VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Otto Karinen

BEACONIEN HYÖDYNTÄMINEN IOT-JÄRJESTELMÄSSÄ

Tekniikka
2016

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Otto Karinen
Opinnäytetyön nimi	Beaconien hyödyntäminen IoT-järjestelmässä
Vuosi	2016
Kieli	suomi
Sivumäärä	40
Ohjaaja	Pirjo Prosi

Opinnäytetyö toteutettiin Devatus Oy:lle. Työn tarkoituksena oli tutkia ja toteuttaa Bluetooth low energy -beaconin integraatio jo olemassa olevaan maintBox-järjestelmään. Beacon välittää siihen kytketyiltä antureilta luettua dataa iBeacon-protokollan avulla Android-laitteelle, josta data välitetään edelleen REST-rajapinnan kautta maintBox-järjestelmään. Koska mittausdata välitetään Android-laitteelle vähävirtaisen Bluetooth low energy -teknologian avulla, voidaan antureita lukevan laitteen virrankulutus saada niin pieneksi, että yksi paristo kestää koko sen eliniän.

Opinnäytetyössä mittalaitteena käytettiin DHT11 kosteus- ja lämpötila-anturia Raspberry Pi 3 -tietokoneen GPIO-pinnien kautta. Mittausdatan lukeminen ja eteenpäin lähettäminen toteutettiin C-ohjelmointikielellä Linux-ympäristössä. Natiivin Android-sovelluksen ja REST-rajapinnan toteuttamiseen käytettiin Java-ohjelmointikieltä ja Jersey-sovelluskehystä.

Opinnäytetyön tuloksena saatiin kustannustehokas ja hyvin skaalautuva IoT-järjestelmä, joka soveltuu pienellä muokkauksella moneen eri käyttöön.

Avainsanat Beacon, iBeacon, Android, REST-rajapinta, IoT

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Otto Karinen
Title	Benefiting from Beacons in IoT System
Year	2016
Language	Finnish
Pages	40
Name of Supervisor	Pirjo Prosi

This thesis was done for Devatus Ltd. The main objective of the thesis was to research and implement Bluetooth low energy beacon integration to the existing maintBox system. Beacon transmits data from the measuring device to the Android device using iBeacon protocol which forwards the data to the maintBox system through the REST API. Because the measurement data is transmitted to the Android device with Bluetooth low energy technology, the power consumption of the measuring device can be reduced to so low that it can run its lifetime with only single battery.

In this thesis humidity and temperature measurements were measured with DHT11 humidity and temperature sensor directly connected to the GPIO pins of Raspberry Pi 3 computer. Reading and transmitting measurement data was implemented with C programming language in a Linux environment. Native Android application and REST API were implemented with Java programming language and Jersey-framework.

The result of this thesis was a cost-effective and a well scalable IoT system which can be used in many different cases with small modifications.

Keywords Beacon, iBeacon, Android, REST API, IoT

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LYHENTEET JA TERMIT

BLE Bluetooth Low Energy, vähävirtainen lyhyen kantaman langaton tiedonsiirtotekniikka.

GPIO General-Purpose Input/Output, yleiskäyttöinen portti elektronisen signaalin lähettämiseen ja/tai vastaanottamiseen.

Jersey Avoimen lähdekoodin sovelluskehys REST Web-palvelun toteuttamiseen.

API Application Programming Interface, määritelmä, jonka mukaan ohjelmat voivat keskustella keskenään.

CMMS Computerized Maintenance Management System, yrityksille suunnattu järjestelmä tuotantolaitosten tarkkailuun.

ADT Android Development Tools, Eclipse-kehitysympäristön liitännäinen Android-sovellusen kehittämiseen.

HAL Hardware Abstraction Layer, Android-käyttöjärjestelmän rajapinta, jonka avulla sovellukset voivat suoraan kommunikoida laitteiston kanssa.

SSL Secure Sockets Layer, tietoliikenteen salausprotokolla.

ART Android Runtime, Dalvik-virtuaalikoneen suorituskykyisempi korvaaja Android 5.0 (Lollipop) -versiosta eteenpäin.

AOT Ahead of Time, ART-virtuaalikoneen ominaisuus, joka nopeuttaa sovellusten käynnistysaikaa ja parantaa muistinkäyttöä.

HTTP Hypertext Transfer Protocol, tiedonsiirtoprotokolla.

SOAP Simple Object Access Protocol, tietoliikenneprotokolla, jonka tehtävänä on mahdollistaa web-palvelujen toimintojen etäkutsut.

GPL General Public License, lisenssi vapaiden ohjelmistojen julkaisemiseen.

XML Extensible Markup Language, merkintäkieli datan säilytykseen, siirtämiseen ja esittämiseen.

MAC Media Access Control, sovittimen yksilöivä tunniste.

RSSI Received Signal Strength Indication, signaalin vastaanottoteho.

1 JOHDANTO

IoT-järjestelmät ovat olleet jo jonkin aikaa erittäin kovassa nousussa, eikä tilanne näytä tasaantuvan. Järjestelmät tuovat monia hyötyjä jokapäiväiseen elämään ja niiden käyttömahdollisuudet ovat lähes rajattomat. Järjestelmillä voidaan ehkäistä myös monia vaaratilanteita, kuten maatilan heinäpaalien ”itsestäänsyttymisiä” niiden kosteutta ja lämpötilaa tarkkailemalla. IoT-järjestelmien määrä kasvaa jatkuvasti, joten aihe on erittäin ajankohtainen järjestelmien kysyntää ajatellen.

1.1 IoT käsitteenä

IoT (Internet of Things), eli esineiden internet, tarkoittaa esineen tai asian ympäristöstä kerätyn datan jatkohyödyntämistä, esimerkiksi jonkin tuotteen valmistusprosessin optimoinnissa. Jotta esineestä tai asiasta saataisiin älykäs, tarvitaan siihen ympäristöä aistivia antureita, ohjelmistoja ja jokin tietoliikenneyhteys, jolla se välittää keräämäänsä dataa eteenpäin. IoT-järjestelmä on kustannustehokas ratkaisu ongelmaan kuin ongelmaan ja uusi data on aina järkevästi hyödynnettävissä.

1.2 Opinnäytetyön tarkoitus

Opinnäytetyön toimeksiantajana toimi Devatus Oy. Työn tarkoituksena oli tutkia ja toteuttaa Bluetooth low energy -teknologiaa hyödyntävän beaconin integraatio jo olemassa olevaan maintBox-järjestelmään. Beacon on pientä datamäärää ennaltamääritellyin väliajoin ympärilleen lähettävä laite, jonka dataa voidaan lukea riittävällä etäisyydellä olevalla Bluetooth low energy -tuellisella laitteella, kuten nykyaikaisella älypuhelimella.

Bluetooth low energy -teknologian käyttö datan eteenpäin välittämiseen, on vähävirtainen ja antureita lukevan laitteen valmistuskustannusten kannalta edullinen tapa rakentaa monipuolinen IoT-järjestelmä, jonka kaikille osille verkkovirta ei ole välttämättömyys. Matalien valmistuskustannusten ja verkkovirrattomuuden ansiosta, antureita voitaisiin sijoittaa täysin uusiin kohteisiin kilpailukykyiseen hintaan.

1.3 Devatus Oy

Devatus Oy on vuonna 2010 Vaasassa perustettu ohjelmistoyritys, jonka henkilöstön vahva ohjelmisto-osaaminen ja kokemus takaavat loppuasiakkaan tyytyväisyyden. Yritys toteuttaa kokonaisia ohjelmistoprojekteja, mobiilisovelluksia ja projektien osia, sekä toimii tarvittaessa työvoimana asiakkaiden projekteissa. Yrityksen henkilöstömäärä on tällä hetkellä alle 20 ja se on täysin työntekijöiden omistuksessa. Sen asiakkaita ovat muun muassa Anvia ja Vacon. /13/

1.4 maintBox

maintBox on vuonna 2013 Tähtipiste ja Devatus Oy:n toimesta kehitetty tuotantolaitosten tehokkuuden, sekä palveluntoimittajan ja asiakkaan yhteistyön parantamiseen suunniteltu CMMS-työkalu. /14/

maintBox sisältää kaikki tuotannon hallinnointiin tarvittavat toiminnot, sekä tarjoaa täysin uudenlaisen tavan laitosten tapahtumien selkeään esittämiseen sijoittamalla tapahtumatiedot tuttuun ympäristöön pohjapiirustusten tai valokuvien päälle (**Kuvio 1.**). Järjestelmä toimii pilvipalvelussa, joten sen käyttöönotto on erittäin helppoa.

Kuvio 1. maintBox-tapahtumien esimerkkinäkymä /15/

2 KÄYTETYT LAITTEET JA TEKNOLOGIAT

2.1 RuuviTag

RuuviTag (**Kuvio 2.**) on edistyksellinen monipuolisia antureita sisältävä Bluetooth-beacon, jonka pieni koko ja vähävirtaisuus tekevät siitä ihanteellisen laitteen nykyaikaisiin IoT-järjestelmiin. Laitteen antureilla voidaan mitata lämpötilaa, kosteutta, ilmanpainetta ja kiihtyvyyttä, joten erilaisia käyttötapauksia on useita.

Kuvio 2. RuuviTag-beacon /16/

RuuviTag sisältää ARM Cortex M4F -prosessorin, 512 kt Flash- ja 64 kt RAM-muistia, sekä tuen lyhyen kantaman NFC-A-, BLE- , ANT- ja Gazel-protokollille. Laite käyttää toimiakseen yhtä CR2477- tai CR2450-paristoa, jonka kapasiteetti riittää sen tarpeisiin useammaksi vuodeksi. Sen halkaisija on 45 mm ja toimintalämpötila -40 °C:sta +85 °C:een. /16/

Tässä opinnäytetyössä RuuviTagia oli tarkoitus käyttää ilmankosteutta ja lämpötilaa mittaavana ja välittävänä beaconina, mutta RuuviTagin alkuvaiheen tuotannon ongelmien, sekä opinnäytetyön erittäin rajallisen ajan vuoksi, se jouduttiin vaihtamaan Raspberry Pi -tietokoneeseen.

RuuviTag ottaa tulevaisuudessa Raspberry Pi -tietokoneen paikan toteutetussa järjestelmässä.

2.2 Raspberry Pi 3 Model B

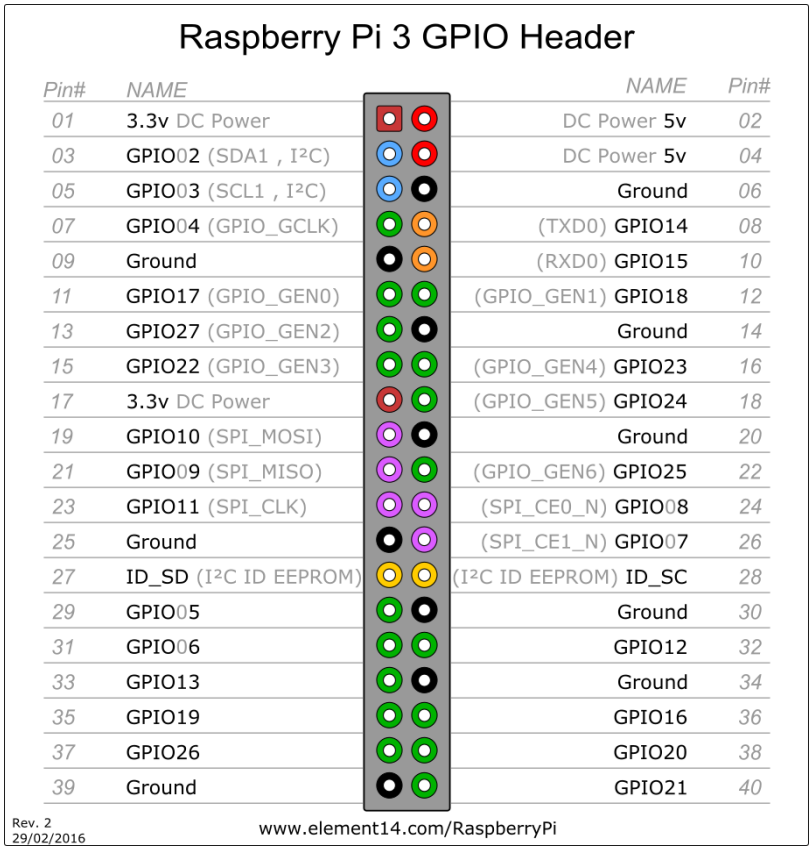
Raspberry Pi on brittiläisen Raspberry Pi Foundationin kehittämä luottokortin kokoinen yhden piirilevyn tietokone, jonka edullinen hinta ja monipuoliset liitännät tekevät siitä erinomaisen alustan erilaisille projekteille.

Ensimmäinen Raspberry Pi -tietokone (Raspberry Pi Model B) julkaistiin 29. helmikuuta 2012, jonka jälkeen eri malleja on toimitettu maailmalle yhteensä yli kahdeksan miljoonaa kappaletta. /1/ Raspberry Pi -tietokoneesta on tällä hetkellä saatavilla viisi eri mallia: Raspberry Pi Model A+ ja B+, Raspberry Pi 2 Model B, Raspberry Pi 3 Model B ja Raspberry Pi Zero. /2/ Kaikille Raspberry Pi -malleille yhteistä on MicroUSB-liitännän kautta syötettävä 5 V käyttöjännite.

Tähän opinnäytetyöhön valikoitui Raspberry Pi -malliston uusin jäsen: Raspberry Pi 3 Model B (**Kuvio 3.**). Raspberry Pi 3 sisältää neliytimisen 1.2 GHz kellotaajuudella toimivan 64-bittisen ARM Cortex-A53 prosessorin, joka on noin 10 kertaa Raspberry Pi 1 -malliston prosessoreita tehokkaampi. Broadcomin BCM2837-järjestelmäpiirin ansiosta Raspberry Pi 3:ssa on integroitu WiFi 802.11n- ja tämän opinnäytetyön kannalta välttämättömän Bluetooth low energy -tuki.

Kuvio 3. Raspberry Pi 3 Model B /1/

Muistipuolelta löytyy 1 GB keskusmuistia jaettuna näytönohjaimen kanssa sekä microSDHC-korttipaikka käyttöjärjestelmää varten. Raspberry Pi 3 sisältää myös neljä kappaletta perinteisiä USB 2.0 -portteja, 10/100 Mbit/s Ethernet-portin, analogisen 3.5 mm naarasliittimellisen äänilähdön ja HDMI-videolähdön. GPIO-pinnejä ulkoisia kytkentöjä varten löytyy 26 kappaletta, joista loput 14 ovat maadoitusta ja virransyöttöä varten (**Kuvio 4.**).



Kuvio 4. Raspberry Pi 3 Model B GPIO-pinnien sijoittelu /3/

Raspberry Pi 3 Model B -tietokoneen tehokkaampi rauta ja uudet langattomat yhteydet näkyvät ymmärrettävästi myös virrankulutuksessa, joten virtalähteen suositeltu antovirta 5 V jännitteellä tulisi olla 2,5 A. Virtalähteen suositusvirta perustuu laitteen virrankulutukseen maksimirasituksessa, joten minimoimalla piirin langattomien toimintojen ja ulkoisten lisälaitteiden määrä, voidaan virrankulutus saada laskemaan jopa 400 mA:iin (**Kuvio 5.**).

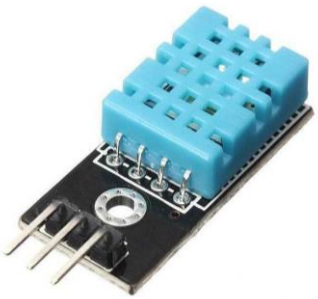
Product	Recommended PSU current capacity	Maximum total USB peripheral current draw	Typical bare-board active current consumption
Raspberry Pi Model A	700mA	500mA	200mA
Raspberry Pi Model B	1.2A	500mA	500mA
Raspberry Pi Model A+	700mA	500mA	180mA
Raspberry Pi Model B+	1.8A	600mA/1.2A (switchable)	330mA
Raspberry Pi 2 Model B	1.8A	600mA/1.2A (switchable)	
Raspberry Pi 3 Model B	2.5A	1.2A	~400mA

Kuvio 5. Raspberry Pi -mallien virtatietoja /2/

Käyttöjärjestelmäksi työhön valittiin Debian-pohjainen Raspberry Pi:lle optimoitu Raspbian Jessie Lite -Linux-jakelu pienen fyysisen kokonsa ja toimintavalmiutensa vuoksi.

2.3 DHT11

DHT11 (**Kuvio 6.**) on edullinen ja vähävirtainen sisätilojen ilmankosteuden ja lämpötilan mittaamiseen tarkoitettu digitaalinen anturi, joka tarkkailee ympäröivää ilmaa kapasitiivisen kosteusanturin ja termistorin avulla. DHT11 palauttaa mitatun datan 40 bittiä pitkänä jonona, joka sisältää suhteellisen kosteuden, lämpötilan ja tarkistussumman.



Kuvio 6. DHT11 kosteus- ja lämpötila-anturi /4/

DHT11 mittaa kosteuden ±5 % tarkkuudella 20-80 % alueella ja lämpötilan ±2 °C tarkkuudella 0-50 °C alueella. Anturi toimii 3-5.5 V käyttöjännitteellä ja sen virrankulutus mittauksen aikana on 0,3 mA ja levossa 60 μA. /5/

2.4 Android

Android on Googlen omistama mobiililaitteille suunnattu käyttöjärjestelmä, joka perustuu avoimen lähdekoodin Linux-ytimeen. Se on yksi maailman käytetyimmistä mobiilikäyttöjärjestelmistä ja sen sovelluskaupasta löytyy yli 1,6 miljoonaa sovellusta. /7/

Android-käyttöjärjestelmän kehityksestä vastaa 84 eri yrityksestä koostuva Open Handset Alliance -yhteenliittymä. Open Handset Alliancen tarkoituksena on kiihdyttää nykyistä puhelinkehitystä ja tarjota kuluttajille rikkaampi Android-kokemus. Yhteenliittymään kuuluu muun muassa HTC Corporation, Huawei Technologies, LG Electronics Inc, Motorola Inc, Samsung Electronics ja monia muita tunnettuja puhelinvalmistajia. /6/

Maaliskuussa 2013, Google esitteli Android Studio -nimisen kehitysympäristön alpha-version Android-sovellusten kehittämiseen, jonka ensimmäinen vakaa versio julkaistiin joulukuussa 2014. /8/ Android Studio korvasi aiemmin Eclipse-kehitysympärisössä käytetyn ADT-liitännäisen, ja onkin nyt Googlen ensisijainen ympäristö natiivien Android-sovellusten kehittämiseen.

2.4.1 Android-arkkitehtuuri

Android-käyttöjärjestelmä koostuu viidestä eri kerroksesta (**Kuvio 7.**).

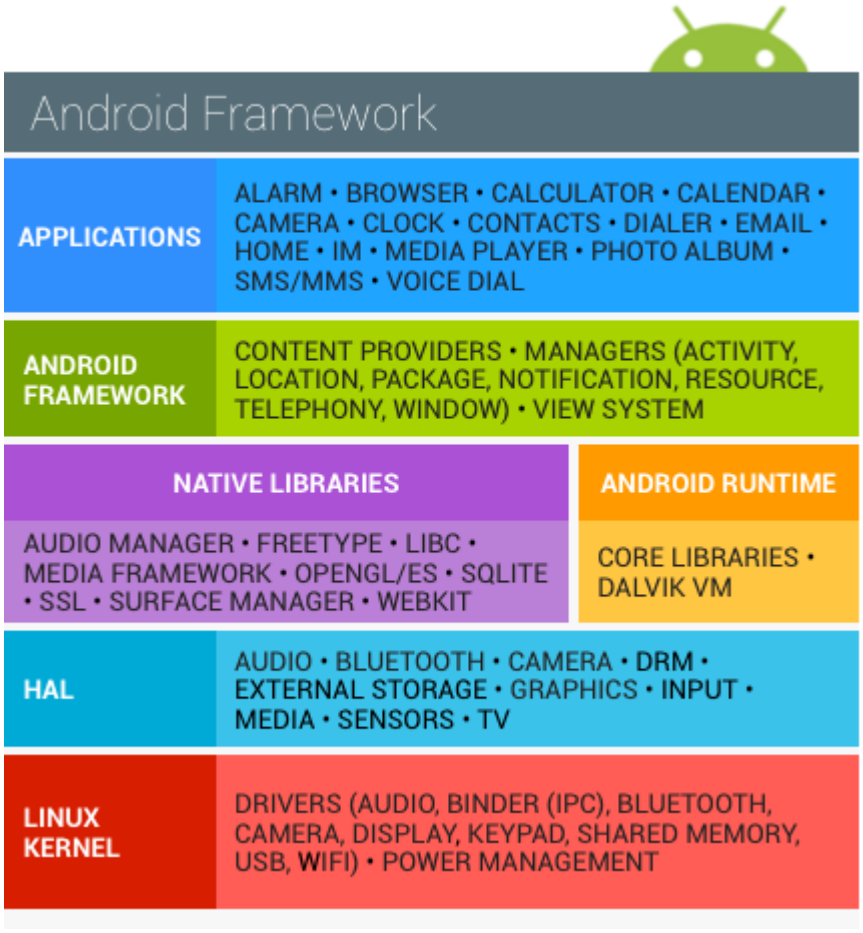
Alimmaiset Linux-ydin ja HAL -kerrokset, tarjoavat yhdessä rajapinnan, jonka avulla sovellukset voivat suoraan kommunikoida fyysisen laitteiston kanssa. Linux-ydin huolehtii myös järjestelmän resurssien jaosta.

HAL-kerroksen yläpuolella oleva kirjastokerros sisältää muun muassa SQLite-tietokannan ja SSL-salausprotokollan kirjastot ja näin ollen mahdollistaa niiden suoran käytön Android-laitteella. Samasta kerroksesta löytyy myös Android

Runtime, joka käsittää Dalvik-virtuaalikoneen ja Javan ydinkirjastot. Android 5.0 (Lollipop) ja uudemmat Android-versiot käyttävät Dalvik-virtuaalikoneen sijasta ART:tä. ART parantaa huomattavasti sovellusten suorituskykyä tehokkaamman roskankeräyksen ja AOT-käännöksen avulla.

Toiseksi ylin kerros, eli sovelluskehyskerros hallitsee Android-laitteen perustoimintoja, kuten sovellusten elinkaarta, sovellusten välistä datan siirtoa, puhelujen hallintaa ja laitteen sijainnin välittämistä sovelluksille.

Ylimmästä kerroksesta, eli sovelluskerroksesta, löytyvät esiasennetut ja käyttäjän asentamat sovellukset. /10/



Kuvio 7. Android-käyttöjärjestelmän arkkitehtuuri /9/

2.5 JSON

JSON (JavaScript Object Notation) on kevyt, yksinkertainen ja helppolukuinen esitysmuoto tiedon jäsentämiseen. Sitä käytetään pääasiassa palvelimen ja web-sovelluksen väliseen tiedonsiirtoon, sekä datan tallentamiseen.

JSON koostuu avain-arvo -pareista, joissa avaimen tietotyyppi on string ja arvon tyyppi string, boolean, array, objekti tai numeerinen. JSON-objektin sisältämät avain-arvo -parit erotetaan toisistaan pilkulla, ja itse avain ja arvo kaksoispisteellä. /11/

Esimerkki yksinkertaisesta kahden avain-arvo -parin JSON-objektista:

```
{”first”:”foo”,”second”:”bar”}
```

2.6 Palvelimen teknologiat

2.6.1 Tomcat

Tomcat on Apache Software Foundationin kehittämä avoimen lähdekoodin web-sovelluspalvelin, jota käytetään Java-pohjaisten web-palveluiden suorittamiseen. Se on toteutettu käyttäen Java-yhteisön kehittämiä Java Servlet, Java Server Pages, Java Expression Language ja Java WebSocket -teknologioita. /12/

2.6.2 REST

REST (REpresentional State Transfer) on arkkitehtuurimalli, jota käytetään web-palveluiden toiminnallisuuksiin HTTP-protokollan avulla. HTTP-pyyntöjä voidaan käyttää kaikkiin CRUD (Create/Read/Update/Delete) -operaatoihin eli datan lukemiseen, lähettämiseen ja poistamiseen.

REST on web-palveluiden tapaan alusta- ja kieliriippumaton. Se on myös palvelimelle kevyempi, sekä pyynnöiltään huomattavasti yksinkertaisempi kuin esimerkiksi SOAP. /17/

2.7 Ohjelmointikielet

2.7.1 C

C on Dennis Ritchien yhdysvaltalaisessa Bell Labs -tutkimuskeskuksessa 1970-luvun alkupuolella kehittämä yleiskäyttöinen ohjelmointikieli UNIX-käyttöjärjestelmille. Sen yksinkertaisuus, tehokkuus ja siirrettävyys ovat pitäneet kielen erittäin laajalti käytössä.

C on tällä hetkellä käytetyin kieli käyttöjärjestelmien ohjelmoimiseen ja sillä on kirjoitettu muun muassa tunnettu avoimen lähdekoodin tietokantajärjestelmä MySQL. /18/

2.7.2 Java

Java on Sun Microsystemsin 1990-luvun alussa kehittämä GPL-lisenssin alainen oliopohjainen ohjelmointikieli. Java on laitteistoriippumaton, mutta tarvitsee JRE (Java Runtime Environment) -ympäristön käännettyjen ohjelmien suorittamiseen. Ennen ohjelman suorittamista, Java-lähdekoodi käännetään tavukoodiksi, joka voidaan tämän jälkeen suorittaa JVM (Java Virtual Machine) -virtuaalikoneella. JVM-virtuaalikone tulkitsee käännettyä tavukoodia suorituksen aikana käyttöympäristön tukemiksi komennoiksi. /19/

3 JÄRJESTELMÄN MÄÄRITTELY

Järjestelmä toteutetaan Devatus Oy:n toiveiden mukaisesti, tarkoituksena integroida Bluetooth low energy -beacon jo olemassa olevaan maintBox-järjestelmään.

Integroitavan beaconin tulee välittää lämpötila- ja kosteusdataa Android-laitteelle iBeacon-protokollan avulla. Android-laitteelta data välitetään edelleen maintBox-järjestelmään jatkokäytettäväksi. Android-sovelluksessa on huomioitava, että mittausdataa lähettäviä beaconeita voi olla yksi tai useampia, sekä dataa vastaanottavan palvelimen tietojen tulee olla helposti käyttäjän vaihdettavissa.

3.1 Toiminnot

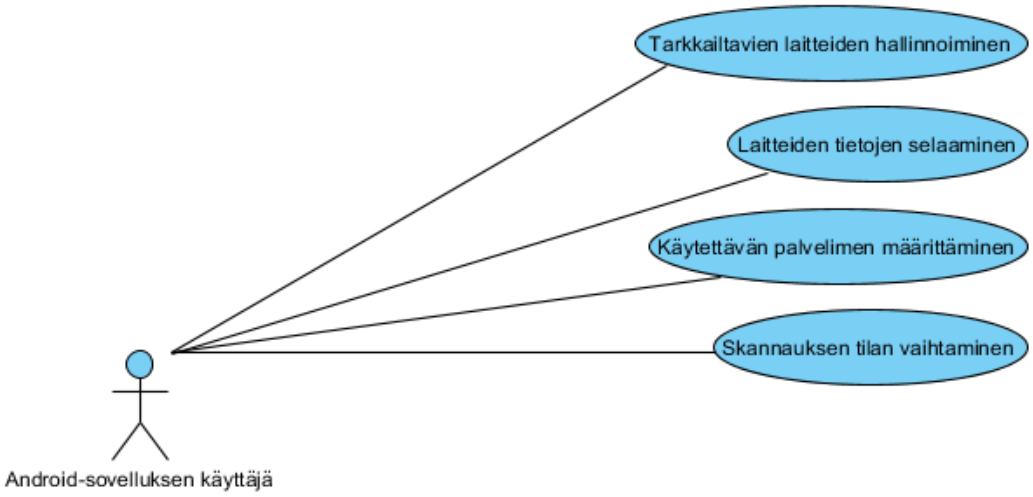
Taulukossa 1 on järjestelmältä odotetut toiminnot niiden tärkeyden mukaan numeroituna: 1 - tulee olla (Must-have), 2 - hyvä jos olisi (Good-to-have) ja 3 - kiva jos olisi (Nice-to-have).

Taulukko 1. Järjestelmän toiminnot

Viite	Toiminnon kuvaus	Tärkeys
T1	Lämpötila- ja kosteustiedon välittäminen maintBox-järjestelmään iBeacon-protokollan avulla	1
T2	Sovelluksella tarkkailtavien laitteiden valitseminen	1
T3	Sovelluksen dataa vastaanottavan palvelimen määrittäminen	1
T4	Sovelluksella mahdollista useamman laitteen samanaikainen tarkkailu	1
T5	Sovelluksen vastaanottaman datan väliaikainen tallentamien palvelimen ollessa tavoittamattomissa	2
T6	Sovelluksesta palvelimelle lähetettävän datan lähetysaikavälin muuttaminen	3

3.2 Käyttötapauskaavio

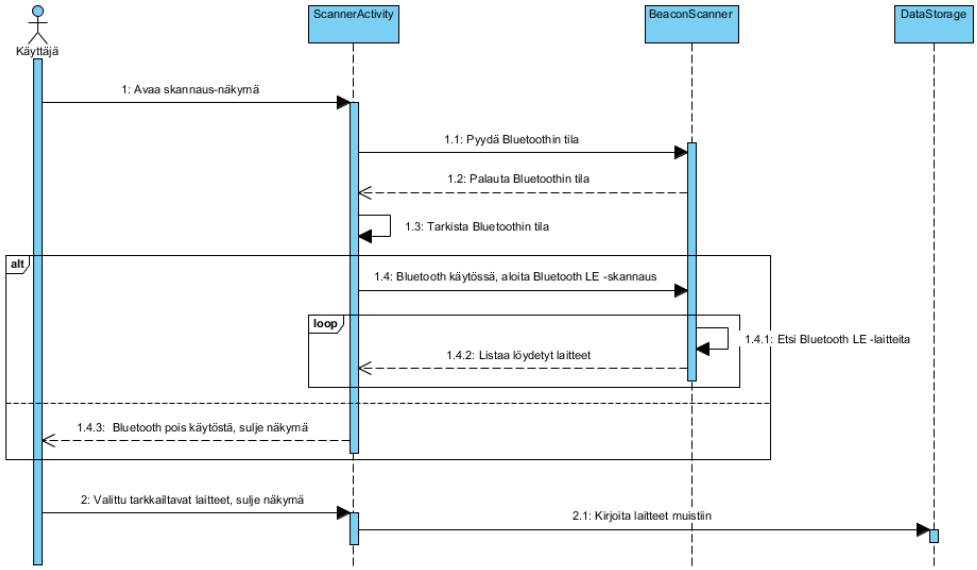
Käyttötapauskaaviolla (**Kuvio 8.**) esitetään Android-sovelluksen käyttäjän hallittavissa olevat toiminnot. Käyttäjä voi lisätä tai poistaa tarkkailtavia laitteita, selata niiden tietoja, määrittää mittausdatan keräämiseen käytettävää palvelinta, sekä vaihtaa skannauksen tilaa.



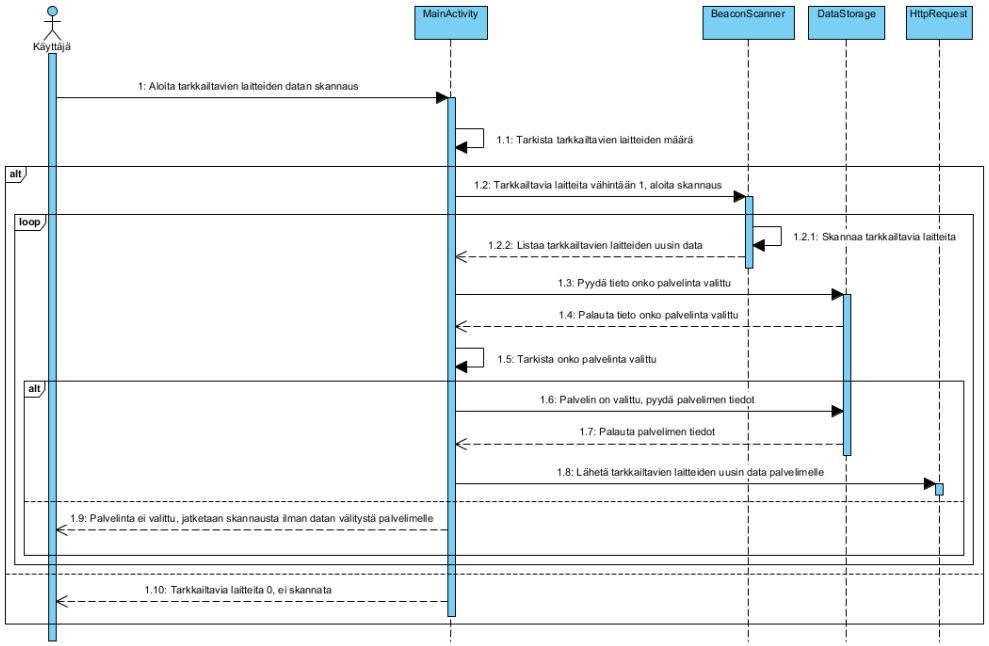
Kuvio 8. Käyttötapauskaavio

3.3 Sekvenssikaaviot

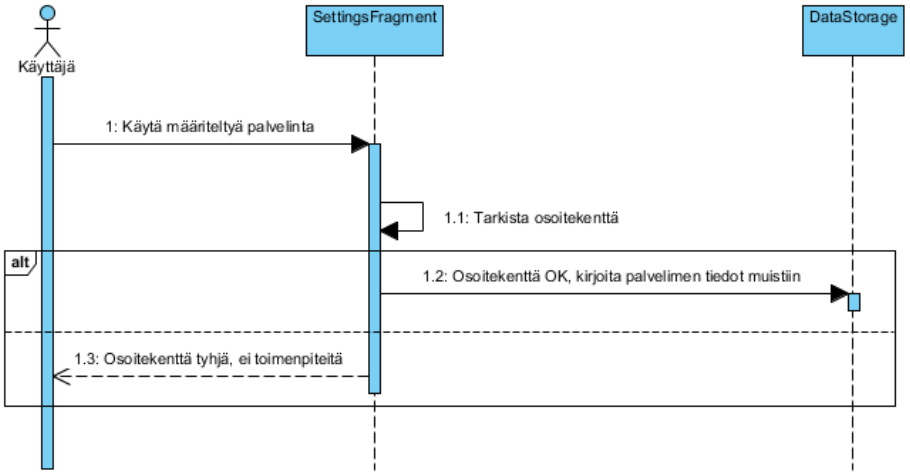
Sekvenssikaaviolla kuvataan yksinkertaistetusti käyttäjän aloittaman toiminnon etenemistä järjestelmässä. Kaavioilla on esitetty yllä olevan käyttötapauskaavion (**Kuvio 8.**) kayttötapauksista monimutkaisemmat eli laitteiden lisäys (**Kuvio 9.**), laitteiden datan skannaus (**Kuvio 10.**), palvelimen määritys (**Kuvio 11.**) sekä laitteiden poisto (**Kuvio 12.**).



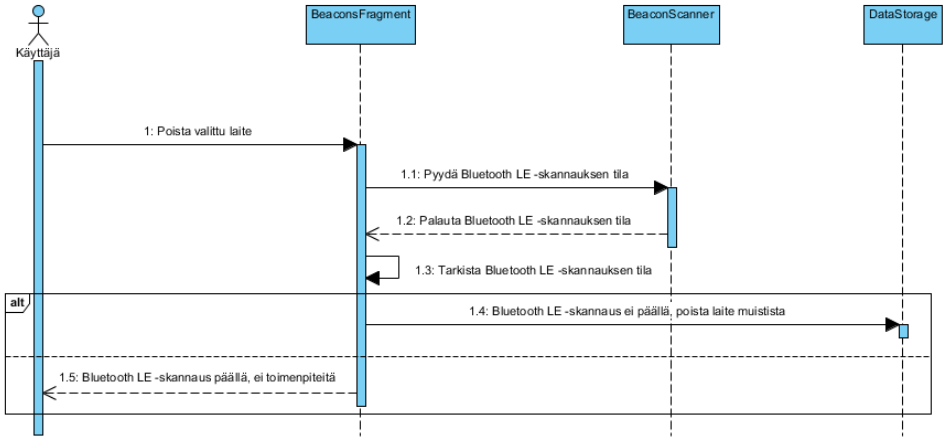
Kuvio 9. Laitteiden lisäys



Kuvio 10. Laitteiden datan skannaus



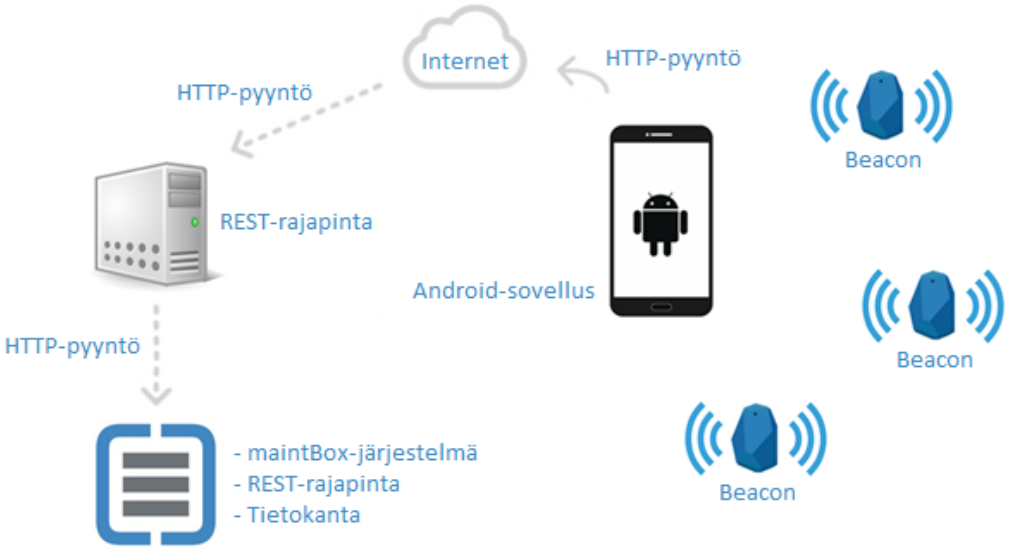
Kuvio 11. Palvelimen määritys



Kuvio 12. Laitteiden poisto

4 JÄRJESTELMÄN TOTEUTUS

Toteutettu järjestelmä (**Kuvio 13.**) koostuu ilmankosteutta ja lämpötilaa mittaavista Bluetooth low energy -beaconeista, jotka lähettävät mitattua dataa ympärilleen iBeacon-protokollan avulla. Lähetetty mittausdata vastaanotetaan Android-laitteella, jossa data käsitellään ja esitetään käyttäjälle selkokielisenä (esim. temperature 21.7, humidity 22.0). Android-laitteelta dataa voidaan lähettää käyttäjän valitsemalle palvelimelle REST-rajapinnan avulla, josta se lopuksi välitetään maintBox-järjestelmän tietokantaan jatkokäytettäväksi.



Kuvio 13. Järjestelmän rakenne ja toiminta

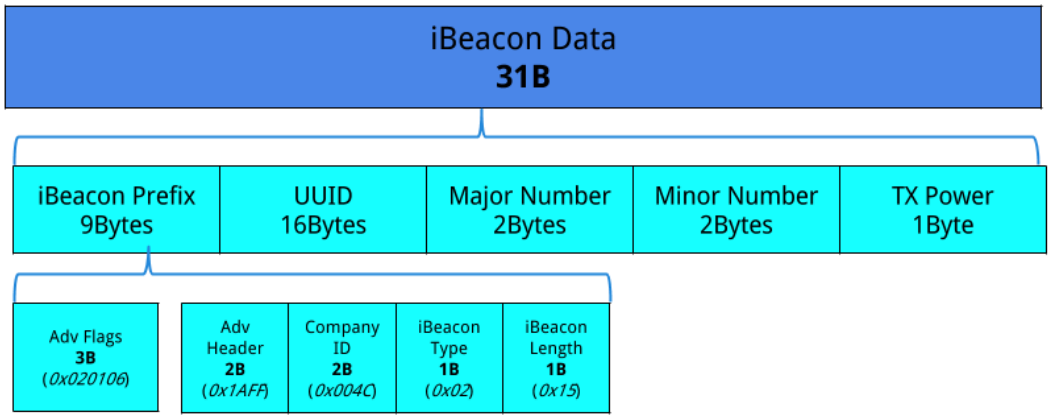
4.1 iBeacon-protokolla

iBeacon-protokollan mukainen datapaketti (**Kuvio 14.**) on pituudeltaan 31 tavua, joka sisältää 9 tavun prefiksin, 16 tavun UUID:n, 2 tavun major- ja 2 tavun minor-tunnisteen, sekä 1 tavun lähetystehon.

iBeacon prefiksin ensimmäiset 3 tavua sisältävät Bluetooth lähetykseen liittyvät liput. Lipuilla voidaan kertoa paketin vastaanottajalle esimerkiksi, että sen lähettänyt laite on tarkoitettu ainoastaan yksisuuntaiseen lähetykseen. Seuraavat 2 tavua kertovat paketissa jäljellä olevan tavumäärän, sekä valmistajakohtaisen datan aloituskohdan. Valmistajakohtaisen datan ensimmäiset 2 tavua sisältävät

valmistajan tunnisteen, jota seuraavat kaksi viimeistä tavua sisältävät iBeaconin tyypin, sekä prefiksin jälkeen jäljellä olevan tavumäärän yhteensä.

iBeacon UUID on tyypillisesti yrityskohtainen 16 tavua pitkä tunniste, jota seuraa major- ja minor-tunnisteet. Kaksi tavua pitkän major-tunnisteen avulla määritellään mihin kokonaisuuteen laite kuuluu. Kaksi tavua pitkän minor-tunnisteen avulla taas määritellään minkä kokonaisuuden laite on kyseessä. Viimeinen tavu paketissa kertoo valmistajan tai käyttäjän kalibroiman laitekohtaisen lähetystehon. Lähetysteho on tärkeä osa vataanottavan- ja lähettävän laitteen välisen etäisyyden laskemisessa.



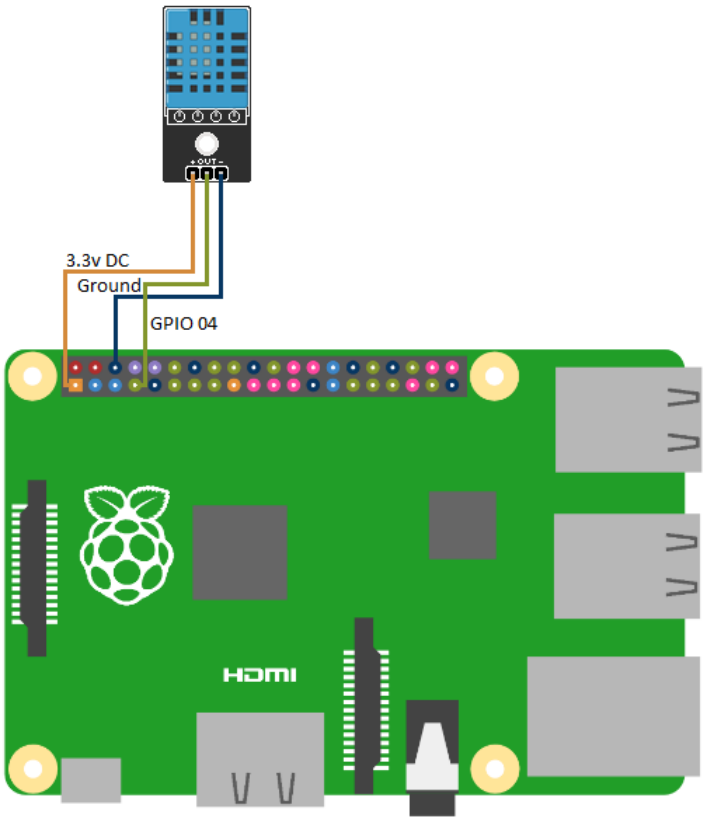
Kuvio 14. iBeacon-datan rakenne /20/

4.2 Beacon toteutus

Ilmankosteutta ja lämpötilaa välittävä beacon toteutettiin Raspberry Pi 3 Model B -tietokoneen ja sen GPIO-pinneihin kytketyn DHT11-kosteus- ja lämpötila-anturin avulla (**Kuvio 15.**).

Raspberry Pi:lle tehtiin C-kielinen ohjelma, jolla DHT11-anturin dataa luetaan ja lähetetään eteenpäin iBeacon-protokollan avulla. Mitattu data sijoitetaan lähetettävän datapaketin UUID-tunnisteeseen, josta luetut arvot voidaan myöhemmin erotella toisistaan ja muuttaa luettavaan selkokieliseen muotoon.

Datan lähettämiseen käytettiin Linux-ytimen omaaville käyttöjärjestelmille suunnitellun BlueZ Bluetooth-protokollapinon hciconfig ja hcitool -työkaluja.



Kuvio 15. DHT11-anturin kytkentä

4.2.1 DHT11-anturin datan lukeminen

DHT11-anturin hetkellisten ilmankosteus- ja lämpötilatietojen lukeminen aloitetaan start-signaalilla, jonka seurauksena anturi siirtyy virransäästötilasta käyttötilaan odottamaan signaalin päättymistä. Jotta anturi ehtii varmasti reagoida start-signaaliin, sen on oltava pituudeltaan vähintään 18 ms. Start-signaalin päätyttyä anturi lähettää 40 bittiä pitkän vastauksen mikrokontollerille, joka sisältää mitatun suhteellisen ilmankosteuden, lämpötilan ja tarkistussumman. Mittausdata luetaan tämän tapahtuman jälkeen anturille osoitetusta Raspberry Pi -tietokoneen GPIO-pinnistä, jonka jälkeen anturi asettaa itsensä takaisin virransäästötilaan.

DHT11-anturilla mitatut ilmankosteus- ja lämpötilatiedot on ilmoitettu 40 bitin vastauksessa kokonaislukuina. Vastauksen ensimmäinen tavu kertoo ilmankosteuslukeman ennen desimaalierotinta ja toinen tavu

ilmankosteuslukeman desimaalierottimen jälkeen. Seuraavat kaksi tavua ilmoittavat lämpötilan ennen desimaalierotinta ja desimaalierottimen jälkeen. Viimeinen tavu sisältää tarkistussumman, jolla varmistetaan luetun datan virheettömyys. Mittausarvot on luettu onnistuneesti, jos tarkistussumma on luettujen mittausarvojen kokonaislukujen summa.

Jos tarkistussumma ei täsmää, odotetaan 200 ms ja aloitetaan mittausprosessi alusta. Mittausprosessia suoritetaan uudestaan kunnes tarkistussumma täsmää.

Jos tarkistussumma täsmää, muodostetaan luetuista arvoista kaksi float-tyyppistä lukua, jotka lähetetään seuraavassa vaiheessa eteenpäin iBeacon-protokollan avulla.

Ilmankosteutta ja lämpötilaa DHT11-anturilta lukeva C-ohjelma toteutettiin Raspberry Pi -tietokoneen GPIO-pinnien käsittelyä helpottavan wiringPi -C-kirjaston avulla.

4.2.2 Datan lähettäminen

Datan lähettäminen eteenpäin toteutettiin beaconeille tyyppillisesti Bluetooth low energy advertising -tilassa, joka mahdollistaa monen beaconin yhtäaikaisen tarkkailemisen yhdellä lukevalla laitteella. Beaconia ja lukevaa laitetta ei tarvitse erikseen parittaa, vaan data on luettavissa suoraan etäisyyden ollessa laitteiden välillä riittävä.

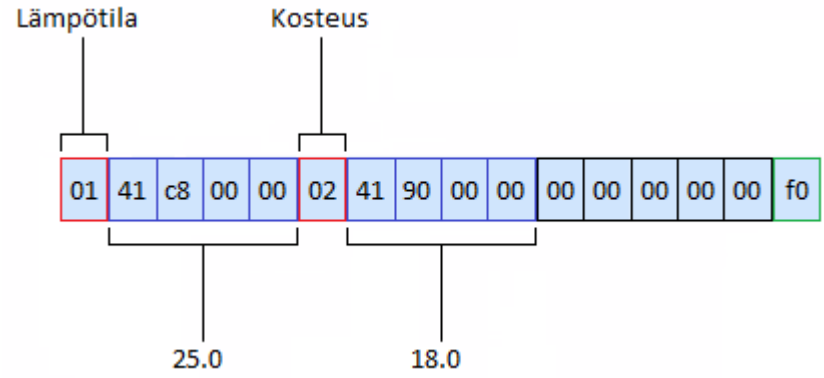
Toteutetussa C-kielisessä ohjelmassa laite valmistellaan Bluetooth low energy advertising -tilaa varten hciconfig-työkalun avulla heti ohjelman käynnistyttyä. Bluetooth-laitteena toimii Raspberry Pi 3 Model B -tietokoneen integroitu Bluetooth-piiri, jolle suoritetaan seuraavat komennot C-ohjelmasta käsin:

```
/* Alustetaan ja otetaan hci0-laite käyttöön */
system(“hciconfig hci0 up”);

/* Pystäytetään Bluetooth-laitteiden skannaus */
system(“hciconfig hci0 noscan”);

/* Hylätään yhteyspyynnöt laittelta */
system(“hciconfig hci0 leadv 3”);
```

Kun Bluetooth-laite on valmisteltu, luetaan mittausdata DHT11-anturilta, joka palauttaa ilmankosteuden ja lämpötilan float-tyyppisessä pointterissa. Palautetut float-tyyppiset mittausarvot muutetaan 4-tavuisiksi heksoiksi ja niiden alkuun lisätään 1 tavun mittainen tunniste, jonka avulla tiedetään onko mitattu arvo lämpötilaa vai kosteutta. 5 tavun mittaiset mittausarvot tunnisteineen sijoitetaan peräkkäin ja niistä muodostetaan 16-tavuinen UUID myöhemmin tapahtuvaa lähetystä varten (**Kuvio 16.**). UUID:n viimeistä tavua vuorotellaan 0x0F ja 0xF0 välillä mittausarvojen vaihtuessa, jotta lukeva laite tietää vastaanottaneensa uuden mittausarvon.



Kuvio 16. Mittausarvojen sijoittelu UUID-tunnisteessa

Kun mittausdatan sisältävä UUID on muodostettu, lisätään se hcitool-työkalun Bluetooth low energy advertising -lähetyskomentoon iBeacon-protokollan mukaiseen paikkaan ja aloitetaan lähetys:

```
/* Muodostetaan hcitool-komento char-tyyppiseen ”ibeacon_ad”-listaan */
snprintf(ibeacon_ad, CMD_LENGTH,
          “hcitool –i hci0 cmd 0x08 0x0008 1E %s %s %s %s %s”,
          BEACON_PREFIX, uuid, MAJOR, MINOR, TX_POWER);

/* Suoritetaan muodostettu hcitool-komento */
system(ibeacon_ad);
```

Edellä viestin lähetykseen käytetty komento voidaan purkaa seuraavanlaisiin osiin: **-i hci0** osoittaa Bluetooth-laitteen jolla komento suoritetaan, **cmd** kertoo hcitool-työkalulle, että halutaan lähettää seuraavanlaista dataa, **0x08** kertoo datan olevan tietyssä muodossa, **0x0008** kertoo käytettävän komennon eli tässä

tapauksessa HCI_LE_Set_Advertising_Data ja **1E** kertoo merkitsevien tavujen määrän lähetettävässä datassa. /21/

Raspberry Pi lähettää nyt iBeacon-protokollan mukaista dataa ympärilleen yhden sekunnin välein ja päivittää lähetettävän viestin sisältämät mittausarvot uusiin C-ohjelmassa määritellyn DHT11-anturin lukuajan, eli viiden sekunnin välein.

4.3 Android-sovellus

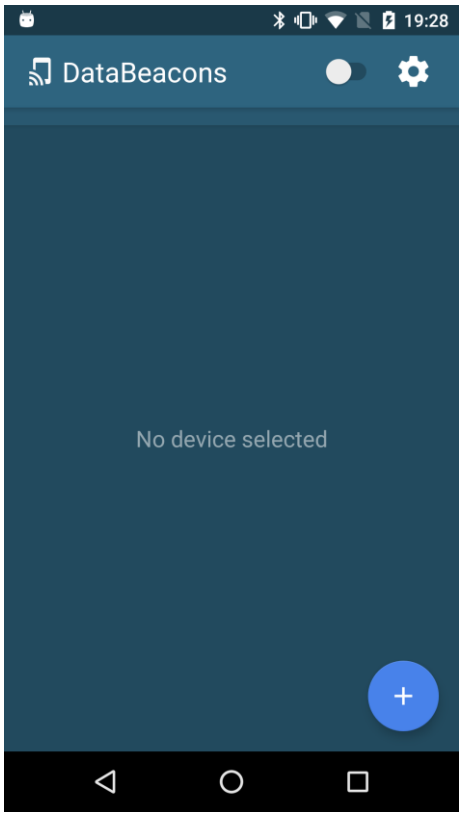
Android-sovellus toteutettiin natiivisti käyttäen Java-kieltä. Laitevaatimuksina sovelluksen toiminnalle on vähintään Android 5.0 (Lollipop) -järjestelmäversio, sekä Bluetooth low energy -tuki. Android-sovelluksen avulla voidaan vastaanottaa aiemmin esitellyn beaconin lähettämää iBeacon-protokollan mukaista dataa. Sovellus tulkitsee vastaanotetun datan selkokieliseen muotoon käyttäjän nähtäville ja lähettää sen eteenpäin palvelimelle HTTP-pyynnön avulla.

4.3.1 Käyttöliittymä

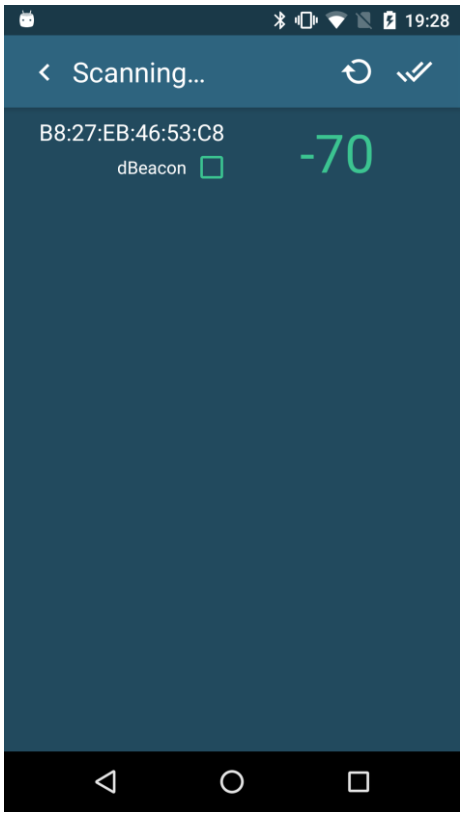
Sovelluksen käyttöliittymä suunniteltiin mahdollisimman helppokäyttöiseksi, intuitiiviseksi ja yksinkertaiseksi. Suunnittelussa oli huomioitava, että luettavia beaconeita voi olla yksi tai useampia, sekä beaconkohtaista dataa on voitava esittää tekemättä ulkoasusta vaikeasti luettavaa ja sekavaa. Käyttöliittymä toteutettiin kokonaisuudessaan Android Studio -kehitysympäristön XML-layout editorilla.

Kun käyttäjä haluaa lisätä tarkkailtavan beaconin sovelluksen laite-näkymään (**Kuvio 17.**), aloitetaan laiteskannaus painamalla näytön oikeassa alanurkassa leijuvaa sinistä lisäyspainiketta, jolloin skannausnäkymä (**Kuvio 18.**) aukeaa ja beaconien etsintä alkaa välittömästi.

Skannausnäkymässä löydetyt beaconit lisätään listaan allekkain ja niistä esitetään laitekohtainen MAC-osoite, sekä sekunnin välein päivittyvä RSSI-vastaanottoteho desibelimilliwatteina. Listasta käyttäjä voi valita tarkkailtavat beaconit koskettamalla valintaruutua ja hyväksymällä ne tarkkailtavaksi painamalla oikeassa ylänurkassa olevaa valintapainiketta.



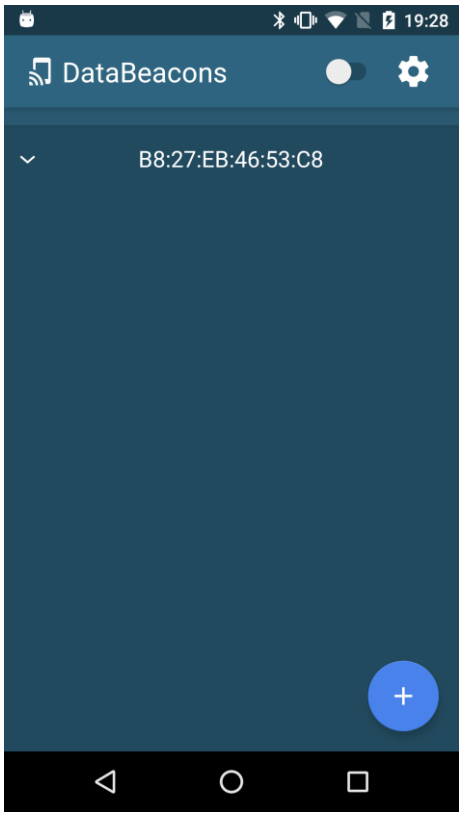
Kuvio 17. Tyhjä laitenäkymä

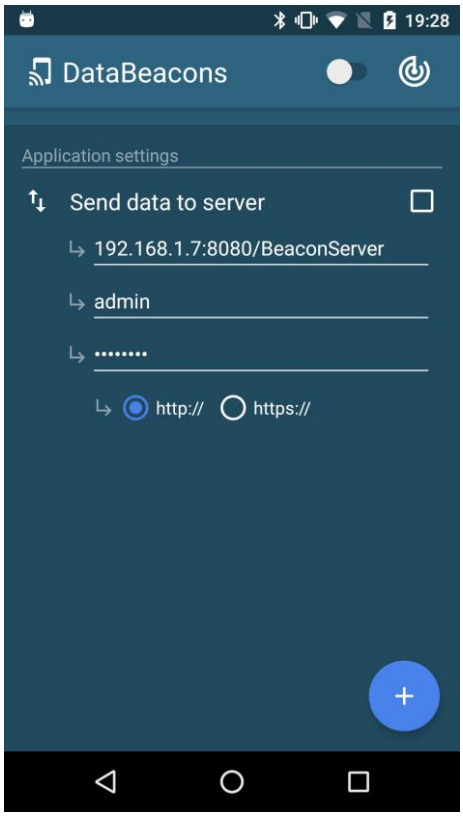


Kuvio 18. Skannausnäkymä

Valintapainike palauttaa sovelluksen takaisin alun laitenäkymään, jossa on nyt listattuna käyttäjän valitsemat beaconit (**Kuvio 19.**). Laitenäkymään lisätyistä beaconeista saadaan näkyviin lisätietoja koskettamalla halutun beaconin MAC-osoitetta.

Laitenäkymää oikealle liu'uttamalla tai ylänurkan asetus-painiketta painamalla päästään asetusnäkymään (**Kuvio 20.**), jossa käyttäjä voi määrittää mittausdataa vastaanottavan palvelimen tiedot. Mittausdataa voidaan myös tarkkailla vain paikallisesti Android-laitteen näytöltä lähettämättä sitä ollenkaan palvelimelle, jolloin "Send data to server" -valintaruutu tulee jättää valitsematta. Jos mittausdata kuitenkin halutaan lähettää palvelimelle, on palvelimen tiedoista täytettävä vähintään osoitekenttä ja hyväksyttävä "Send data to server" -valintaruutu. Käytettävää palvelinta ei voi vaihtaa kesken mittausdatan kuuntelemisen.





Kuvio 19. Laitenäkymä beaconilla

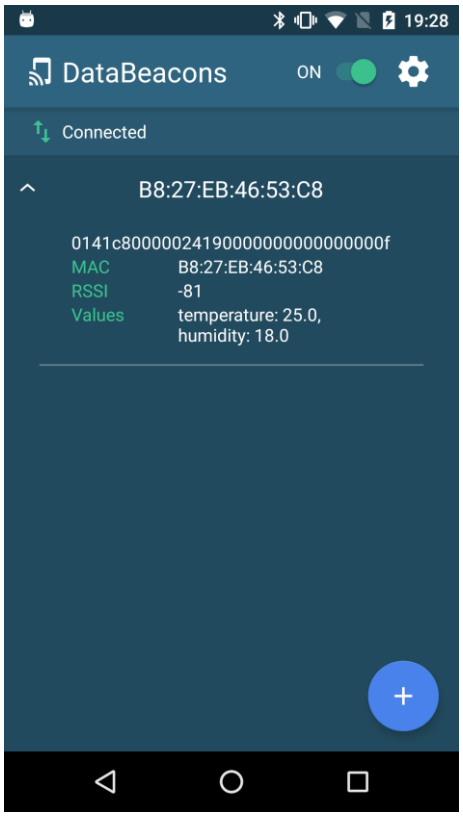
Kuvio 20. Asetusnäkymä

Kun palvelinasetukset on määritelty, voidaan aloittaa valittujen beaconien mittausdatan kuunteleminen kytkemällä yläreunassa oleva kytkin ON-asentoon. Mittausdatan kuuntelu, sekä sen lähetys valitulle palvelimelle alkaa välittömästi.

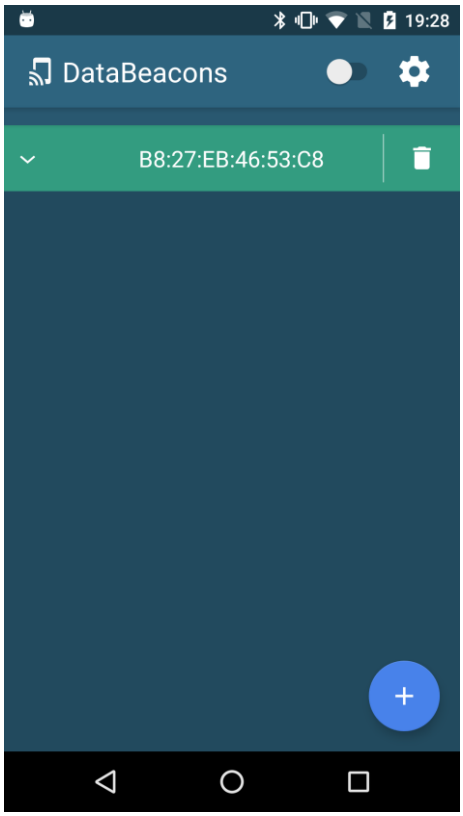
Vastaanotettu mittausdata päivitetään käyttäjän nähtäville laitenäkymään, jossa oletuksena näkyvillä on ainoastaan kuunneltavien beaconien MAC-osoitteet. Osoitetta koskettamalla vastaanotettu mittausdata laajentuu beaconin alapuolelle (**Kuvio 21.** [Mittaus suoritettu asuntoni sähkökaapissa]). Mittausdatasta esitetään viimeisimmät vastaanotetut beaconkohtaiset tiedot: mittausdata heksoina, MAC-osoite, RSSI-vastaanottoteho, sekä Values eli tulkattu mittausdata. Jos käyttäjä on valinnut mittausdatan lähetettäväksi palvelimelle, sovelluksen ja palvelimen välisen yhteyden tila esitetään ylhäällä työkalupalkin alapuolella.

Kun käyttäjä haluaa lopettaa mittausdatan kuuntelemisen ja katkaista mahdollisen palvelinyhteyden, siirretään yläreunassa oleva kytkin OFF-asentoon. Aiemmin

valittuja beaconeita voidaan nyt poistaa pitkään koskettamalla, jolloin valitun beaconin oikeaan reunaan ilmestyy roskakori, jota lyhyesti koskettamalla beacon poistetaa sovelluksesta (**Kuvio 22.**). Poistettu beacon voidaan lisätä uudestaan tarkkailtavaksi skannausnäkymän kautta.



Kuvio 21. Laitenäkymä toiminnassa



Kuvio 22. Laitenäkymä poistotilassa

4.3.2 Beacon-datan lukeminen

Beaconin lähettämän mittausdatan lukemiseen ja laitteiden etsimiseen käytettiin Androidin BluetoothLeScanner-luokkaa. Luokan käyttö vaatii laitteelta seuraavat oikeudet lisättynä sovelluksen AndroidManifest -tiedostoon:

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

<uses-permission android:name=”android.permission.
                                        ACCESS_COARSE_LOCATION” />
<uses-permission android:name=”android.permission.
                                        ACCESS_FINE_LOCATION” />
```

BLUETOOTH- ja BLUETOOTH_ADMIN -oikeuksilla annetaan sovellukselle lupa etsiä uusia Bluetooth-laitteita ja tarvittaessa muodostaa löydettyjen laitteiden kanssa Bluetooth-pariyhteys. Käyttöoikeuksilla ACCESS_COARSE_LOCATION ja ACCESS_FINE_LOCATION, sovellus saa luvan lukea laitteen tarkan sijainnin. /22/

Jos sovellusta käytetään Android 6.0 (Marshmallow) -käyttöjärjestelmäversion omaavassa laitteessa, on edellisten AndroidManifest -tiedostoon lisättyjen käyttöoikeuksien lisäksi sovelluksen käyttäjältä kysyttävä lupa laitteen sijainnin käyttämiseen ennen skannauksen aloittamista. Lupa on hyväksyttävä sovelluksen aloituksen yhteydessä vain kerran. Luvan kysyminen toteutettiin seuraavasti:

```
/**
 * Pyydetään suorituksen aikana BluetoothLeScanner-luokan vaatima
 * ACCESS_COARSE_LOCATION-lupa
 */
private void requestCoarseLocationPermission() {

     // Tarkistetaan käyttöjärjestelmäversio
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

          // Tarkistetaan onko lupa jo myönnetty
          if (ContextCompat.checkSelfPermission(this,
               Manifest.permission.ACCESS_COARSE_LOCATION) !=
               PackageManager.PERMISSION_GRANTED) {

               // Pyydetään lupa lukea laitteen sijainti
               requestPermissions(new String[]{
                    Manifest.permission.ACCESS_COARSE_LOCATION
               }, DEVICE_LOCATION_REQUEST_CODE);
          }
     }
}
```

Kun sovelluksella on tarvittavat käyttöoikeudet, voidaan BluetoothLeScanner-luokan avulla skannata Bluetooth low energy -laitteita, kuten tässä työssä aiemmin esiteltyjä beaconeita. Ennen skannausta kuitenkin varmistetaan, että Android-laitteella on tarvittava Bluetooth low energy -tuki, sekä Bluetooth kytkettynä päälle. Valmistellaan vielä BluetoothLeScanner-objekti skannausta varten:

```
// Valmistellaan BluetoothLeScanner-objekti Bluetooth low energy –
// skannausta varten
BluetoothLeScanner m_bleScanner = BluetoothAdapter.
                        getDefaultAdapter().getBluetoothLeScanner();
```

Bluetooth low energy -laitteiden skannaus voidaan nyt aloittaa kutsumalla BluetoothLeScanner-luokan startScan()-metodia. Metodista startScan() löytyy kaksi erilaista toteutusta, joista toiseen voidaan määritellä ainoastaan käytettävä callback-objekti ja toiseen callbackin lisäksi erilaisia suodattimia sekä asetuksia. Molempia startScan()-toteutuksia päästiin tässä työssä käyttämään, sillä suodattimellisella toteutuksella saatiin kätevästi kuunneltua ainoastaan käyttäjän valitsemia beaconeita, ja suodattamattomalla toteutuksella taas saatiin esiin kaikki riittävällä etäisyydellä olevat beaconit listaan lisäämistä varten.

Kun startScan()-metodia on kutsuttu, sen parametrina tuotuun callback-objektiin palautetaan ScanResult-tyyppisiä skannaustuloksia sitä mukaan kun dataa vastaanotetaan. Callback-objektiin palautuu siis kaikki vastaanotettu data, joten skannauksessa löytyneitä beaconeita listattaessa on tarkistettava onko uutta dataa lähettänyt beacon jo listattu, jos on, korvataan beaconin vanhat tiedot uusilla. Löytyneiden beaconien päivitys listaan toteutettiin seuraavasti:

```
/**
 * Uusien beaconien päivitys listaan
 */
public void updateBeaconList(ScanResult _result) {
     boolean listed = false;

     // Tarkistetaan onko listassa beaconeita
     if (!m_beaconList.isEmpty()) {

          // Verrataan uusinta ja aiempia beaconeita keskenään
          for (Beacon beacon : m_beaconList) {
               // Tarkistetaan löytyykö beacon jo listalta
               if (beacon.getMacAddress().
                         equals(_result.getMacAddress())) {
                    // Varmistetaan että beacon ainoastaan päivitetään
                    listed = true;
                    // Beacon löytyy listalta, joten tarkistetaan onko
                    // vastaanotettu mittaus uusi
                    if (beacon.getMeasurementID() !=
                         _result.getMeasurementID()) {
                         // Jos mittaus on uusi, päivitetään mittaudata
                         m_beaconList.set(m_beaconList.indexOf(beacon),
                              _result);
                    }
                    break; // Beacon on listalla, keskeytetään vertailu
               }
          }
     }
     // Jos lista on tyhjä tai uusi beacon löytyy, lisätään uusin beacon
     // listaan
     if (!listed) {
          m_beaconList.add(_result);
     }
}
```

Edellisessä toteutuksessa verrataan aina uusimman datan lähettäneen beaconin MAC-osoitetta jo listalla oleviin. Jos vastaanotettu beacon löytyy jo listalta, tarkistetaan measurementID:n avulla onko sen lähettämä mittausdata täysin uutta vai jo aiemmin vastaanotettua. getMeasurementID palauttaa beaconkohtaisen viimeisimmän vastaanotetun UUID-tunnisteen viimeisen tavun, jota beacon vaihtaa jokaisen uuden muttauksen jälkeen 0x0F ja 0xF0 välillä. Kun tavu vaihtuu toiseen tiedetään, että mittausarvokin on uutta ja vastaanotettu data on syytä ottaa talteen.

Skannaus lopetetaan yksinkertaisesti kutsumalla stopScan()-metodia, jolle viedään parametrina startScan()-metodissa käytetty callback-objekti. Callback-objektista listaan kerätyt ScanResult-tulokset sisältävät kaiken vastaanotetun beaconkohtaisen datan, joka on pienellä käsittelyllä valmista käyttäjälle esitettäväksi ja palvelimelle lähetettäväksi.

4.3.3 Datan käsittely

Jokaisesta vastaanotetusta ScanResult-tyyppisestä tuloksesta muodostetaan Beacon-objekti, joka sisältää beaconin MAC-osoitteen, varsinaisen mittausdatan sisältävän BeaconData-objektin, RSSI-vastaanottotehon, aikaleiman, sekä tiedon onko uusin data lähetetty palvelimelle. BeaconData-objekti muodostetaan vastaanotetun ScanResult-tuloksen sisältämästä tavujonosta, josta vain mittausdatan sisältävä UUID-tunniste otetaan talteen:

```
/**
 * Parsitaan UUID talteen vastaanotetusta datasta
 *
 * UUID:n ensimmäinen tavu sijaitsee iBeacon-protokollan mukaisesti
 * 9:nnen tavun jälkeen, josta viimeinen on ensimmäisestä tavusta 16
 * tavua eteenpäin
 */
private byte[] parseUuid(_scanResultBytes) {
     int uuidIndex = FIRST_BYTE_OF_UUID;
     byte[] uuidBytes = new byte[UUID_LENGTH];

     for (int currentPos = 0; currentPos < uuidBytes.length;
          currentPos++) {
          uuidBytes[currentPos] = _scanResultBytes[uuidIndex]
          uuidIndex++;
     }
     return uuidBytes;
}
```

Palautetusta UUID-tunnisteesta voidaan nyt erotella itse mittausarvot. Mittausarvojen erottelu toteutettiin seuraavasti:

```
/**
 * Eritellään mittausdata UUID-tunnisteesta ja luodaan jokaisesta
 * Measurement-objekti
 *
 * Eritellyt mittausdatat palautetaan Measurement-tyyppisessä listassa
 */
public ArrayList<Measurement> parseMeasurements(_uuidBytes) {
     ArrayList<Measurement> measurements = new ArrayList<>();

     DecimalFormat decFormat = new DecimalFormat("#");
     decFormat.serMaximumFractionDigits(2);

     // Käydään läpi 5 tavua pitkät mittausarvot
     for (int currentPos = 0; currentPos < MEASUREMENTS_TOTAL_LENGTH;
          currentPos += MEASUREMENT_LENGTH) {

          // Tarkistetaan mittausarvon tyyppi. Jos tyyppi on jotai muuta
          // kun 0x00 eli tyhjä, jatketaan mittausarvon käsittelyä
          if (_uuidBytes[currentPos] != (byte)0x00) {

               // Yhdistetään mittauksen tyyppiä eli ensimmäistä tavua
               // seuraavat tavut
               int mergedBytes = ByteBuffer.wrap(new byte[] {
                    _uuidBytes[currentPos + 1],
                    _uuidBytes[currentPos + 2],
                    _uuidBytes[currentPos + 3],
                    _uuidBytes[currentPos + 4]
               }).getInt();

               // Muutetaan yhdistetyt tavut kahden desimaalin
               // tarkkuudella pyöristetyksi double-tyyppiseksi
               // mittausarvoksi
               double measurement = Double.parseDouble(
                    decFormat.format(Float.intBitsToFloat(mergedBytes)));

               // Luodaan mittausarvosta Measurement-tyyppinen objekti ja
               // lisätään se listaan. Parametreina viedään mittausarvon
               // tyyppi (0x01 -> lämpötila tai 0x02 -> kosteus) ja itse
               // mittausarvo
               measurements.add(
                    new Measurement((int)_uuidBytes[currentPos],
                    measurement));
          }
     }
     return measurements;
}
```

Measurement-objekteista beacon-kohtainen mittausdata on helppo esittää käyttäjälle ja näin ollen myös lähettää eteenpäin palvelimelle.

Jos käyttäjä on valinnut, että vastaanotettu mittausdata lähetetään palvelimelle, käydään Beacon-objektit läpi kahden sekunnin välein ja valitaan niistä vielä lähettämättömät. Lähettämättömien Beacon-objektien tiedoista muodostetaan JSON-objekti, joka sisältää beacon-kohtaisen tunnisteen eli MAC-osoitteen,

mittausarvot selkokielisenä tyyppeineen, sekä UNIX-aikaleiman millisekunteina. Muodostetut JSON-objektit sijoitetaan JSON-array -kokoelmaan, joka lopuksi lähetetään valitulle palvelimelle HttpURLConnection-luokan avulla toteutetulla HTTP-pyynnöllä.

4.4 Palvelin

Mittausdataa maintBox-järjestelmälle välittävän palvelimen REST-rajapinta toteutettiin Jersey-sovelluskehyksen avulla. Palvelin vastaanottaa Android-laitteelta POST-tyyppisiä pyyntöjä JSON-muodossa ja palauttaa vastaukseksi HTTP-vastauskoodeja.

REST-rajapinnalle toteutettiin käyttäjän tunnistus, joka vaatii käyttäjänimen ja salasanan base64 koodattuna ennen mittausdatan jatkokäsittelyä. Jos käyttäjää ei tunnisteta, palautetaan HTTP-vastauskoodi ”401” eli luvaton käyttäjä, ja hylätään mittausdata. Jos käyttäjä tunnistetaan, jatketaan vastaanotetun mittausdatan käsittelyä.

Käyttäjän onnistuneen tunnistuksen jälkeen Android-laitteelta vastaanotetusta JSON-muotoisesta mittausdatasta muodostetaan Measurement-objekteja, jotka sisältävät datan lähettäneen beaconin MAC-osoitteen, mitatut arvot ja tyypit, sekä aikaleiman ”yyyy-MM-dd HH:mm:ss” -muodossa. Measurement-objekteista muodostetaan maintBox-järjestelmän dokumentaation mukaiset merkkijonot, jotka lähetetään sisällöltään ”x-www-form-urlencoded” -tyyppisenä POST-pyyntönä eteenpäin maintBox-järjestelmälle. Järjestelmä sijoittaa vastaanotetun mittausdatan MySQL-tietokantaan, jonka sisältöä voidaan tarkkailla ja hallinnoida maintBox-hallintasivuston avulla.

4.5 Testaus

Järjestelmän kaikki toiminnot testattiin tarkasti toteutuksen aikana jokaisen valmistuneen osan jälkeen. Valmiille järjestelmälle suoritettiin erikseen suunnitellut testitapaukset kokonaisuuden toimivuuden varmistamiseksi (**Taulukko 2.**).

Taulukko 2. Testitapaukset

ID	Vaatimukset	Testattava toiminto	Odotettu tulos
TC1	Android-sovelluksen laitenäkymään lisätty vähintään yksi beacon	Beaconin poisto sovelluksen laitenäkymästä	Valittu beacon poistettu sovelluksesta
TC2	Vähintään yksi tuettu beacon sovelluksen havaittavissa	Beaconin lisäys Android-sovelluksen laitenäkymään	Valittu beacon lisätty sovelluksen laitenäkymään
TC3	Verkkoyhteys ja vähintään yksi tuettu beacon sovelluksen havaittavissa	Beaconilta vastaanotetun mittausdatan välitys Android-sovelluksesta maintBox-järjestelmälle	maintBox-järjestelmä vastaanottaa lähetetyn mittausdatan
TC4	Vähintään yksi tuettu beacon sovelluksen havaittavissa	Beaconin mittausdatan vastaanotto Android-sovelluksella ilman mittausdatan lähetystä palvelimelle	Sovellus vastaanottaa mittausdataa lähettämättä sitä palvelimelle
TC5	Android-sovelluksen laitenäkymään lisättyjen beaconien skannauksen oltava suljettuna	Mittausdataa vastaanottavan palvelimen määritys Android-sovellukseen	Palvelimen tiedot tallennetaan sovellukseen tulevaa mittausdatan lähetystä varten
TC6	Android-sovelluksen laitenäkymään lisättyjen beaconien skannaus ja välitys palvelimelle aktivoituna	Beaconien mittausdatan vastaanotto sovelluksella Android-laitteen ollessa lukittuna	Mittausdatan vastaanotto ja välitys palvelimelle jatkuu normaalisti

5 YHTEENVETO

Opinnäytetyön lopputuloksena saatiin ilmankosteutta ja lämpötilaa Bluetooth low energy -beaconin avulla keräävä kokonaisuus, jossa kerätty mittausdata välitetään Android-sovelluksen avulla eteenpäin maintBox-järjestelmään jatkokäsiteltäväksi.

Työn suurimmat ongelmat sijoittuivat aivan toteutuksen alkuvaiheeseen, sillä alkuperäiseen suunnitelmaan kuuluva RuuviTag Bluetooth-beacon kärsi alkuvaiheen tuotannon ongelmista ja oli näin ollen opinnäytetyön tiukan aikataulun vuoksi vaihdettava Raspberry Pi 3 Model B -tietokoneeseen. Ongelmista huolimatta työ kuitenkin valmistui suunniteltuun ajankohtaan mennessä tärkeimmät ominaisuudet toteutettuina. Työn edetessä sain huomata useaan otteeseen hyvän suunnittelun tärkeyden, sillä jouduin monesti palaamaan aiempiin toteutuksiin korjailemaan suunnitteluvirheitäni.

Järjestelmän jatkokehityksessä iBeacon-protokollaa voitaisiin hyödyntää tehokkaammin suurempien beaconmäärien kanssa, esimerkiksi jakamalla eri arvoja mittaavat beaconit eri rymiin iBeacon-protokollan minor- ja major-tavujen avulla, jolloin Android-sovelluksesta voitaisiin valita näkyviin esimerkiksi vain tärinää tai lämpötilaa mittaavat beaconit. Myös sovelluksella tarkkailtavien beaconien nimeäminen olisi mittausdatan tarkkailun kannalta hyvä ominaisuus.

LÄHTEET

/1/ Raspberry Pi 3 on sale.
Viitattu 9.5.2016.
https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/

/2/ Raspberry Pi faqs.
Viitattu 9.5.2016.
https://www.raspberrypi.org/help/faqs/

/3/ Raspberry Pi 3 Model B GPIO 40 Pin Block Pinout.
Viitattu 9.5.2916.
https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout

/4/ DHT11 digitaalinen lämpötila- ja ilmankosteusanturi.
Viitattu 9.5.2016.
http://ihmevekotin.fi/product/297_dht11-digitaalinen-l%C3%A4mp%C3%B6tila-ja-ilmankosteusanturi

/5/ DHT11 Humidity & Temperature Sensor datasheet.
Viitattu 9.5.2016.
http://www.micropik.com/PDF/dht11.pdf

/6/ Open Handset Alliance FAQ.
Viitattu 10.5.2016.
http://www.openhandsetalliance.com/oha_faq.html

/7/ Number of apps available in leading app stores.
Viitattu 10.5.2016.
http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

/8/ Android Studio Release Notes.
Viitattu 10.5.2016.
http://developer.android.com/tools/revisions/studio.html

/9/ Security of Android.
Viitattu 10.5.2016.
https://source.android.com/security/

/10/ Android Architecture.
Viitattu 10.5.2016.
http://www.eazytutz.com/android/android-architecture/

/11/ What is JSON?
Viitattu 11.5.2016.
http://developers.squarespace.com/what-is-json/

/12/ The Apache Tomcat software.
Viitattu 11.5.2016.
http://tomcat.apache.org/index.html

/13/ Devatus Oy.
Viitattu 12.5.2016.
http://devatus.fi/

/14/ Finanssiesittely Maintscope Oy.
Viitattu 13.5.2016.
https://kasvuopen.fi/blogi/finalistiesittely-maintscope-visuaalista-kaynnissapidon-hallintaa

/15/ maintBox-verkkosivut.
Viitattu 13.5.2016.
https://www.morgan.fi/wp-content/uploads/2015/05/MaintBox-verkkosivut.jpg

/16/ RuuviTag, Open-Source Bluetooth 4.2 Sensor Beacon.
Viitattu 12.2016.
http://ruuvitag.com/

/17/ Learn REST.
Viitattu 13.5.2016.
http://rest.elkstein.org/

/18/ C Language - Overview.
Viitattu 13.5.2016.
http://www.tutorialspoint.com/cprogramming/c_overview.htm

/19/ Java.
Viitattu 13.5.2016.
http://www.webopedia.com/TERM/J/Java.html

/20/ Understanding the different types of BLE Beacons.
Viitattu 22.5.2016.
https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/

/21/ iBeacon with Raspberry Pi.
Viitattu 23.5.2016.
http://www.theregister.co.uk/2013/11/29/feature_diy_apple_ibeacons/

/22/ Manifest.permission.
Viitattu 27.5.2016.
https://developer.android.com/reference/android/Manifest.permission.html