portal 2015)

Palvelimella oleva data koostuu profiileista (profile), palveluista (services) ja karaktereista (characteristic), jotka muodostavat kuvan 4 mukaisen hierarkian. (Bluetooth Developer Portal 2015)



Kuva 4. GATT-profiili.

Päällimmäisenä hierarkiassa on profiili, joka on rakenteellinen määrittely siitä, miten oheislaitteen tarjoama data näkyy muille laitteille. Profiili on jaettu palveluihin, jotka kukin kuvaavat jotakin laitteen funktiota. (Bluetooth Developer Portal 2015)

Palvelu on kokoelma ominaisuuksia, jotka kaikki ovat yhteydessä jonkin tietyn toiminnon toteuttamiseen (Bluetooth Developer Portal 2015). Näitä ovat

esimerkiksi tässä työssä käytetyn SensorTagin sensorit. Palvelut on yksilöity UUID-tunnisteella (Universally Unique Identifier), joka on palvelusta riippuen joko 16- tai 128-bittinen (Adafruit Learning System 2015).

Jokaisella palvelulla on tietty määrä karaktereita, jotka pitävät sisällään konkreettista luettavissa tai kirjoitettavissa olevaa dataa. Tavallisesti yksi karakteri sisältää ominaisuuksia (esim. asetukset, onko jokin palvelu päällä vai ei), data-arvoja (esim. jonkin sensorin arvo sillä hetkellä) ja kuvauksia (descriptor), jotka ovat tekstimuotoisia kuvauksia ominaisuuksista tai data-arvoista. Karaktereita voidaan joissain tapauksissa käyttää myös asetusten määrittelyyn. (NewCircle Inc. 2013)

3.4 Bluetooth Smart tulevaisuudessa

Version 4.2 myötä Bluetooth otti entistä suuremman harppauksen kohti universaalimpaa yhdistettävyyttä teollisessa internetissä (engl. The Internet of Things, IoT). Suurimpana edistysaskeleena voidaan pitää Internet Protocol Support Profilea, joka tulee mahdollistamaan IPv6-tuen Bluetooth-laitteissa. Käytännössä tämä tarkoittaa sitä, että laitteet voivat tulevaisuudessa olla (reitittimien kautta) itsenäisesti yhteydessä pilvipalveluihin ilman välissä olevaa isäntälaitetta, kuten esimerkiksi älypuhelinta. Suuremmalla aikavälillä Bluetooth Smartista voidaankin odottaa kasvavan vakavasti otettava pikkuveli hallitsevan WLANin rinnalle. (Bluetooth Technology Website 2014)

Suuria parannuksia on odotettavissa myös Bluetoothin yleiseen suorituskykyyn. Versio 4.2 toi mukanaan entistä nopeamman sekä luotettavamman datansiirron kahden laitteen välillä kasvattamalla datapakettien kapasiteettia. Tämän ansiosta mm. virrankulutus ja lähetyksen aikana tapahtuvien virheiden määrä pienenee. (EE Times 2014)

Bluetoothin laajentuessa kohti IoT:a myös riski altistua perinteisille verkkohyökkäyksille lisää paineita pitää tietoturva ajan tasalla, varsinkin käyttäjien saadessa enemmän valtaa päättää laitteidensa näkyvyydestä langattomissa verkoissa. Uutena ominaisuutena Smart-laitteen voi asettaa

vaihtamaan osoitettaan tasaisin väliajoin estäen entuudestaan tuntemattomien laitteiden yhteydenotot. Tämäkin vaikuttaa suoraan suurempaan virransäästöön, sillä se vähentää merkittävästi laitteen hereilläoloaikaa. (EE Times 2014)

4 KEHITYSTYÖKALUT

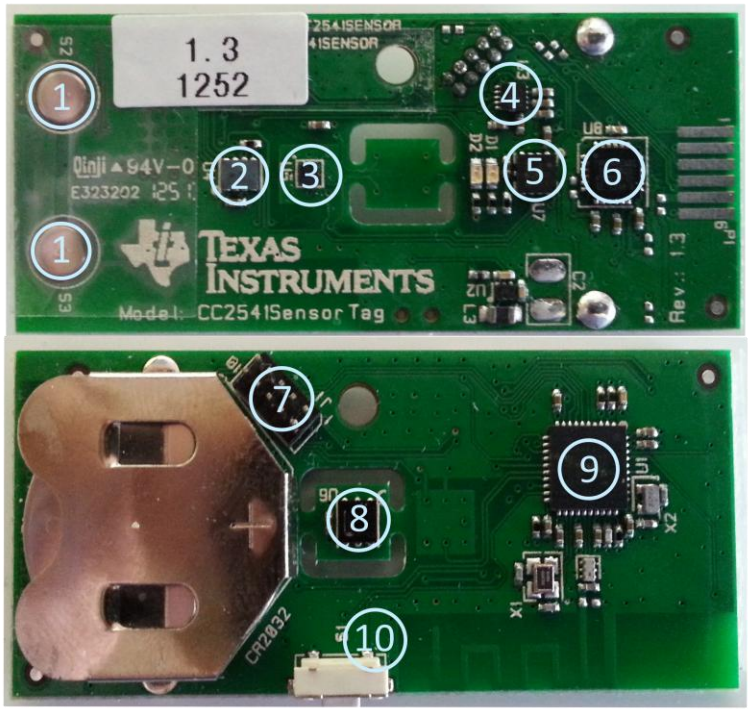
4.1 TI CC2541 Sensortag -kehitystyökalu

CC2541 SensorTag Development Kit (kuva 5) on Texas Instrumentsin valmistama kehitystyökalu, jonka tarkoituksena on esitellä TI:n CC2541-mikropiiriä sekä tarjota sovelluskehittäjille kokonaisvaltainen ratkaisu Bluetooth Smartin ominaisuuksien demonstroimiseen. Siinä on kuusi erilaista sensoria, jotka kaikki ovat erikseen aktivoitavissa. Virrankulutuksen minimoimiseksi kaikki sensorit ovat oletuksena pois päältä ja ne pysyvät unitilassa silloin kun niitä ei lueta. Tästä syystä koko piirilevy toimii yhdellä CR2032-kolikkoparistolla. (Make Magazine 2013)

Kuva 5. TI SensorTag -kehitystyökalu.

SensorTagin ydin CC2541 on system-on-chip-piiri (SoC), eli siinä on MCS8051-mikrokontrollerin lisäksi sisäänrakennettuna mm. Bluetooth Smart -piiri, 2 USARTia, 23 digitaalista IO-pinniä sekä 12-bittinen ADC. CC2541:n ja

sensoreiden lisäksi SensorTagin piirilevyssä on kaksi yleispainiketta sekä parituspainike BLE:n mainostuksen aloittamista ja lopettamista varten. Kaikki komponentit on saatu mahtumaan kompaktille, noin 3 x 6 cm:n kokoiselle piirilevylle, joka esitellään kuvassa 6. (Texas Instruments Wiki 2015)



Kuva 6. SensorTagin pääkomponentit.

SensorTagin sensorit ovat seuraavat:

- **IR -lämpötilasensori TI TMP006** mittaa halutun kohteen lämpötilaa välimatkan päästä 1 °C:n tarkkuudella. Valmistajan mukaan suositeltu välimatka riippuu kohteen koosta ja sen tulisi olla maksimissaan kohteen säde/2. (Texas Instruments Wiki 2015)

- **Kosteussensori Sensirion SHT21** mittaa ilman suhteellista kosteutta ja lämpötilaa. Parhaan mittaustuloksen saavuttamiseksi piiri vaatii kosketuksen ympäröivän ilman kanssa, mikä on otettu huomioon SensorTagin kotelossa lisäämälle siihen ilmanottoaukot kosteussensoria varten. (Texas Instruments Wiki 2015)

- **Gyroskooppi Invensense IMU-3000** mittaa kallistuskulman muutosta kolmella akselilla (x, y, z) maksimissaan 16 b:n tarkkuudella. Gyroskoopin ja kiihtyvyysmittarin arvojen perusteella voidaan määritellä SensorTagin asento 3D-avaruudessa. (Texas Instruments Wiki 2015)

- **Kiihtyvyysmittari Kionix KXTJ9:n** avulla voidaan määritellä laitteen kiihtyvyys kolmella akselilla maksimissaan 14-bitin tarkkudella. Tämän lisäksi kiihtyvyysmittari havaitsee myös painovoiman suunnan. (Texas Instruments Wiki 2015)

- **Magnetometeri Freescale MAG3110** mittaa magneettikenttiä kolmella akselilla. Sillä voidaan mitata joko elektronisten laitteiden tai maapallon magnettikenttiä, jolloin sitä voidaan käyttää esimerkiksi kompassina yhdessä kiihtyvyysmittarin kanssa. (Texas Instruments Wiki 2015)

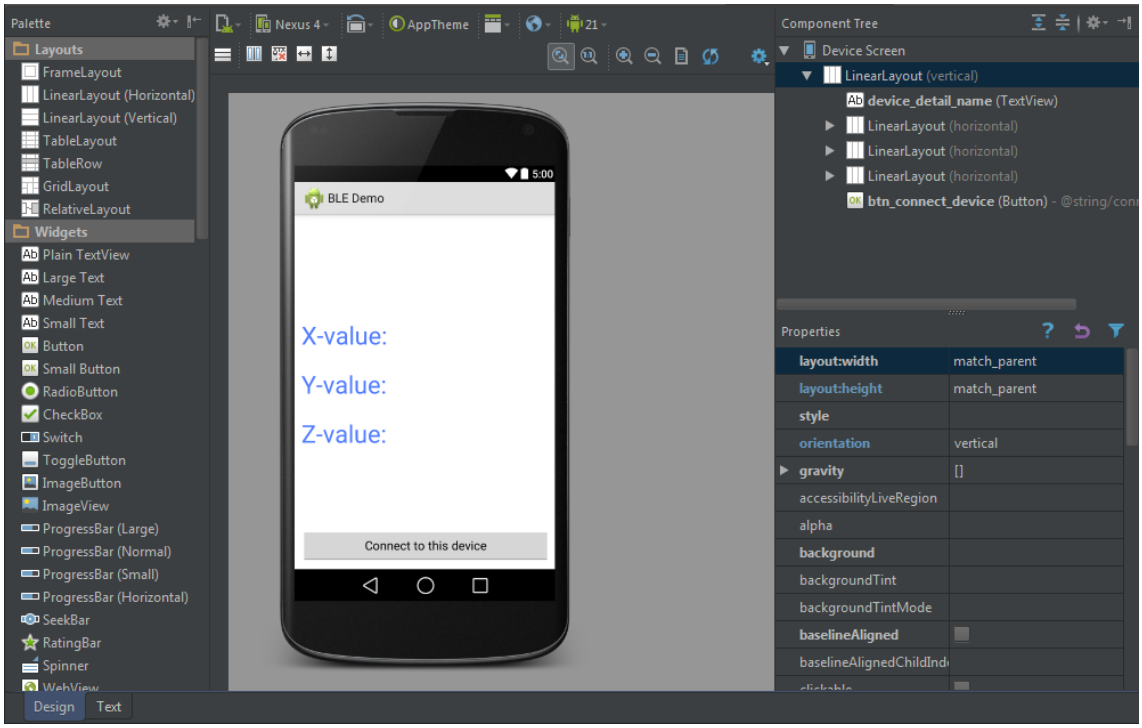
- **Painesensori Epcos T5400:lla** voidaan mitata barometrista ilmanpainetta. Tarkan mittaustulokden saavuttamiseksi kosteussensorin tavoin myös painesensori mittaa ilman lämpötilaa. (Texas Instruments Wiki 2015)

4.2 Android Studio

Android Studio on virallinen Android-sovelluskehitykseen tarkoitettu IDE, joka perustuu JetBrainsin IntelliJ IDEA -kehitysympäristöön Java-ohjelmointikielelle. Ohjelmiston esikatseluversio julkaistiin ensimmäisen kerran toukokuussa 2013, minkä jälkeen kehitys eteni tasaisesti ja johti lopulta ensimmäinen vakaan version julkaisemiseen joulukuussa 2014 (Venturebeat 2014). Android Studio on vapaasti saatavilla sekä Windows, Max OS X että Linux -alustoille Apache License 2.0 lisenssin alla (Wikipedia 2015).

Android Studion edeltäjänä voidaan pitää Googlen Android Development Tools -kehitystyökaluja, joiden avulla on mahdollista kehittää natiiveja Android-sovelluksia Eclipse-kehitysympäristössä. ADT-pakettiin verrattuna Android Studio on käytettävyydeltään valtava harppaus eteenpäin. Yksi näkyvimmistä

eroista on WYSIWYG-editori (kuva 7), jolla Android-sovelluksiin on mahdollista suunnitella toimivia graafisia käyttöliittymiä useille erikokoisille näytöille. Android Studio myös käyttää ohjelmakoodin kääntämisessä Gradle-työkalua, joka varsinkin isommissa projekteissa helpottaa sovelluksen kehitystä ja testausta. (Wikipedia 2015)



Kuva 7. Android Studion WYSIWYG-editori.

Android Studio on erinomainen työkalu varsinkin testattaessa ohjelmakoodia fyysisellä laitteella. Ajon aikaiset tapahtumat on mahdollista kirjata virheiden jäljittäjän lokiin, joka päivittyy tietokoneen näytölle reaaliajassa. Samalla tavalla saadaan selville myös mahdolliset virheilmoitukset.

5 BLUETOOTH SMART ANDROID-SOVELLUKSESSA

5.1 Valmistelut

Bluetooth Smart -tuki vaatii Android-laitteissa vähintään 4.3-version, mikä tulee ottaa huomioon sovelluksen määrityksiä tehtäessä. Android Studiossa sovelluksen minimi- ja kohdeversiota voi muuttaa build.gradle-tiedostossa määrittelemällä kutakin versiota vastaavan API-tason. Smart-ominaisuutta tukevan sovelluksen API-tasoksi tulee siis asettaa vähintään 18.

Minimiversion lisäksi sovellukselle tulee antaa lupa käyttää laitteen Bluetooth-radiota, mikä tehdään sovelluksen AndroidManifest.xml-tiedostossa. Luvat määritellään uses-permission-tagilla, jotka sijoitetaan tiedostossa manifest-tagin sisään. Täyttä Bluetooth-tukea varten tiedostoon tulee määritellä kaksi eri lupaa, BLUETOOTH ja BLUETOOTH_ADMIN, seuraavasti:

Koodi 1. Bluetooth-luvat.

```
<uses-permission
    android:name="android.permission.BLUETOOTH" />
<uses-permission
    android:name="android.permission.BLUETOOTH_ADMIN" />
```

Tämä varmistaa sen, että asentaessaan esim. Google Play -kaupasta käyttäjän tulee hyväksyä sovelluksen vaatimat ominaisuudet laitteessa.

Laitteen Bluetooth Smart -tuki tulee vielä varmistaa erikseen uses-feature-tagilla:

Koodi 2. Smart-tuen tarkastaminen.

```
<uses-feature
    android:name="android.hardware.bluetooth_le"
    android:required="true" />
```

5.2 Bluetoothin käyttöönotto

Kaikki sovelluksen ja laitteen Bluetooth-radion välinen interaktio vaatii toimiakseen adapterin, joka on uniikki koko käyttöjärjestelmässä. Tässä tapauksessa adapteri voidaan hakea BluetoothManager-luokan kautta sijoittamalla seuraava esimerkki halutun Activity-luokan onCreate-metodin sisälle:

Koodi 3. Bluetooth adapterin hakeminen.

```
BluetoothManager bluetoothManager =
    (BluetoothManager) getSystemService(
                       Activity.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

Tällä tavalla adapterin alustus tapahtuu heti sovelluksen käynnistyessä.

Sovelluksen toimivuuden kannalta on myös tärkeä tarkistaa, onko käyttäjä asettanut laitteen Bluetoothin päälle. Jos näin ei ole, voidaan käyttäjälle lähettää pyyntö ominaisuuden käynnistämiseksi. Tämä voidaan tehdä esimerkiksi onResume-metodin sisällä, jolloin tarkistus tapahtuu aina sovelluksen aktivoituessa.

Koodi 4. Bluetoothin käynnistäminen.

```
if (bluetoothAdapter == null || !bluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new
        Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivity(enableBtIntent);
}
```

5.3 Smart-laitteiden skannaus

Kun kaikki edellämainitut esivalmistelut on tehty, voidaan aloittaa lukuetäisyydellä olevien Smart-laitteiden etsiminen. Koska skannaus kuluttaa

huomattavasti akkua, tulisi sitä rajoittaa esimerkiksi ajallisesti. Jos haettavan laitteen UUID on tiedossa, voidaan skannaus myös lopettaa heti halutun laitteen löytyessä. SensorTagin mainostus tulee asettaa päälle sen kyljessä olevaa parituspainiketta painamalla.

Edellä mainittu bluetoothAdapter-olio sisältää metodit skannauksen aloittamiseen ja lopettamiseen. Seuraavassa esimerkissä sovellus skannaa lukuetäisyydellä olevat laitteet ja lopettaa automaattisesti ennalta määritellyn ajan kuluttua.

Koodi 5. Smart-laitteiden skannaus.

```
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        bluetoothAdapter.stopLeScan(leScanCallback);
    }
}, SCAN_PERIOD);
bluetoothAdapter.startLeScan(leScanCallback);
```

SCAN_PERIOD on long-muotoinen kokonaislukuvakio, jolle annetaan haluttu aika-arvo millisekunneissa. Skannaus löytää melko nopeasti lähettyvillä olevat laitteet, joten sen tulisi olla kerrallaan päällä maksimissaan 10 s.

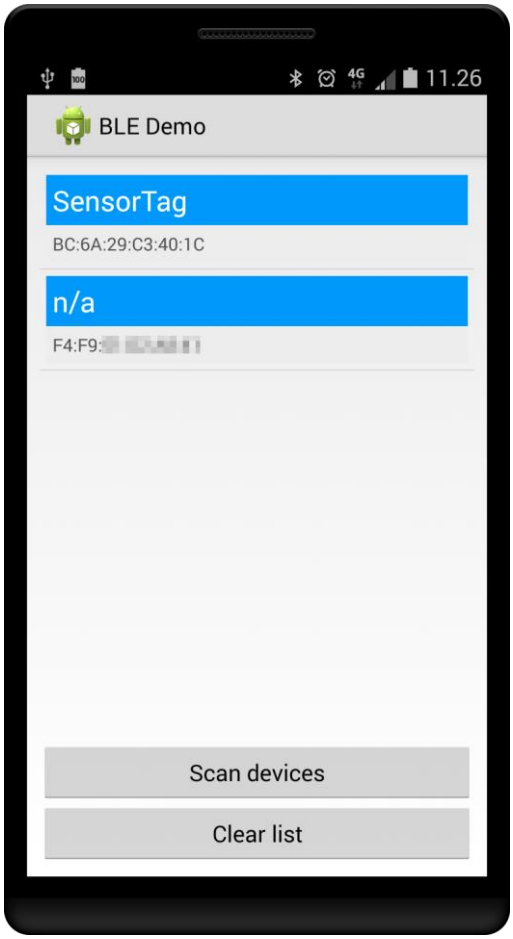
Molemmissa metodeissa käytetään parametrina BluetoothAdapter-luokkaan kuuluvan LeScanCallback-rajapintaa, jossa tehdään halutut toimenpiteet jokaisen laitteen löytyessä. Tämä tapahtuu määrittelemällä callback-metodi onLeScan, joka saa kutsuttaessa parametreikseen laitteen kuvauksen, signaalin voimakkuuden eli RSSI-arvon sekä laitteen mainosdatan.

Koodi 6. onLeScan-metodin määrittely.

```
private BluetoothAdapter.LeScanCallback leScanCallback = new
        BluetoothAdapter.LeScanCallback() {
    @Override
    public void onLeScan(final BluetoothDevice device, int rssi,
```

```
                        byte[] scanRecord) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (!bleDeviceList.contains(device)) {
                    bleDeviceList.add(device);
                    deviceArrayAdapter.add(device);
                    deviceArrayAdapter.notifyDataSetChanged();
...
```

Tässä esimerkissä löydetyt laitteet kerätään listaan, jonka avulla tarkistetaan, ettei samaa laitetta listata kahteen kertaan. Löydetyt laitteet esitetään näytöllä listView-objektina, johon laitteet lisätään DeviceAdapterin kautta (kuva 8). Tässä työssä DeviceAdapter on yksinkertainnen ArrayAdapter-luokka, jonka avulla voidaan muokata listan esitysasua näytöllä.



Kuva 8. Skannattujen laitteiden nimet ja osoitteet älypuhelimen näytöllä.

5.4 GATT-profiiliin yhdistäminen

Kun haluttu Smart-laite on löytynyt, voidaan yhteys muodostaa sen GATT-profiiliin. Tämä tapahtuu kutsumalla BluetoothDevice-olion connectGatt-metodia, joka onnistuessaan palauttaa kyseessä olevan laitteen GATT-profiiliin viittaavan BluetoothGatt-olion.

Koodi 7. GATT-profiiliin yhdistäminen.

```
bluetoothGatt = device.connectGatt(this, false,
    bluetoothGattCallback);
```

Metodille syötetään kolmanneksi parametriksi BluetoothGattCallback-instanssi, jossa määritellään callback-metodit, joita kutsutaan tiettyjen muutosten tapahtuessa GATT-profiilissa. Näitä ovat esimerkiksi muutokset yhteydessä tai karakterien arvoissa sekä luku- ja kirjoitusoperaatioiden päättymiset.

Koodi 8. BluetoothGattCallback-instanssin esimerkkimetodit.

```
private final BluetoothGattCallback bluetoothGattCallback = new
             BluetoothGattCallback() {
    @Override
    public void onConnectionStateChange(BluetoothGatt gatt,
              int status, int newState) {
        // Muutos yhteydessä
    }
    @Override
    public void onServicesDiscovered(BluetoothGatt gatt,
             int status) {
        // Palveluiden löytyminen
    }
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt,
             BluetoothGattCharacteristic characteristic) {
        // Muutos karakterin arvossa
...
```

5.5 Palvelun käyttöönotto

Kun yhteys laitteen GATT-profiilin on muodostettu, voidaan ryhtyä etsimään laitteessa tarjolla olevia palveluja. Ennen palveluiden varsinaista käyttöönottoa tulee kaikki palvelut hakea discoverServices-metodilla. Tämä voidaan tehdä heti GATT-yhteyden muodostuessa eli päivittämällä koodi 8:ssa esitetty onConnectionStateChange-metodi. On syytä huomata, että discoverServices-metodi ei itsessään palauta vielä yhtäkään palvelua.

Koodi 9. onConnectionStateChange-metodi.

```
@Override
public void onConnectionStateChange(BluetoothGatt gatt, int
            status, int newState) {
    super.onConnectionStateChange(gatt, status, newState);
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        bluetoothGatt.discoverServices();
    }
    else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
        if (bluetoothGatt != null) {
            bluetoothGatt.close();
            bluetoothGatt == null;
...
```

Onnistuessaan discoverServices-metodi aiheuttaa onServicesDiscovered-kutsun, jonka jälkeen halutun palvelun käyttöönotto on mahdollista.

Koodi 10. onServicesDiscovered-metodi.

```
@Override
public void onServicesDiscovered(BluetoothGatt gatt, int status)
{
    if (status == BluetoothGatt.GATT_SUCCESS) {
        enableSensor(ACCELEROMETER_SERVICE_UUID,
                    ACCELEROMETER_CONFIG);
...
```

Koodi 11 esittelee enableSensor-metodin, jota nimensä mukaisesti käytetään halutun palvelun eli tässä tapauksessa kiihtyvyysmittarin aktivoimisessa. Tätä toimenpidettä varten tulee ottaa ensin selville palvelun yksilöllinen Primary Service Declaration UUID sekä saman palvelun asetusten hex-arvo, jotka voidaan hakea Texas Instrumentsin tarjoamasta 'SensorTag Attribute Table'-dokumentista [23]. Kaikki kiihtyvyysmittarin tässä työssä käytetyt attribuutit löytyvät koodista 12.

Koodi 11. enableSensor-metodi.

```
public void enableSensor(String serviceUuid,
                         String configUuid) {
    BluetoothGattService service = bluetoothGatt
        .getService(UUID.fromString(serviceUuid));
    BluetoothGattCharacteristic characteristic =
        service.getCharacteristic(UUID.fromString(configUuid));
    characteristic.setValue(new byte[] {0x01});
    bluetoothGatt.writeCharacteristic(characteristic);
}
```

Koodi 12. Kiihtyvyysmittarin attribuuttivakiot.

```
public final static String ACCELEROMETER_SERVICE_UUID =
    "f000aa10-0451-4000-b000-000000000000";
public final static String ACCELEROMETER_DATA_UUID =
    "f000aa11-0451-4000-b000-000000000000";
public final static String ACCELEROMETER_CONFIG_DESCRIPTOR =
    "00002902-0000-1000-8000-00805f9b34fb";
public final static String ACCELEROMETER_CONFIG =
    "f000aa12-0451-4000-b000-000000000000";
```

Tässä metodissa haetaan ensin sensorin UUID:n avulla kiihtyvyysmittarin palvelu, jonka kautta voidaan hakea sensorin aktivoinnissa käytettävä karakteri. Karakterin arvo määrittelee sensorin mittaamien arvojen g-voiman vaihteluvälin, jonka arvoiksi voidaan asettaa joko ±2, ±4 tai ±8. Tässä tapauksessa valitaan ensimmäinen vaihtoehto, joten karakterin arvoksi kirjoitetaan hex-arvo 1.

Näiden toimenpiteiden jälkeen karakteri kirjoitetaan takaisin laitteelle, jolloin tehdyt muutokset astuvat voimaan.

5.6 Sensorin arvojen lukeminen

Kun sensori on aktiivinen, voidaan ryhtyä lukemaan sen keräämää dataa sensorin palvelun karakterien kautta. Lukeminen voidaan tehdä joko manuaalisesti tai automaattisten ilmoitusten avulla, kuten tässä työssä menetellään. Koodi 13 etsii GATT-profiilista halutun palvelun sekä sen sisältämän karakterin, josta sensorin antamat arvot noudetaan.

Koodi 13. Karakterin hakeminen.

```
List<BluetoothGattService> services =
        bluetoothGatt.getServices();
for (BluetoothGattService service : services) {
    if (ACCELEROMETER_SERVICE_UUID
            .equals(service.getUuid().toString())) {
        BluetoothGattCharacteristic characteristic =
                service.getCharacteristic(UUID.fromString(
                ACCELEROMETER_DATA_UUID));
        enableCharacteristicNotification(characteristic);
...
```

Oikean karakterin löydyttyä se syötetään parametriksi enableCharacteristicNotification-metodille, jossa hoidetaan tarvittavat toimenpiteet ilmoitusten aktivoimiseksi sensorin arvon automaattista lukemista varten. Tässä vaiheessa tulee huomioida, että toivottuja tuloksia varten ilmoitukset tulee aktivoida sekä Android- että Smart-laitteessa. Androidissa tämä onnistuu yksinkertaisesti BluetoothGatt-luokassa olevan metodin kautta. Smart-laitteen aktivointia varten valitusta karakterista haetaan SensorTagin attribuuttitaulukon (Texas Instruments Wiki 2015) mukaan oikean descriptor-rekisteri, johon kirjoitetaan oikea arvo ilmoitusten aktivoimista varten.

Koodi 14. enableCharacteristicNotification-metodi.

```
public void enableCharacteristicNotification(
        BluetoothGattCharacteristic characteristic) {
    bluetoothGatt.setCharacteristicNotification(characteristic,
        true);
    final BluetoothGattDescriptor descriptor =
        characteristic.getDescriptor(
        UUID.fromString(ACCELEROMETER_CONFIG_DESCRIPTOR));
    descriptor.setValue(
        BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
    descriptorWriteQueue.add(descriptor);
}
```

Kaikki datan kirjoittaminen karakteriin tapahtuu asynkronisesti, mikä saattaa joissain tapauksissa estää tapahtuman kokonaan, mikäli useampia kirjoitusoperaatioita on käynnissä samanaikaisesti. Tässä tapauksessa kävi ilmi, että koodissa 11 esitetty kirjoitusoperaatio esti koodissa 14 esitetyn descriptor-rekisterin kirjoituksen, jos se suoritettiin heti enableCharacteristicNotification-metodin sisällä. Tästä syystä descriptor-arvo laitetaan kirjoitusjonoon, josta se haetaan vasta kun edellinen operaatio on suoritettu, toisin sanoen onCharacteristicWrite-metodikutsun tapahtuessa.

Koodi 15. onCharacteristicWrite-metodi.

```
@Override
public void onCharacteristicWrite(BluetoothGatt gatt,
        BluetoothGattCharacteristic characteristic,
        int status) {
    if (descriptorWriteQueue.size() > 0)
        bluetoothGatt.writeDescriptor(
                descriptorWriteQueue.remove());
}
```

Kun ilmoitusten aktivoiminen on tehty, alkaa Smart-laite lähettää tasaisin väliajoin valittujen sensoreiden keräämää dataa. Kunkin arvon muuttuessa

tapahtuu onCharacteristicChanged-metodikutsu, jossa voidaan tehdä halutut toimenpiteet arvojen keräämistä ja esittämistä varten.

Koodi 16. onCharacteristicChanged-metodi.

```
@Override
public void onCharacteristicChanged(BluetoothGatt gatt,
        BluetoothGattCharacteristic characteristic) {
    int x = c.getIntValue(
            BluetoothGattCharacteristic.FORMAT_SINT8, 0);
    int y = c.getIntValue(
            BluetoothGattCharacteristic.FORMAT_SINT8, 1);
    int z = (c.getIntValue(
            BluetoothGattCharacteristic.FORMAT_SINT8, 2) * -1);
    int t[] = { x, y, z };
    displayData(t);
}
```

Kiihtyvyysmittarin palauttama data koostuu yhdestä kolmen tavun kokoisesta data-arvosta, joka voidaan jakaa kolmeen osaan. Kukin osa pitää sisällään yhden kiihtyvyysmittarin akselin arvoista, joiden yksikkö on (1/64)g. Z-akselin arvo kerrotaan luvulla -1, jotta sen positiivinen suunta on oikea suhteessa kahteen muuhun akseliin.

Arvot haetaan tässä tapauksessa karakterin getIntValue-metodilla, jolle annetaan parametreiksi haluttu kokonaislukumuuttujatyyppi sekä data-arvon indeksi, josta kunkin akselin arvo löytyy. Nyt data voidaan esittää älpyhelimen näytöllä, kuten esimerkiksi koodissa 17.

Koodi 17. displayData-metodi.

```
public void displayData(int[] data) {
    accValueX.setText(String.valueOf(data[0]));
    accValueY.setText(String.valueOf(data[1]));
    accValueZ.setText(String.valueOf(data[2]));
}
```

Jokaisen akselin arvo asetetaan kuvan 9 mukaisesti yhden näytöllä näkyvän TextView-objektin tekstikentän arvoksi, joka päivittyy tasaisin väliajoin sensoria luettaessa. Kuvan 9 arvot vastaavat tilannetta, jossa SensoTag makaa selällään paikoillaan vaakasuoralla alustalla. Vastaavalla tavalla myös muut sensorit voidaan ottaa käyttöön lukemista ja tulosten käsittelyä varten.

Kuva 9. Kiihtyvyysmittarin lähettämiä arvoja älypuhelimen näytöllä.

6 LOPUKSI

Lopputuloksena syntyi varsin onnistunut Android-sovellusmalli, joka pienellä jatkojalostuksella sopisi käytettäväksi pohjana laajemman mittakaavan projekteissa. Työn aikana tosin huomattiin, että Android-käyttöjärjestelmän versiossa 5 myös Bluetooth Smart API on uudistettu siinä määrin, että tätä versiota varten tulisi lähes kaikki Smart-yhteyteen liittyvä ohjelmakoodi muokata uutta APIa vastaavaksi.

Projekti opetti paljon varsinkin Bluetoothista, mutta myös yleisesti Android-sovelluskehityksestä, joka oli entuudestaan tuttua lähinnä pienistä harjoitustöistä. Positiiviseksi yllätykseksi osoittautui varsinkin kehitysympäristönä käytetyn Android Studion virheenjäljitystyökalu, joka helpotti sovelluskehitystä huomattavasti.

Kehitystyökaluna myös SensorTag havaittiin yllättävänkin toimivaksi tavaksi tutustua Bluetooth Smartin ominaisuuksiin. Texas Instrumentsin tarjoamat dokumentaatiot osoittautuivat alun ihmettelyjen jälkeen hyvinkin kattaviksi.

Kaiken kaikkiaan työn aihe oli mielenkiintoinen mutta sopivan haastava. Projektin aikataulu venyi lopussa hieman, mutta lopulta tavoitteisiin kuitenkin päästiin.

LÄHTEET

Wareable. Android Wear: The essential guide. Viitattu 22.5.2015 http://www.wareable.com/android-wear/what-is-android-wear-comprehensive-guide.

Wikipedia. Google Glass. Viitattu 22.5.2015 http://en.wikipedia.org/wiki/Google_Glass.

Bluetooth Technology Website. Fast Facts. Viitattu 2.2.2015 http://www.bluetooth.com/Pages/Fast-Facts.aspx.

Flicksoftware. bluetooth-logo. Viitattu 25.5.2015 http://flicksoftware.com/flick-technologies/bluetooth-peripherals/bluetooth-logo/.

Radio-Electronics. What is Bluetooth. Viitattu 2.2.2015 http://www.radio-electronics.com/info/wireless/bluetooth/bluetooth_overview.php.

Bluetooth Technology Special Interest Group. About the Bluetooth SIG. Viitattu 2.4.2015 https://www.bluetooth.org/en-us/members/about-sig.

Bluetooth Technology Website. Our History. Viitattu 6.3.2015 http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx.

Wikipedia. Bluetooth. Viitattu 6.3.2015 http://en.wikipedia.org/wiki/Bluetooth.

Radio-Electronics. Bluetooth radio interface, modulation, & channels. Viitattu 9.3.2015 http://www.radio-electronics.com/info/wireless/bluetooth/radio-interface-modulation.php.

Hewlett-Packard. Wi-Fi and Bluetooth - Interference Issues. Viitattu 9.3.2015 http://www.hp.com/rnd/library/pdf/WiFi_Bluetooth_coexistance.pdf.

Intelligent Hospital Today. Can Bluetooth and 802.11b/g/n Wi-Fi Devices Coexist? Viitattu 9.3.2015 http://intelligenthospitaltoday.com/can-bluetooth-and-802-11bgn-wi-fi-devices-coexist/.

Wikipedia. Bluetooth low energy. Viitattu 11.3.2015 http://en.wikipedia.org/wiki/Bluetooth_low_energy.

Laptop Mag. What is Bluetooth 4.0? Viitattu 13.3.2015 http://blog.laptopmag.com/just-what-is-bluetooth-4-0-anyway.

LitePoint Corporation. Bluetooth Low Energy. Viitattu 13.3.2015 http://www.litepoint.com/wp-content/uploads/2014/02/Bluetooth-Low-Energy_WhitePaper.pdf .

Townsend, K.; Cufí, C.; Davidson, A. & Davidson, R. 2014 Getting Started with Bluetooth Low Energy, First Edition. O'Reilly Media Inc..

Adafruit Learning System. GAP | Introduction to Bluetooth Low Energy. Viitattu 17.3.2015 https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap.

Adafruit Learning System. GATT | Introduction to Bluetooth Low Energy. Viitattu 17.3.2015 https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt.

Bluetooth Developer Portal. GATT. Viitattu 17.3.2015 https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx.

NewCircle Inc. Tutorial: Intro to Developing Bluetooth Smart Applications for Android. Viitattu 18.3.2015 https://thenewcircle.com/s/post/1553/bluetooth_smart_le_android_tutorial.

Bluetooth Technology Website. New Bluetooth® Specifications Enable IP Connectivity and Deliver Industry-leading Privacy and Increased Speed. Viitattu 22.4.2015 http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=220.

EE Times. Bluetooth 4.2 Unveiled: No Mesh Yet, But Big on IoT. Viitattu 22.4.2015 http://www.eetimes.com/document.asp?doc_id=1324835.

Make Magazine. Teardown of the TI SensorTag. Viitattu 18.3.2015 http://makezine.com/2013/04/18/teardown-of-the-ti-sensortag/.

Texas Instruments Wiki. SensorTag User Guide. Viitattu 18.3.2015 http://processors.wiki.ti.com/index.php/SensorTag_User_Guide.

Venturebeat. Google releases Android Studio 1.0, the first stable version of its IDE. Viitattu 22.4.2015 http://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/.

Wikipedia. Android Studio. Viitattu 22.4.2015 http://en.wikipedia.org/wiki/Android_Studio.

Texas Instruments Wiki. SensorTag attribute table. Viitattu 18.3.2015 http://processors.wiki.ti.com/images/a/a8/BLE_SensorTag_GATT_Server.pdf.