IoT-laitteen data yhdyskäytävän kautta pilvipalveluun

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeen Ammattikorkeakoulu Riihimäki, Tietotekniikka

2012-2017, 2017

Riku Lehtinen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietotekniikka
HAMK Riihimäki

Tekijä Riku Lehtinen **Vuosi** 2017

Työn nimi IoT-laitteen data yhdyskäytävän kautta pilvipalveluun

TIIVISTELMÄ

Opinnäytetyöni tarkoitus oli selventää Hämeen Ammattikorkeakoulun Älykkäät palvelut -tutkimusyksikölle kuinka saada Bluetooth Low Energy -laitteen keräämä data Internetiin yhdyskäytävän avulla. Pyrin saamaan yhdyskäytävänä toimivan Rasperry Pi 3 -laitteen skannaamaan iBeacon-laitteita Bluetoothin avulla ympäristössään ja lähettämään niiden tietoja eteenpäin Internetin avulla valitsemaani pilvipalveluun.

Selvitin iBeacon ja Raspberry Pi 3 -laitteiden yleistä toimintaa. Työssä tutustutaan syvällisemmin Microsoft Azuren eri palveluihin ja erityisesti Microsoft Azure IoT Hubin toimintaan. IoT on alati kasvava alalaji tietotekniikassa, joten sen tutkimuksille ja osaajille on alati lisääntyvä kysyntä ja siksi aihealue on uusi sekä kiinnostava.

Työssäni onnistuin löytämään ratkaisun iBeacon -laitteen tietojen keräämiseen sekä lähetykseen ”Node.js” -menetelmän avulla. Työtä voisi kehittää tulevaisuutta ajatellen tutustumalla syvällisemmin Microsoft Azuren tarjoamiin mahdollisuuksiin rakentaa kokonainen toimiva IoT-ratkaisu aina tiedon visualisoinnista sen talletukseen.

Avainsanat iBeacon, IoT, Microsoft Azure, yhdyskäytävä

Sivut 20 sivua, joista liitteitä 0 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information Technology
Riihimäki

Author Riku Lehtinen **Year** 2017

Subject Data from IoT device to cloud service

ABSTRACT

The purpose of my thesis work was to find a working solution on how to get data from a Bluetooth Low Energy device to a cloud service through a gateway. This thesis was conducted for Häme University of Applied Sciences and their Intelligent Services – research unit. I tried to make a Raspberry Pi 3 device act as a gateway and to scan Internet of Things (IoT) devices called iBeacons and send their telemetry data to a cloud service using an Internet connection.

I examined in this project the overall functions of iBeacon devices and Raspberry Pi 3 -computer. Here in my thesis the focus is on the different services of Microsoft Azure and particularly it's Azure IoT Hub. Internet of Things is a new and growing subgenre in information technology and which's why its experts have a growing demand in the IT field.

In the thesis I managed to find a solution for collecting data information from iBeacon by using the "Node.js" method. The project could be further developed by focusing more on the Microsoft Azures services and now this work acts as a base where a proper IoT-architecture could be built in the future.

Keywords iBeacon, IoT, Microsoft Azure, gateway

Pages 20 pages including appendices 0 pages

SISÄLLYS

1 JOHDANTO

IoT, eli Internet of Things, tarkoittaa suomeksi esineiden tai asioiden Internetiä. Se on käsitteenä tuttu jo 1980-luvulta, vaikka sille annettiin nimi vasta huomattavasti myöhemmin. Ensimmäisenä IoT-laitteena voidaan pitää “Cola-laitetta”, johon Yhdysvaltalaisen yliopiston opiskelijat ohjelmoivat tavan tarkistaa onko laitteessa kylmää juomaa jäljellä ja liittivät laitteen Internetiin, josta pystyi etänä tarkistamaan juomatilanteen (IoT-Agenda, 2016).

Käytännössä IoT:lla tarkoitetaan erilaisten laitteiden ja esimerkiksi rakennusten yhdistymistä Internetiin, jolloin niitä voidaan ohjata ja mitata etäältä verkon ylitse. IoT:n suosio varsinkin teollisuudessa on hurjassa kasvussa ja eri tutkimusten mukaan vuoteen 2020 mennessä maailmassa saattaa olla 20-50 miljardia erilaista laitetta liitettynä verkkoon. Vuonna 2016 arvio tuosta luvusta on noin 6,4 miljardia laitetta (IEEE Spectrum, 2016). IoT:n avulla jokainen laite on mahdollista yksilöidä ja yhdistää se Internetiin osaksi isompaa kokonaisuutta, mikä houkuttelee varsinkin yrityksiä panostamaan aiheeseen nyt ja tulevaisuudessa.

Työssäni tutustun IoT-laitteeseen nimeltä iBeacon. Pyrin saamaan sen ilmoitusviestit käsiini yhdyskäytävälle Bluetoothin avulla ja tämän jälkeen lähettämään tiedot eteenpäin WLAN-yhteydellä Microsoft Azure IoT Hub -palveluun, josta iBeaconeiden lähettämiin viesteihin pääsee käsiksi mistä tahansa Internet-yhteyden avulla.

2 MICROSOFT AZURE

Mikä on Azure?

Microsoft julkisti Windows Azuren vuonna 2010. Nimi kuitenkin muutettiin sittemmin vuonna 2014 Microsoft Azureksi (SearchCloudComputing, 2016). Azure tarjoaa monia erilaisia pilvipalveluita tietojen tallennukseen, analytiikkaan, laskentaan ja verkkoihin liittyviin asioihin. Azure kilpailee muun muassa Amazon Web Services (AWS)-palvelun ja Google Cloud Platform-palvelun kanssa markkinoista. Se käyttää useiden muiden pilvipalvelutarjoajien tapaan ”pay-as-you-go”-hinnoittelutapaa, jossa laskutetaan pääosin käytön runsauden mukaan. Azuren palvelut on jaettu yhteentoista eri pääkategoriaan, ja niistä tulen keskittymään ”Internet of Things”-kategorian alla olevaan Azure IoT Hubiin (SearchCloudComputing, 2016).

Microsoft Azure tarjoaa IoT:n ohella palveluita esimerkiksi virtuaalipalvelimien ylläpidossa. Azurella voi luoda virtuaalipalvelimen ainakin Linuxille, Windows Serverille, SQL Serverille, Oraclelle sekä SAP:ille (Azure.Microsoft.Com, 2017). Näillä virtuaalipalvelimilla voi tehdä esimerkiksi testaus- ja kehitystyötä, ylläpitää sovelluksia pilvipalvelussa sekä liittää virtuaalipalvelimia yrityksen verkkoon lisäkapasiteetin saamiseksi. Azurella voi kehittää sovelluksia ”Azure App Service”-ohjelman avulla, tallentaa tietoja ”Azure Storage”-palveluun, hyödyntää ”Azure Virtual Network”-palvelua virtuaalipalvelimien ylläpidossa turvallisessa ja eristetyssä verkkoympäristössä sekä monia muita palveluita, jotka tekevät Microsoft Azuresta alan toiseksi suurimman palveluntarjoajan heti Amazon Web Services -pilvipalvelun jälkeen (crn.com, 2017).

Microsoft Azure IoT-palveluntarjoajana

Azure IoT Hub on varta vasten IoT:lle varten luotu palvelu, johon voi liittää tuhansia eri laitteita. Niin ”device-to-cloud” kuin ”cloud-to-device” -viestit ovat mahdollisia, joten IoT Hubilla voi sekä tarkkailla laitteiden tilaa että lähettää niille komentoja. Microsoft lupaa IoT Hubin olevan helppokäyttöinen sekä turvallinen IoT-laitteiden kanssa ja sen ominaisuuksia lueteltu alla olevassa luettelossa (Microsoft.com, 2016):

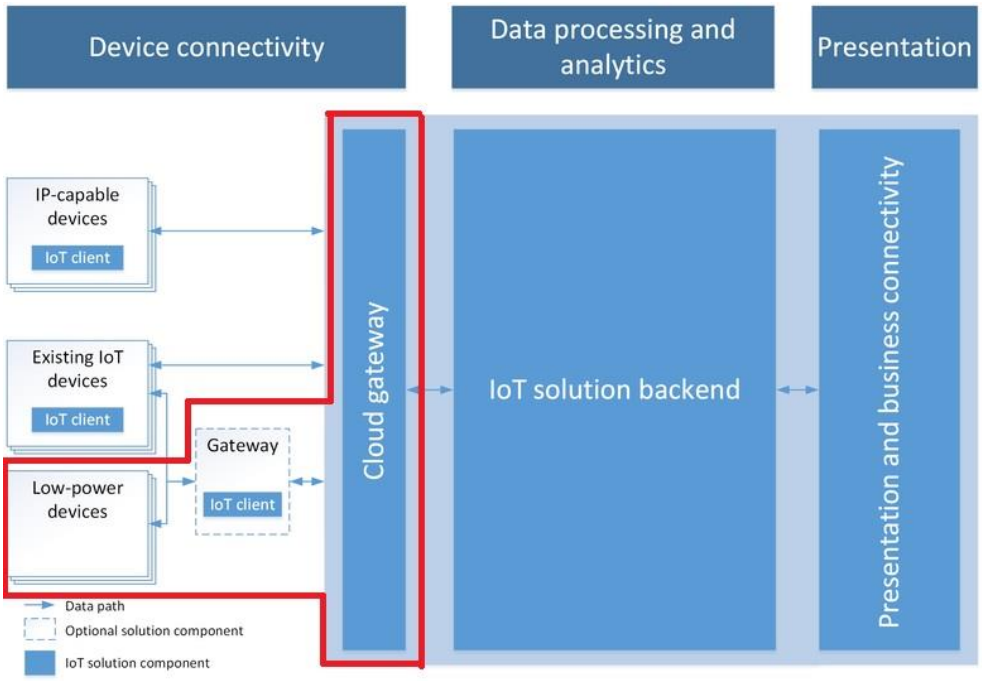
- Useita vaihtoehtoja kommunikaatiolle laitteen ja pilven välillä, kuten yksisuuntainen viestitys, tiedostojen siirto ja pyyntö-vastaus menetelmät
- Sisäänrakennettu viestien reititys muihin Azuren palveluihin
- Tallennustila laitteiden metadatoille sekä synkronosoidut tilatiedot

- Turvallinen viestintä ja kulunvalvonta laitekohtaisten turva-avaimien ansiosta
- Kattava seuranta laitteen yhteyden ja hallinnan tapahtumista
- Mukana laitekirjastot suosituimmista ohjelmointikielistä ja alustoista

Microsoft Azure IoT Hubissa turvallisuus ottaa etusijan kaikessa toiminnassa. Viestintäpolku IoT-laitteen sekä IoT Hubin välillä on turvattu sovelluskerroksella (Application layer) OSI-mallissa. Koska järjestelmätason valtuutus ja todennus pohjautuvat aina yksittäiseen laitteeseen, on käyttöoikeudet helppo ottaa pois heti jos jotain epäilyttävää on havaittavissa.

IoT ratkaisuarkkitehtuuri

Microsoft ehdottaa verkkosivuillaan kuvan 1 mukaista arkkitehtuuria tyypillisille Azure IoT -ratkaisuille.



Kuva 1 IoT-arkkitehtuuri tyypillisille Azure IoT-ratkaisuille (Microsoft.com, 2017). Kuvaan on punaisella korostettu osa-alueet, jotka koskevat tätä työtä.

Kyseisessä arkkitehtuurissa IoT-laitteet keräävät dataa, jonka ne lähettävät eteenpäin. Mikäli IoT-laite on ns. ”low-power device”, tarvitsee se väliyhdyskäytävän vastaanottamaan ja lähettämään datan eteenpäin pilviyhdyskäytävälle. Usein pienen virrankulutuksen laitteet lähettävät tietojaan Bluetoothin avulla eivätkä ne sisällä mahdollisuutta yhdistää laitetta suoraan Internetiin, vaan toimintoon tarvitaan jokin väliyhdyskäytävä. Muun muassa sykemittarit synkronoivat tietonsa Bluetoothin avulla sovellukseen älypuhelimessa, josta on pääsy Internetiin. Tässä tapauksessa älypuhelin toimii väliyhdyskäytävänä prosessissa. IoT-laitteilla on yleensä muutamia

yhteisiä piirteitä eri aloista riippuen: monet ovat heikkotehoisia, jotta virtaa riittää pidemmäksi ajaksi sekä monet voivat olla hankalissa paikoissa fyysisesti joihin ei pääse käsiksi helpolla.

Pilviyhdyskäytävä tekee datan saavutettavaksi seuraavalla vaiheelle, jota kutsutaan nimellä ”IoT solution backend” eli suomeksi ”IoT ratkaisuosa”. Sillä data voidaan prosessoida, suodattaa, tallettaa ja ohjata eteenpäin muille palveluille. ”Presentation” eli esitysosassa data on mahdollista saada näkyviin analysoitavaksi. Tämä data voi olla joko historiallista tai lähes reaaliaikaista riippuen arkkitehtuurista. ”Business connectivity”- tarkoittaa datan yhdistämistä tukemaan liiketoimintaa, helpottamaan valvomista ja auttamaan ratkaisemaan ongelmia. Jokaisen IoT-ratkaisun tulee olla aina luotettava ja turvallinen, mikäli se on yhdistetty tärkeäksi osaksi liiketoimintaa (docs.microsoft.com, 2017).

Esimerkkinä tämän arkkitehtuurin toiminnasta voidaan antaa pumppaamon toiminnalla. Pienet IoT-laitteet voivat toimia sensoreina pumpuissa ja ratkaisuosalla on mahdollisuus ennustaa historiallisesta datasta milloin kukin pumppu on huollon tarpeessa. Ja koska useimmiten myös ”cloud-to-device”-välinen viestitys on mahdollista, ratkaisuosa saattaa pyytää muita pumppuja automaattisesti uudelleenohjaamaan virtauksia muihin osiin, jotta huollettavaa pumppua on mahdollista korjata (docs.microsoft.com, 2017).

Microsoft tarjoaa kokovaltaista palvelua erilaisille IoT-palveluille IoT Suite-nimisen työkalun muodossa. IoT Suiten ominaisuuksia voisi olla muun muassa etävalvontajärjestelmä myyntiautomaateille, jolloin etäältä on mahdollista nähdä koska automaatti tarvitsee täyttöä. Toinen esimerkki on yllä olevan pumppaamon tapainen ennakoiva huolto, jolloin historiallisen datan perusteella ratkaisuosa huomaa milloin pumppu on huollon tarpeessa ja näin vältetään ylimääräisiä pysähdyksiä.

Laitehallinta IoT-yrityskäytössä

Microsoft Azure IoT Hub tuo yrityskäytössä mukanaan myös muutamia haasteita laitteiden hallinnan kanssa. Parhaimmillaan yrityksellä voi olla miljoonia IoT-laitteita hallittavanaan ja tämän vuoksi he tarvitsevat yksinkertaiset ja luotettavat työkalut laitteiden järkevään hallitsemiseen. Microsoft on ottanut tämän huomioon kehittäessään IoT Hubia ja se on pyrkinyt kehittämään laajan valikoiman erilaisia vaihtoehtoja erilaisten laitteiden hallitsemisessa.

Jotta laajaa IoT-laitemäärää voi hallita mahdollisimman pienimääräisellä henkilöstöllä, tarvitaan automaatiota auttamaan tekemään yksinkertaisimmat ja päivittäiset tehtävät. Henkilöstön tulisi pystyä keskittymään hallitsemaan isoja joukkoja laitteita kerrallaan ja keskittymään tiettyihin laitteisiin vain jos tulee tietoa ongelmista. Laitteet eivät lisäksi suinkaan aina

ole samanlaisia vaihdellen kaikista yksinkertaisimmista yhden prosessin piirilevyistä jopa täysin toimiviin tietokoneisiin. Näin ollen laitehallinnan yhteensopivuus on ensiarvoisen tärkeää. IoT-ympäristö on lähes aina dynaaminen ja se muuttuu vähän väliä, joten hallinnan tulee olla luotettavaa eivätkä pysähtymisajat saa olla liian pitkiä. Pahimmassa tapauksessa jopa työntekijöiden turvallisuus voi olla vaarassa mikäli laitehallinta pettää, puhumattakaan taloudellisista tappioista yritykselle.

Itse IoT-laitteella on elämällään viisi vaihetta Azure IoT -maailmassa. Ensimmäisessä vaiheessa eli suunnittelussa kehitetään joukolle laitteita niin kutsuttu metadatasuunnitelma, jotta laitteita voi hallita yhdessä. Seuraavassa vaiheessa, asennuksessa, pyritään turvallisesti asentaa uudet laitteet IoT Hubiin ja samalla ottaa laitteiden ominaisuudet käyttöön. Tämän jälkeen kolmannessa eli konfigurointivaiheessa pidetään huolta mahdollisista päivityksistä yms. muutoksista, jotta turvallisuus pysyy ajan tasalla. Neljännessä vaiheessa monitoroidaan laitteiden tilaa mm. hälytystoimintojen avulla ja lopuksi viimeisessä vaiheessa poistetaan laite käytöstä vian tai normaalin päivityssyklin jälkeen (docs.microsoft.com, 2017).

3 IOT JA IBEACON

iBeacon tutuksi

Applen vuonna 2013 kehittämän ja julkaiseman iBeacon-protokollan ympärille useat eri valmistajat ovat kehitelleet omia laitteita, eli niin kutsuttuja ”beaconeita”. Itse iBeacon-laitteen idea on hyvin yksinkertainen: se lähettää Bluetoothin avulla ympäriinsä tiedon itsestään, ja vastaanottaja, eli esimerkiksi älypuhelin, huomaa signaalin. Kaikki tapahtumat tämän jälkeen riippuu sovelluksesta, jossa iBeaconia halutaan hyödyntää. Signaalin voimakkuudesta eli iBeaconin etäisyydestä riippuen älypuhelinta voi sovelluksen avulla käskeä tekemään monia erilaisia asioita. Esimerkiksi valojen kytkeminen sammuksiin ja päälle, tai erilaisten viestien näkyminen puhelimella, onnistuvat iBeaconeiden avulla kun määritetään etäisyys puhelimen ja tietyn iBeaconin välillä, jolloin tietyn asian halutaan tapahtuvan.

iBeaconit lähettävät signaalia Bluetooth Low Energy -metodilla, joka tuli tutuksi Bluetooth 4.0 standardin yhteydessä. Bluetooth Low Energy (BLE) eroaa normaaliin Bluetoothiin virrankulutuksessa, kuten nimestäkin voi päätellä. Koska iBeaconit lähettävät hyvin yksinkertaisia viestejä, BLE-yhteys riittää hyvin ja näin virrankulutus on huomattavasti pienempi kuin normaalilla Bluetooth-yhteydellä. Tämä edistää pariston kestoa laitteessa ja monien eri iBeacon-valmistajien laitteet saattavat kestää järkevällä virrankulutuksella jopa useamman vuoden paristoa vaihtamatta. Normaalin iBeaconin kantama on noin 70 metriä ja ns. pitkän matkan iBeaconin kantama voi yltää jopa 450 metriin, riippuen lähetystehosta, maastosta ja esteistä (Wikipedia, 2016).

Perusidealtaan iBeacon lähettää signaalissaan kolme tietoa itsestään, eli UUID:n (Universially Unique Identifier), Major-arvon ja Minor-arvon (Kontakt.io, 2016). Jokaista näistä arvoista voi käyttäjä muokata haluamallaan tavalla. Jos ja kun kokonaisuuteen käytetään useampaa iBeaconia, kannattaa UUID-arvo valita samaksi jokaiselle. Major-arvolla saadaan iBeaconit eriteltyä ryhmiin, ja minor-arvolla ne saa yksilöistettyä. Alla on havainnollistava kuva kauppakeskusketjusta, joka käyttää iBeaconeita (Estimote, 2016):

Store location		San Francisco	Paris	London
UUID		D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C		
Major		1	2	3
Minor	Clothing	10	10	10
	Housewares	20	20	20
	Automotive	30	30	30

Kuva 2. Kauppakeskusketjun määrittelemät iBeacon asetukset (Estimote, 2016).

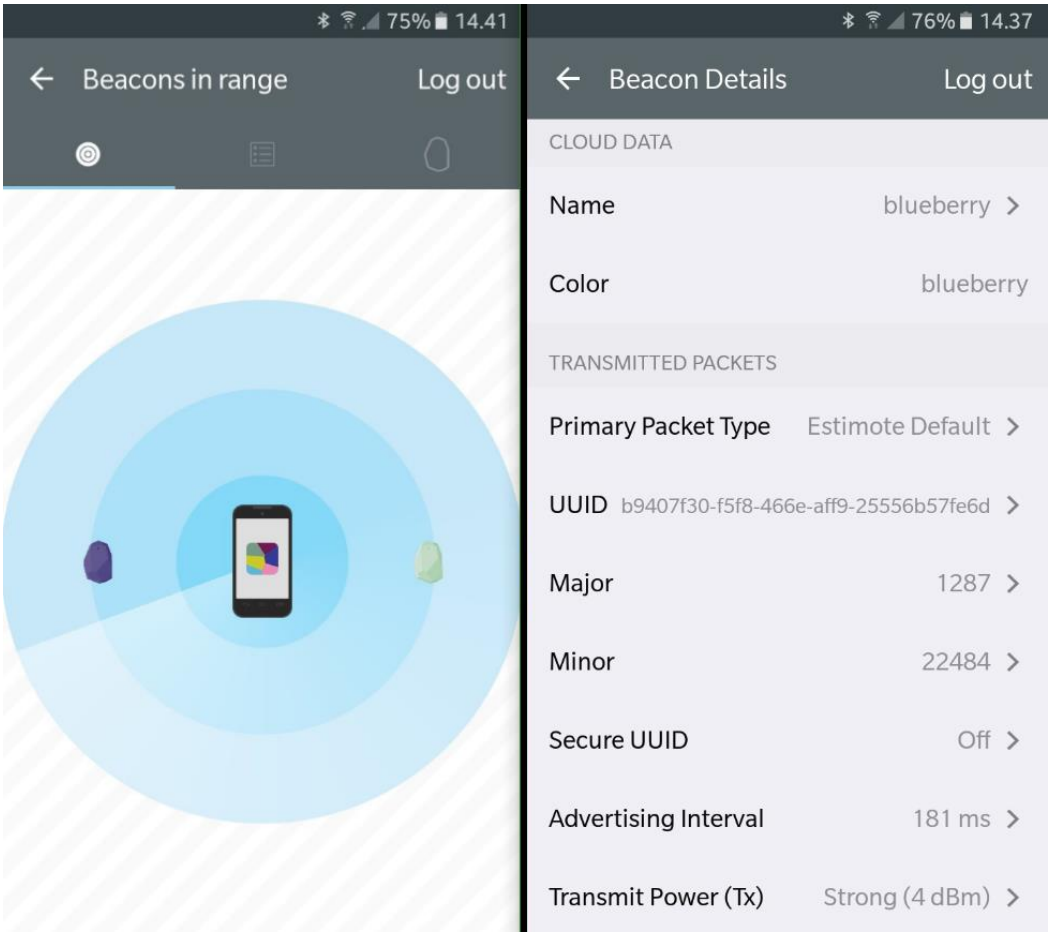
Estimote iBeacon

Käytän työssäni Estimote-valmistajan iBeaconeita (kuva 3). Niissä on sisäänrakennettuna lämpötilasensori sekä liikesensori, eli ne lähettävät normaalin signaalin lisäksi tietoa myös lämpötilasta sekä siitä, onko laite liikkeessä vai ei.

Kuva 3 Estimote iBeaconeita (Estimote, 2016)

Estimote iBeaconin haltuunotto ja asetusten määritteleminen on helppoa Estimoten omalla iOS/Android sovelluksella. Kun on määritelty mille käyttäjätilille iBeacon kuuluu, pääsee sovellukseen kirjautumalla tarkastelemaan lähellä olevia iBeaconeita sekä muuttamaan niiden asetuksia, mikäli siihen on valtuudet. Älypuhelimessa täytyy olla Bluetooth käytössä sekä sen pitää käyttää vähintään Bluetooth 4.0 standardia, jotta BLE-signaalia lähettävät iBeaconit näkyvät. Sovelluksella pystyy lisäksi muuttamaan älypuhelimen signaalia lähettäväksi iBeaconiksi, mikäli sellaisella on tarvetta.

Itse pidin iBeaconeiden asetukset lähes ennallaan, sillä niillä ei itse työn kannalta ole juurikaan merkitystä. Laitoin virransäästöominaisuudet päälle laitteisiin, jotta niissä riitti virtaa koko projektin ajaksi. Alla on kuva hallitsemastani iBeaconista Estimoten Android-sovelluksen kautta:



Kuva 4 Estimote-sovelluksessa pääsee muuttamaan iBeaconin asetuksia

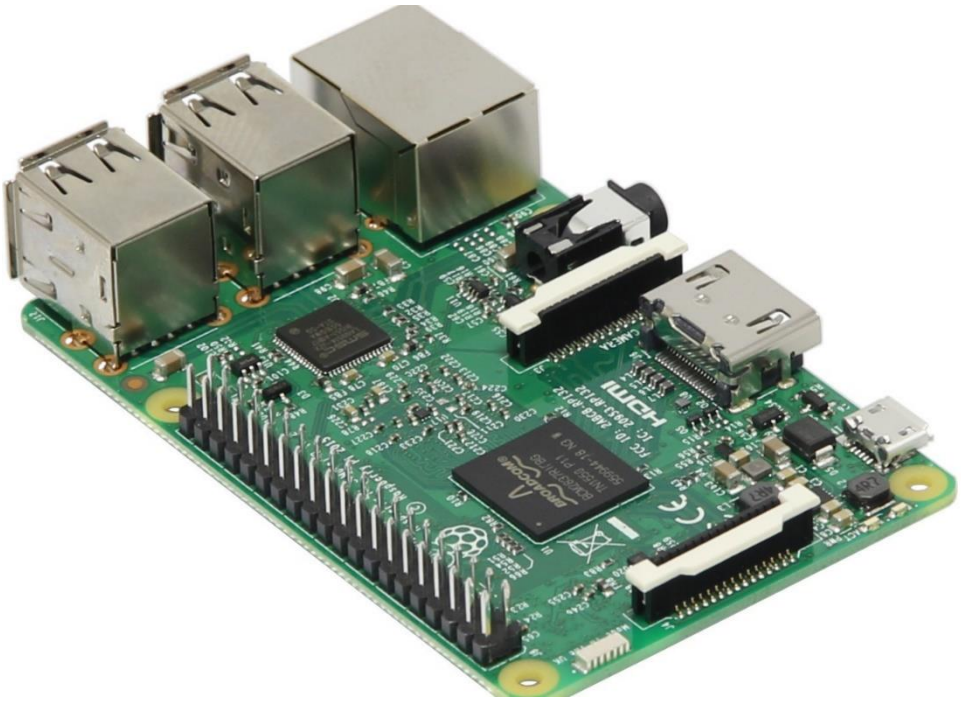
4 YHDYSKÄYTÄVÄ JA SEN KÄYTTÖJÄRJESTELMÄ

Opinnäytetyötäni varten olisin voinut valita yhdyskäytäväksi lähes minkä tahansa tietokoneen, johon saa asennettua käyttöjärjestelmän. Esimerkiksi Arduino ja Intel Compute Stick olivat vaihtoehtoina, mutta päädyin Raspberry Pi 3:seen sen edukkuuden ja käytännöllisyyden vuoksi.

Käyttöjärjestelmää valittaessa yhdyskäytävälle, aivan kuten normaaleille palvelimillekin, on hyvä ottaa huomioon mm. sen toiminnallisuus halutuissa tehtävissä sekä sen luotettavuus ja tietoturvallisuus.

Raspberry Pi 3

Raspberry Pi 3 on yhden piirilevyn tietokone, ja se julkaistiin paranneltuna versiona edeltäjistään helmikuussa 2016. Sitä mainostetaan ”pankkikortin



Kuva 5 Raspberry Pi 3 - pankkikortin kokoinen tietokone (Verkkokauppa.com, 2016)

kokoisena tietokoneena” (kuva 5). Ilman koteloa sen mitat ovat 85,6 x 53,98 x 17mm, eli ilman erilaisille liitännöille vaadittavia paikkoja (esim. USB ja HDMI) se lähestulkoon täyttäisi mainospuheensa. Raspberry Pi 3 sisältää kaikki tarvittavat liitännät ollakseen oikea tietokone minikoossa. Käyttöjärjestelmän pystyy asentamaan microSD-kortille, joka laitetaan piirilevyssä sijaitsevaan kortinlukijaan.

Yksi tärkeä kriteeri Raspberry Pi 3:sen valinnaksi yhdyskäytäväksi projektiin oli sen sisäänrakennettu Bluetooth-piiri, joka tukee BLE:tä, jonka kautta iBeaconit kommunikoivat. Edelliset Raspberry Pi -versiot olisivat vaatineet lisävarusteen BLE:lle. Lisäksi siinä on tuki langattomalle Wi-Fi-yhteydelle normaalin RJ45-verkkokaapelipaikan lisäksi, joten Internet-yhteys on helppo muodostaa kumpaa kautta tahansa.

Käyttöjärjestelmä

Käyttöjärjestelmän valinta yhdyskäytävälle on yksi tärkeimmistä asioista työn etenemisen kannalta. Katsoin järkevämmiksi vaihtoehdoiksi kaksi ehdokasta: Linux/Debian-pohjainen Raspbian Jessie tai Windows 10 IoT Core. Molemmat käyttöjärjestelmät ovat eritoten optimoitu pienemmille laitteille, jotka voivat olla myös näytöttömiä. Molemmat käyttöjärjestelmät ovat saatavina ilmaiseksi.

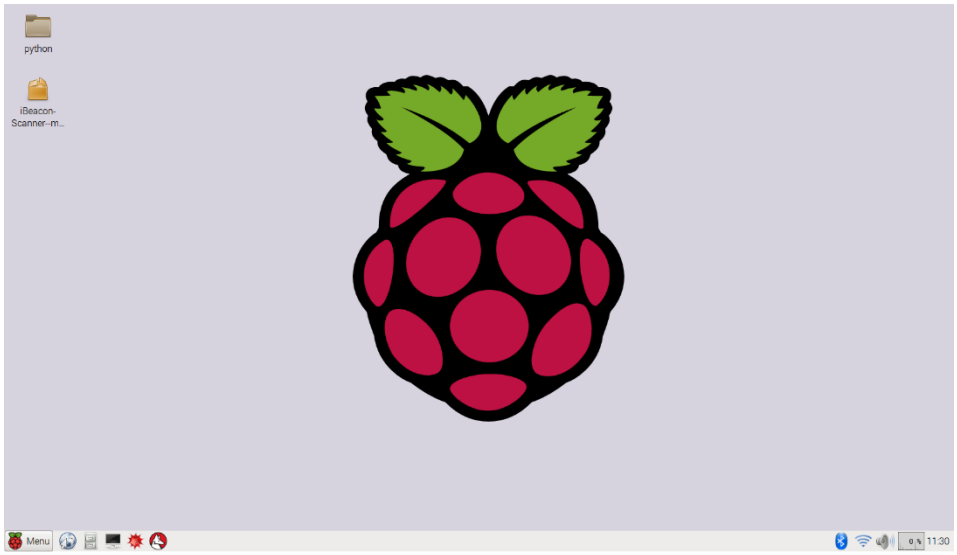
Windows 10 IoT Core on Windows 10 pohjainen järjestelmä, ja se käyttää laajaa Universal Windows Platform (UWP)-ohjelmointirajapintaa ratkaisujen tekemiseen (Microsoft.com, 2016). Käyttöjärjestelmä takaa tutun Windows kokemuksen ja sille voi kehittää sovelluksia Visual Studiolla. Windows 10 IoT Coreen on yhdistetty ominaisuuksia Microsoft Azureen yhdistämiseksi.

Raspbian Jessie-käyttöjärjestelmä perustuu Debian Linuxiin, joten se on käyttökokemuksena erilainen kuin Windows-pohjainen järjestelmä. Sana Jessie-viittaa Raspbianin versionumeroon ja Jessie on tällä hetkellä uusin versio Raspbianista. Toisinkuin edeltäjänsä, Jessie latautuu käynnistyessään suoraan graafiseen käyttöliittymään, mikä on nykyajan standardi lähes kaikille käyttöjärjestelmille. Raspbian on varta vasten kehitelty Raspberry Pi:tä varten, joten laitteen ominaisuudet on otettu hyvin huomioon käyttöjärjestelmää suunniteltaessa ja Raspbian on täten helppo asentaa ja käyttää Raspberryllä.

Näistä kahdesta vaihtoehdosta käyttöjärjestelmänä tarrauduin loppujen lopuksi Raspbian Jessieen vanhojen Linux-kokemuksien pohjalta. Lisäksi Windows IoT Coressa oli työtä aloittaessa ongelmia WiFi-yhteyden kanssa perustuen muiden käyttäjien kokemuksiin kommentteja lukiessa. Nämä ongelmat voivat hyvin olla jo korjattu ja työn voisi yhtä lailla tehdä perustuen Windows IoT Coreen.

Raspbian asennus ja alkutoimet

Yksi keino asentaa Raspbian Raspberry Pi:lle on lataamalla .img-tiedosto Windows tietokoneelle, jossa on paikka microSD-kortille. Tiedoston saa suoraan Raspberry Pi:n kotisivuilta. Tämän jälkeen Internetistä saatavan Win32DiskImager-ohjelman avulla levykuvan voi asentaa SD-kortille. Asennuksen jälkeen kortti tulee asettaa Raspberry:n SD-korttipaikkaan. Virtojen kytkemisen jälkeen Raspbian latautuu suoraan työpöydälle (kuva 6).



Kuva 6 Raspbian Jessien työpöytänäkymä

Jotta laitteen saa liitettyä langattomaan Internetiin, ensimmäisellä käyttökerralla Raspberry tulee yhdistää joko tietokoneen näyttöön tai televisioon HDMI-piuhalla. WiFi:in kirjautumisen jälkeen Raspbian muistaa salasanan ja näin ollen sitä voi seuraavalla kerralla käyttää suoraan esimerkiksi Windows-tietokoneen ”Remote Desktop Connectionin” tai Putty-ohjelman avulla, kunhan tietää laitteen IP-osoitteen. Mikäli HDMI-kaapelia ei ole saatavilla, voi Raspberryn yhdistää kotiverkkoon RJ45-internetkaapelilla jolloin laitteen IP-osoitteen selvittämisen jälkeen pystyy siihen kirjautumaan sisään etäältä.

Raspbianissa on SSH-palvelu jo valmiiksi asennettuna, mutta se täytyy asettaa päälle komennolla *sudo service ssh start*. Tämän jälkeen käyttöjärjestelemään pääsee käsiksi etäyhteydellä esimerkiksi Windowsille tehdyn Putty-ohjelman avulla. Windowsin Remote Desktop Connectionin saa toimimaan asentamalla Raspbianiin xrdp-laajennus komennolla *apt-get install xrdp*, mikäli haluaa käyttää etänä järjestelmän muitakin osa-alueita kuin pelkästään komentoriviä.

5 IBEACON-TIEDOT YHDYSKÄYTÄVÄLLE JA PILVIPALVELUUN

Kuten johdannossa on mainittu, opinnäytetyössäni on tarkoituksena selvittää Hämeen Ammattikorkeakoulun Älykkäät palvelut -tutkimusyksikölle kuinka saada iBeacon-laitetiedot yhdyskäytävän kautta pilvipalveluun. Tässä kappaleessa tutkitaan asiaa käytännön tasolla.

IoT-laitteiden skannaus yhdyskäytävällä

Tutkittuani asiaa tuli selväksi, että on ainakin kaksi toisistaan eroavaa ratkaisua skannata ”Bluetooth low energy” -laitteita yhdyskäytävällä, kuten esimerkiksi iBeaconeita. Internetistä löytyvien ohjeiden valossa suositumpi keino on Python3-koodin avulla (Instructables, 2017). Toisessa tavassa käytetään Linuxille rakennettua ”node.js”-kirjastoa. Molempiin ratkaisuihin tarvitaan kuitenkin Linuxin virallista Bluetooth-protokollakirjastoa nimeltä BlueZ. Sen asennus onnistuu alla olevalla komennolla Raspbianin komentorivillä:

sudo apt-get install bluetooth bluez libbluetooth-dev libudev-dev

Asennuksen jälkeen skannaaminen Pythonin avulla tapahtuu seuraavasti asentamalla Python3-moduuli Blueziin sekä kopioimalla internetistä vapaassa käytössä olevan lähdekoodin komennoilla:

sudo apt-get install python-dev python3-dev python3-setuptools python3-pip
sudo pip-3.2 install pybluez –
git clone https://github.com/flyinactor91/RasPi-iBeacons

Tämän jälkeen RasPi-iBeacons kansiosta löytyy blescan.py tiedosto, jonka suorittamalla Pythonin avulla skannaus onnistuu komennolla *sudo python3 blescan.py*. Kuvassa 7 on esitelty tavan tulokset.

```
root@raspberrypi:/home/pi/skanni/RasPi-iBeacons# sudo python3 blescan.py
ble thread started
----------
e5:5e:cb:3e:aa:59,b9407f30f5f8466eaff925556b57fe6d,aa59,cb3e,c4
c4:46:57:d4:05:07,b9407f30f5f8466eaff925556b57fe6d,0507,57d4,c4
----------
e5:5e:cb:3e:aa:59,b9407f30f5f8466eaff925556b57fe6d,aa59,cb3e,c4
c4:46:57:d4:05:07,b9407f30f5f8466eaff925556b57fe6d,0507,57d4,c4
----------
```

Kuva 7 Pythonin avulla suoritetun iBeacon-skannauksen tulokset

Minulla oli tuolla hetkellä kaksi iBeaconia toiminnassa lähistöllä, ja niiden UUID-arvot näkyvät kohdassa ”b9407..”, mutta esimerkiksi Major/Minor-arvoja ei näy tuloksissa lainkaan. Ensimmäisillä kerroilla en saanut tuloksia ollenkaan näkyviin, ja kompastuskivenä pitkän pohdinnan jälkeen oli se, ettei Raspberryn Bluetooth ollut ”skannaavassa” tilassa. Sen sai päälle kytkeytymällä Bluetoothiin komennolla *bluetoothctl* sekä tämän jälkeen asettamalla komennon *scan on*.

Toinen tapa skannata iBeaconeita yhdyskäytävällä on ”Node.js”- ja ”npm”-kirjastojen avulla. Node.js ja samalla Node Package Managerin (npm) asentaminen Debian ja Ubuntu-pohjaisille Linux-järjestelmille onnistuu seuraavilla komennoilla (nodejs.org, 2016):

curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install -y nodejs

Node-bleacon kirjaston avulla pystyy tehdä, skannata sekä konfiguroida iBeaconeita (https://github.com/sandeepmistry, 2016). Node-bleacon-kirjaston pystyy asentamaan komennolla *npm install bleacon*. Asennuksen jälkeen kansiosta löytyy test.js tiedosto, jossa on valmiina peruskomennot iBeaconeiden skannaukselle sekä sen avulla voi muuttaa itse Raspberryn iBeaconiksi. Muutin test.js-tiedoston alla olevan kuvan 8 mukaiseen muotoon, jolloin sain skannauksen toimimaan.

```
var Bleacon = require('./index');

Bleacon.on('discover', function(bleacon) {
  console.log('bleacon found: ' + JSON.stringify(bleacon));
});
//Bleacon.startScanning('b9407f30f5f8466eaff925556b57fe6d',1251,2133);
Bleacon.startScanning();
```

Kuva 8 Node-bleacon kirjaston test.js-tiedosto kuvassa muutettuna iBeaconeiden skannausta varten

Bleacon.startScanning(); komennolla on mahdollista skannata kaikki mahdolliset lähistöllä olevat iBeaconit, ja komennon sisälle on mahdollista asettaa parametrejä mm. siten, että saa vain tietyn UUID:n ja tietyn Major/Minor-arvojen iBeaconit näkyviin. Kuvassa 9 on esitelty skannauksen tulos. Tuloksista saa enemmän tietoa kuin Python-metodilla saaduista

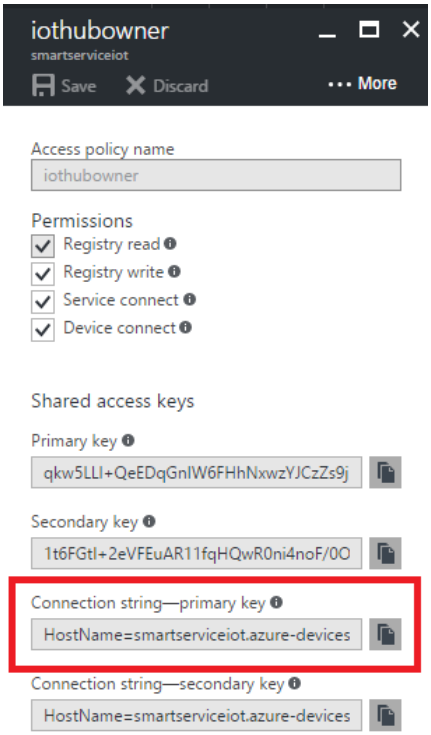
```
root@raspberrypi:/home/pi/skanni/RasPi-iBeacons/node_modules/bleacon# node test.
js
bleacon found: {"uuid":"b9407f30f5f8466eaff925556b57fe6d","major":1287,"minor":2
2484,"measuredPower":-60,"rssi":-59,"accuracy":0.9397676019578316,"proximity":"n
ear"}
bleacon found: {"uuid":"b9407f30f5f8466eaff925556b57fe6d","major":1287,"minor":2
2484,"measuredPower":-60,"rssi":-61,"accuracy":1.0640928649984154,"proximity":"n
ear"}
bleacon found: {"uuid":"b9407f30f5f8466eaff925556b57fe6d","major":43609,"minor":
52030,"measuredPower":-60,"rssi":-34,"accuracy":0.1988530195534621,"proximity":"
immediate"}
bleacon found: {"uuid":"b9407f30f5f8466eaff925556b57fe6d","major":1287,"minor":2
2484,"measuredPower":-60,"rssi":-55,"accuracy":0.7329972482293305,"proximity":"n
ear"}
```

Kuva 9 Node-bleacon kirjaston avulla saatuja iBeaconeiden skannaustuloksia

skannaustuloksista. Niistä näkee mm. iBeaconeiden Major/Minor-arvot sekä sanallisesti kuinka lähellä iBeaconit sijaitsevat yhdyskäytävää eli Raspberryä.

Yhdyskäytävän yhdistys Azure IoT Hubiin

Jotta Raspbianin voi yhdistää Azuren IoT Hubiin, täytyy ensiksi tehdä tunnukset Azure IoT Hub portaaliin ja luoda sieltä uusi IoT Hub kohdasta ”New -> Internet of Things -> IoT Hub”. Tämän jälkeen voi valita muutamia asetuksia, kuten hinnoittelutyylin. Ilmaisessa hinnoittelussa saa maksimissaan 8000-viestiä per päivä, mikä kelpaa minulle työni tekemiseen paremmin kuin riittävästi. IoT Hubin luomisen jälkeen tulee kyseisen Hubin ”General -> Shared access policies” välilehden alta ottaa talteen Connection String – Primary Key (kuva 10). Tällä avaimella on mahdollista myöhemmin yhdistää yhdyskäytävä eli Raspbian osaksi omaa IoT Hubia.



Kuva 10 IoT Hubin yhdistämisavaimella pääsee lisäämään laitteita IoT Hubiin

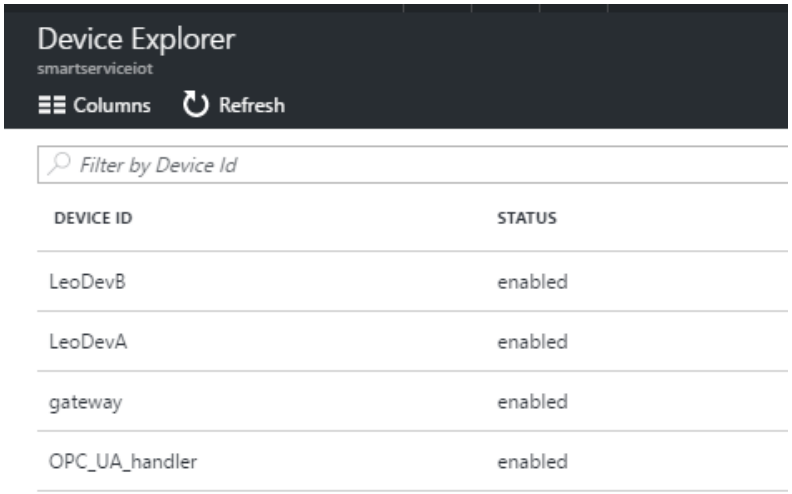
Yhdyskäytävän yhdistäminen Azure IoT Hubiin onnistuu muutamallakin eri tavalla. Ainakin C:llä, .NET:llä, Node.JS:llä ja Javalla on tehty oma oma valmis ”SDK”-paketti Raspbianin yhdistämiseen Azureen (Concurrency, 2016). Valitsin tavaksi Node.JS:n, sillä tällä tavoin on mahdollista saada yhteen ja samaan .js -tiedostoon niin iBeaconeiden skannaus kuin Azureen yhdistys. Täten asiat pysyvät mahdollisimman yksinkertaisena.

Seuraava vaihe vaatii jo edellisessä vaiheessa asennettua Node Package Manager -kirjastoa. Mikäli sitä ei vielä jostain syystä löydy laitteelta, asennus onnistuu komennoilla *sudo apt-get install npm* ja *sudo npm install -g npm@2.x*. Seuraavaksi tulee asentaa Azure IoT Hub-osa sekä IoT Explorer-osa alla olevilla komennoilla komentorivissä:

sudo npm install -g azure-iot-device@latest

sudo npm install -g azure-iot-device-http@latest
sudo npm install -g iothub-explorer@latest

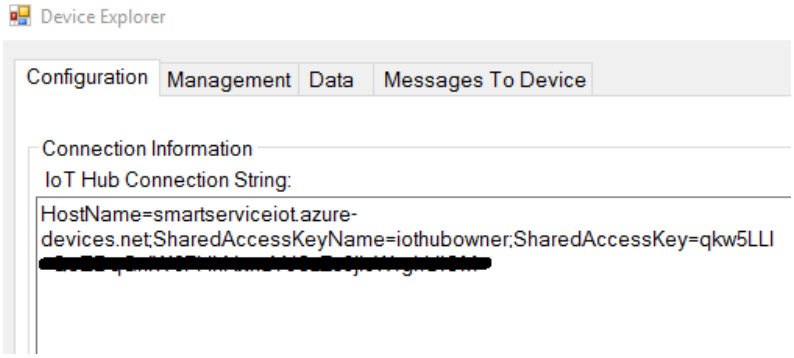
IoT Explorerilla pääsee itse Azure IoT Hubiin kiinni Raspbianilta kirjautumalla sisään komennolla *iothub-explorer login connection-string*. Tässä tapauksessa *connection-string* -komennon tilalle tulee aikaisemmin mainittu *connection string* IoT Hub Portaalista. Mikäli sisäänkirjautuminen onnistui, Raspbianin lisäys uudeksi laitteeksi tapahtuu kommennolla *iothub-explorer create ”laitteen nimi” –connection string*. Laitteen nimeksi voi laittaa minkä haluaa, minä kutsuin omaa laitettani yksinkertaisesti *gateway*. Tämän jälkeen Azure IoT Hub portaaliin tulisi ilmestyä uusi laite Device Explorer-välilehdelle, mikä on esitetty kuvassa 10.



Kuva 10 "Gateway" ilmestynyt Azure IoT Hub portaaliin laitteiden joukkoon

Azure IoT Hubin portaalissa ei itsessään ole integroitu helppoa tapaa nähdä läpikulkevien viestien sisältöä. Tätä varten on kuitenkin kehitelty ohjelma nimeltä Device Explorer (Github.com/Azure-iot-sdks, 2017). Device Explorer pohjautuu Visual Studioon ja sillä saa IoT Hubin läpikulkevat viestit jokaisesta laitteesta näkyviin ja lisäksi sillä voi myös lähettää viestejä jokaiselle laitteelle, jotka on lisätty osaksi IoT Hubia.

Device Explorerilla pääsee kirjautumaan koko IoT Hubiin sisään syöttämällä kuvassa 10 näkyvän IoT Hubin ”connection-stringin” ohjelman Configuration välilehdelle alla olevan kuvan 11 mukaisesti.



Kuva 11 Device Explorerilla kirjautuminen Azure IoT Hubiin

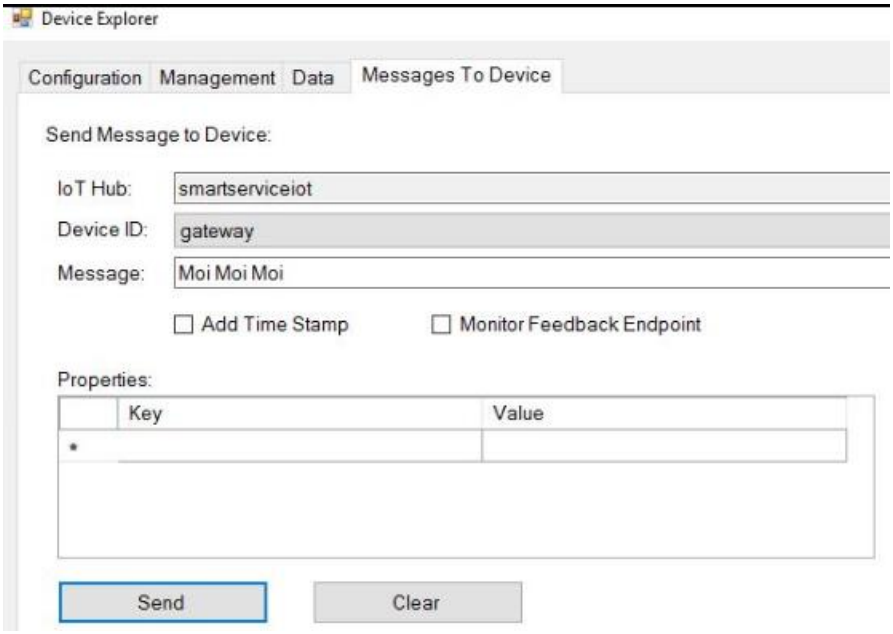
”Management”-välilehdellä näkee kaikki toistaiseksi luodut laitteet omassa IoT Hubissa. Sieltä pystyy myös lisäämään uusia ja poistamaan vanhoja laitteita. ”Data”-välilehti toimii monitorointityökaluna omille laitteille ja niiden viestittelylle. ”Messages To Device”-välilehdellä pystyy lähettämään laitteille viestejä.

Tämän jälkeen Internetistä ja mm. Microsoftin omilta sivuilta on mahdollista löytää monia erilaisia esimerkkejä kuinka lähettää viesti omalla laitteellaan IoT Hubille. Tein Raspbianilla .js-tiedoston komentorivillä komennolla *nano azure.js* ja kopion siihen kuvan 12 mukaisen koodin (Concurrency.com, 2016) muuttaen ensimmäiselle riville oman IoT Hubin connection-stringin:

```
var connectionString = '<your device connection string>';
var clientFromConnectionString = require('/usr/lib/node_modules/azure-iot-device-http').clientFromConnectionString;
var client = clientFromConnectionString(connectionString);
var Message = require('/usr/lib/node_modules/azure-iot-device').Message;
var msg = new Message('some data from my device');
var connectCallback = function (err) {
  if (err) {
    console.error('Could not connect: ' + err);
  } else {
    console.log('Client connected');
    var message = new Message('some data from my device');
    client.sendEvent(message, function (err) {
      if (err) console.log(err.toString());
    });
    client.on('message', function (msg) {
      console.log(msg);
      client.complete(msg, function () {
        console.log('completed');
      });
    });
  }
};
client.open(connectCallback);
```

Kuva 12, Esimerkkikoodi, joka lähettää viestin IoT Hubille ja odottaa vastausta

Suorittamalla .js tiedoston komennolla *node azure.js* yhdyskäytävä lähettää viestin IoT Hubiin ja jää odottamaan vastausta. Device Explorerilla monitoroidessa yhdyskäytävää viesti tulee näkyviin ajan ja päivämäärän kanssa. Device Explorerilla voi lähettää viestin yhdyskäytävälle kuvan 13 mukaan:



Kuva 13 Viestin lähetys yhdyskäytävälle Device Explorerin avulla.

Esimerkkikoodissa yhdyskäytävä jää odottamaan mahdollisia viestejä ja täten vastaanotettu viesti näkyy komentorivillä alla olevan kuvan 14 muodossa:

```
Client connected
Message {
  data: 'Moi Moi Moi',
  properties: Properties { propertyList: [] },
  messageId: '35f51159-92de-4313-a24d-7917e5720742',
  to: '/devices/gateway/messages/deviceBound',
  expiryTimeUtc: '',
  lockToken: '4667f7dd-c9df-4ba5-ba80-5ae5fddaf46c',
  correlationId: '',
  userId: '' }
completed
```

Kuva 14 Vastaanotettu viesti yhdyskäytävällä

iBeacon-laitetietojen lähetys Azure IoT Hubille

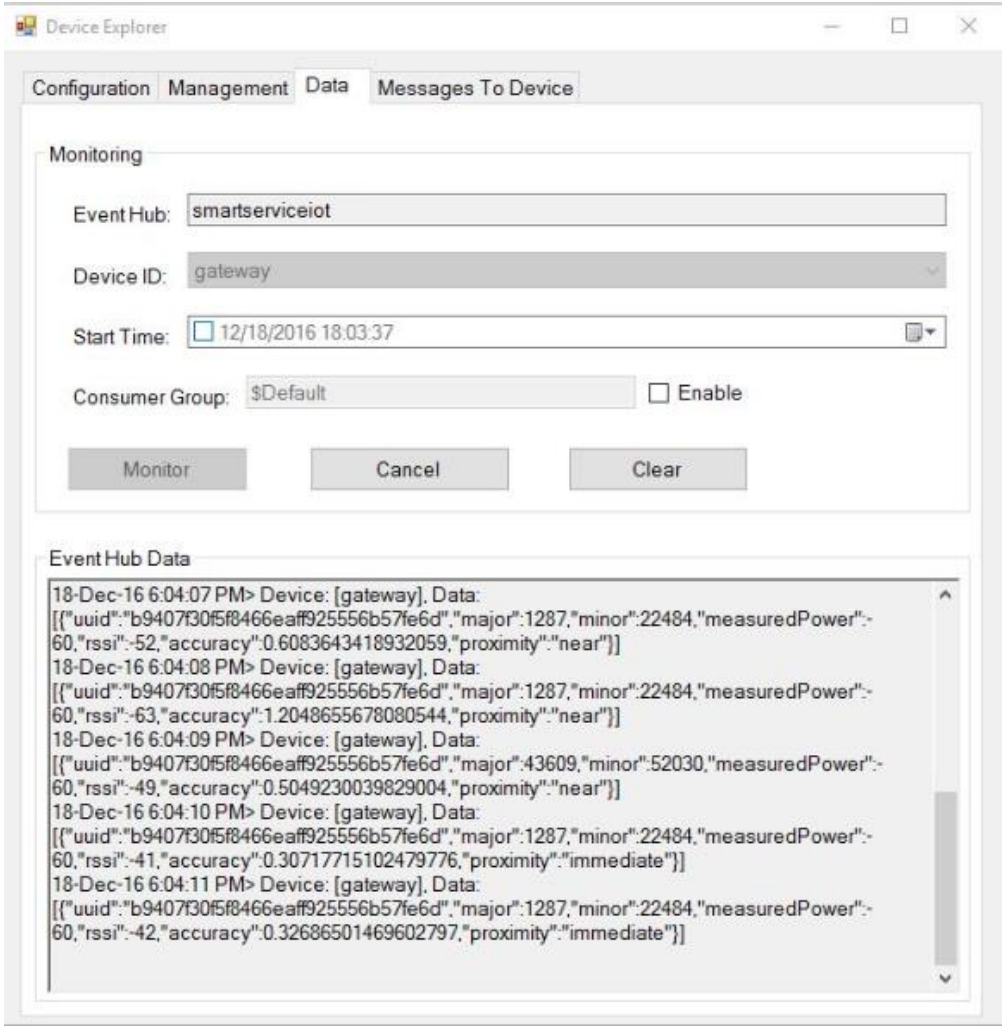
Jotta oli mahdollisuus saada työn alkuperäinen tavoite saavutettua, eli iBeaconeiden skannaaminen sekä niiden tietojen lähettäminen eteenpäin yhdyskäytävältä Microsoft Azure IoT Hubille, jäljelle jäävä tehtävä oli saada luotua yksi toimiva .js-tiedosto, jolla onnistui sekä iBeaconeiden skannaus että samalla tietojen lähetys eteenpäin IoT Hubille. Tämä tiedosto on mahdollista rakentaa edellä esitettyjä Node.js -ratkaisuja yhdistelemällä koodi samaan tiedostoon sekä lisäämällä yksi muuttuja eli variable. Kyseiseen muuttujaan tallennetaan skannauksesta saatu tieto ja sitä käytetään myöhemmin hyväksi tietojen lähetysvaiheessa IoT Hubille. Kutsuin kyseistä

muuttujaa nimellä ”data” ja koko tiedoston koodi on kokonaisuudessaan esitelty alla kuvassa 15:

```
var Bleacon = require('./index');
var connectionString = 'Tähän Connection-String Azure Portaalista';
var clientFromConnectionString =
require('/usr/lib/node_modules/azure-iot-device-http').clientFromConnectionString;
var client = clientFromConnectionString(connectionString);
var Message = require('/usr/lib/node_modules/azure-iot-device').Message;

var data = "";

Bleacon.on('discover', function(bleacon) {
  console.log('bleacon found: ' + JSON.stringify(bleacon));
data = JSON.stringify(bleacon);
});
Bleacon.startScanning();

var connectCallback = function (err) {
  if (err) {
    console.log('Could not connect: ' + err);
  } else {
    console.log('Client connected');

function printResultFor(op) {
  return function printResult(err, res) {
    if (err) console.log(op + ' error: ' + err.toString());
    if (res) console.log(op + ' status: ' + res.constructor.name);
  };
}
    setInterval(function(){
        var message = new Message(data);
        console.log("Sending message: " + message.getData());
        client.sendEvent(message, printResultFor('send'));
    }, 1000);
}
};
client.open(connectCallback);
```

Kuva 15 Lopullinen koodi, jolla yhdyskäytävä skannaa iBeaconit ja lähettää niiden tiedot eteenpäin IoT Hubiin

Koodin alussa määritellään pakolliset muuttujat ja tukitiedostot, joita toiminen vaatii. Lisäksi siinä on määritelty jo tutuksi tullut connection-string, joka on avain Azure IoT Hubiin yhdistämiseen. Bleacon-rivit liittyvät iBeaconeiden skannaukseen ja loppuosa koodista keskittyy tietojen lähettämiseen Azure IoT Hubiin. Viimeisessä vaiheessa oleva luku 1000 tarkoittaa viestien lähettämisen frekvenssiä. Mikäli ko. luku on 1000, yhdyskäytävä lähettää tällöin yhden viestin sekunnissa IoT Hubille. Jos luku on esim. 10000, yhdyskäytävä lähettää yhden viestin kymmenessä sekunnissa. Kuvassa 16 on esitelty Device Managerin näkymä IoT Hubin saamista viesteistä sen monitoroidessa yhdyskäytävän lähettämiä viestejä.



Kuva 16 IoT Hub vastaanottaa iBeacon tietoja yhdyskäytävältä.

6 YHTEENVETO

Tässä työssä on käsitelty yksinkertainen IoT-arkkitehtuuri iBeacon-laitetietojen saamiseksi pilvipalveluun. Suoraa ratkaisua iBeacon-laitetietojen lähettämiseksi Azurelle en löytänyt Internetistä, vaan tehtävässä tuli soveltaa ja testata eri menetelmiä omien kykyjen mukaan. Erityinen kulmakivi työn onnistumisen kannalta oli iBeaconeiden skannaustyylin valinta yhdyskäytävällä. Varmasti on olemassa keinoja, millä myös Python-kielisen koodin saa yksinkertaisesti lähetettyä Azurelle. Node.js kuitenkin tuntui sopivammalta valinnalta, sillä tarkoituksena oli lähes alusta asti tehdä yhdyskäytävän ja Azuren linkittäminen Node.js -avulla, johon löytyi Microsoftin sivuilta jonkin verran apua ja esimerkkejä.

Aikataulullisesti työ venyi liian pitkälle, johtuen suurimmaksi osaksi opinnäytetyön ja oikean työn päällekkäisyydestä sekä vapaa-ajan vähyydestä. Tästä viisastuneena tiedänkin vastaisuudessa pitää lomaa muista töistä mikäli vastaavanlainen työ tulee vielä eteen elämässä. Olen lopputulokseen suhteellisen tyytyväinen ottaen huomioon sen, että työssä joutui soveltamaan ja testaamaan paljon, minkä kautta toimiva ratkaisu löytyi.

Työtä olisi hyvä lähteä jalostamaan eteenpäin telemetriadatojen visualisoinnin kautta Microsoft Azuressa. IoT Suite -palvelu tarjoaa ratkaisuja jatkuvalle monitoroinnille ja erilaisten tapahtumien paikannukselle Azure Stream Analytics -osan avulla. Lisäksi esimerkiksi telemetriatietojen tallennus pilveen onnistuu Azure Storagen tai Azure DocumentDB:n avulla. Tätä kautta voisi olla mahdollista rakentaa täysimuotoinen IoT-ympäristö perustuen iBeaconeihin ja Raspbian-laitteeseen. Monissa iBeacon malleissa on lisäksi lämpötilasensori, jonka tiedot voisi olla mahdollista saada talteen ja näin saada konkreettista dataa pilvipalveluun.

LÄHTEET

IoT Agenda, Internet of Things, viitattu 09/2016
http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

IEEE Spectrum news, viitattu 09/2016
http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

Beaconsandwich, what is iBeacon, viitattu 10/2016
http://www.beaconsandwich.com/what-is-ibeacon.html

Wikipedia iBeacon, viitattu 10/2016
https://en.wikipedia.org/wiki/IBeacon

Kontakt.io, Kontakt.io devices, viitattu 10/2016
https://support.kontakt.io/hc/en-gb/articles/201620741-iBeacon-Parameters-UUID-Major-and-Minor

Estimote, What is iBeacon?, viitattu 10/2016
http://developer.estimote.com/ibeacon/

Verkkokauppa.com, Raspberry Pi, viitattu 10/2016
http://verkkokauppa.com

SearchCloudComputing, Microsoft Azure, viitattu 09/2016
http://searchcloudcomputing.techtarget.com/definition/Windows-Azure

Instructables, viitattu 01/2017
http://www.instructables.com/id/iBeacon-Entry-System-with-the-Raspberry-Pi-and-Azu/step2/Scanning-for-iBeacons/

Node Js, Installation, viitattu 10/2016
https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions

Github, Sandeepmistry, viitattu 10/2016
https://github.com/sandeepmistry/node-bleacon

Concurrency, Integrating Raspberry Pi with Azure, viitattu 10/2016
https://www.concurrency.com/blog/march-2016/azure,-azure-automation-,-azure-iot-hub-,-iot,-l

Github, Azure IoT SDK's, viitattu 01/2017
https://github.com/Azure/azure-iot-sdks/releases

Azure Microsoft, viitattu 01/2017
https://azure.microsoft.com/en-us/services/virtual-machines/

CRN, news, viitattu 01/2017
https://www.crn.com/slide-shows/cloud/300079669/keeping-up-with-the-cloud-top-5-market-share-leaders.htm/pgno/0/6

Microsoft docs, Azure and Internet of things, viitattu 01/2017
https://docs.microsoft.com/en-us/azure/iot-suite/iot-suite-what-is-azure-iot