OULUN AMMATTIKORKEAKOULU

Joonas Heikkinen

Internet of Things -testiympäristö Microsoft Azure -pilvipalvelualustalla

Internet of Things -testiympäristö Microsoft Azure -pilvipalvelualustalla

Joonas Heikkinen
Opinnäytetyö
Syksy 2016
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely, järjestelmäasiantuntemus

Tekijä(t): Joonas Heikkinen
Opinnäytetyön nimi: Internet of Things -testiympäristö Microsoft Azure -pilvipalvelualustalla
Työn ohjaaja: Risto Hinkka
Työn valmistumislukukausi ja -vuosi: Syksy 2016 Sivumäärä: 33 + 3

Opinnäytetyön tavoitteena oli luoda Internet of Things testiympäristö Microsoft Azure -pilvipalvelualustalle. Tarkoituksena oli testata IoT-laitteen toimintaa, datan siirtämistä pilvipalveluun, sekä sen hakemista takaisin pilvipalvelusta, prosessoimista varten. Tavoitteena oli rakentaa oikeaa IoT-ympäristöä vastaava kokonaisuus, jonka avulla saatiin tietoa oikean ympäristön toiminnasta pienessä mittakaavassa.

Teoriaosuudessa käytiin läpi pilvipalveluiden yleistä teoriaa, palvelumalleja, sekä syitä miksi valita pilvipalvelu tavallisen palvelinympäristön sijaan. Esineiden internetistä käytiin läpi yleistä teoriaa, yhteysmalleja, sekä tietoturvaa. Big datan teoriaa käytiin läpi vain lyhyesti, sillä se ei ollut olennaisin osa käytännön osuuden kannalta. Microsoft Azuren teoriaa käytiin läpi yleisellä tasolla, sekä ominaisuuksia tarvittavilta osin.

Pilvipalveluympäristö rakennettiin käyttämällä Microsoft Azuren IoT Hubia. Testaukseen käytetyt ohjelmat ohjelmoitiin Microsoft Visual Studiolla, C# ohjelmointikielellä. Testiympäristöä käytettiin tavallisella työasemalla, joka oli yhteydessä Internetiin.

Lopputuloksena oli toimiva IoT-testiympäristö, jonka avulla IoT-laitteiden ja pilvipalvelun yhteistoiminta simuloitiin onnistuneesti. C# konsoliohjelmia rakennettiin yhteensä 3 kappaletta ja ne kaikki olivat yhteydessä IoT Hubiin. Ensimmäinen konsoliohjelma muodosti lisättäville IoT-laitteille identiteetin, turvallisen yhteyden takaamiseksi. Toinen konsoliohjelma simuloi IoT-laitteen toimintaa. Laite loi ja lähetti dataa onnistuneesti IoT Hubiin. Kolmas konsoliohjelma haki pilveen tallennettua data takaisin, jatkokäsittelyä varten.

Työ oli kaikin puolin onnistunut kokonaisuus. Työ valmistui aikataulussa, teoriaosuus muodosti kattavan tietoperustan käytännön osuutta varten ja käytännön osuudessa rakennettu ympäristö toimi halutulla tavalla.

Asiasanat: Big data, Internet of Things, Microsoft Azure, pilvilaskenta, pilvipalvelut,

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Systems, Computer Systems Expertise

---

Author(s): Joonas Heikkinen
Title of thesis: Internet of Things testing environment on Microsoft Azure cloud platform
Supervisor(s): Risto Hinkka
Term and year when the thesis was submitted: Autumn 2016 Number of pages: 33 + 3

The aim of this thesis was to build and configure Internet of Things testing environment using Microsoft Azure cloud services. The environment was created to test how IoT devices are sending data to the cloud service, how the data is stored in the cloud and how it is retrieved from the cloud for further processing. The primary goal was to build an environment that simulates real IoT environment in a small scale, in order to gain knowledge about the working principles of IoT devices over cloud platform.

Microsoft Azure IoT Hub was selected as the IoT cloud platform. Programs used for testing were programmed using Microsoft Visual Studio and C# programming language. The created programs were programmed and executed on a regular workstation. Microsoft products were selected based on their popularity.

The theoretical part consists on theory of cloud computing, Internet of Things, Big Data and Microsoft Azure. The theory section was prepared to support the practical part of the thesis. Cloud computing and Internet of Things have the biggest roles in the theoretical background. The section of Microsoft Azure introduces the necessary pats for the thesis. Big data has minor role in the theory section.

The results were successful: testing environment worked as it was supposed to and no major problems were encountered during the process. The theory section supported the practical part as intended. It offered the necessary knowledge to understand the practical part of the thesis. In conclusion, the thesis succeeded well in its aim to simulate a real IoT environment.

Keywords: Big data, Cloud computing, Internet of Things, Microsoft Azure,

SISÄLLYS

1 JOHDANTO

Tämä opinnäytetyö käsitteli esineiden internetiä, sekä pilvipalveluita. Molemmat aiheet olivat hyvin ajankohtaisia. Pilvipalvelut olivat yleistymässä voimakkaasti – ne kilpailivat tavanomaisten, yritysten tiloissa sijaitsevien palvelinratkaisujen rinnalla. Esineiden internet yleistyi myös kovaa vauhtia. Laitteet olivat kustannustehokkaita, pienikokoisia ja suorituskykyisiä. Ne alkoivat tarjota uudenlaisia mahdollisuuksia mitä erilaisimmissa käyttötarkoituksissa.

Työn motivaation lähteenä toimi aihepiirin kiinnostavuus ja ajankohtaisuus. Työn tavoitteena oli tuoda esille osaamista ja kiinnostusta valittuun aihepiiriin. Näin työllä oli mahdollisuus johtaa jopa työllistymiseen.

Kaikki työssä käytetyt ratkaisut olivat Microsoftin tuotteita. Pilvipalveluympäristöksi valittiin Microsoft Azure. Valinta oli selkeä Microsoftin maailmanlaajuisen suosion vuoksi. Laajalti käytössä olleet tuotteet sekä ajankohtaiset aihepiirit takasivat, että työtä varten oli saatavilla kattavasti aineistoa. Käytetyt sovellukset rakennettiin myös Microsoftin tuotteiden avulla. Microsoft Visual Studio tarjosi kattavan sovelluskehitysympäristön, sekä mahdollisti Microsoftin luoman C#-ohjelmointikielen käyttämisen.

Käytännön osuudessa rakennettiin testiympäristö, jonka tarkoituksena oli simuloida esineiden internetin toimintaa pilvipalvelualustan avulla. Teoriaosuus rajattiin tukemaan työn käytännön osuutta. Kappaleissa käytiin läpi pilvilaskenta, esineiden internet, big data sekä Microsoft Azure. Aihealueet avattiin selkeästi, tarpeenmukaisella laajuudella. Päämääränä oli luoda käytännön toteutuksen avulla selkeytetty, helppolukuinen ja kattava tietopaketti pilvipalveluiden ja esineiden internetin maailmasta.

Työn tarkoitus oli antaa lukijalle näkökulmaa siitä, millainen prosessi esineiden internet -laiteympäristön pystyttäminen on, sekä herättää ajatuksia millaisissa tarkoituksissa ympäristöä voisi hyödyntää. Olennainen näkökulma oli myös pilvipalveluiden, sekä esineiden internet -ympäristön välinen yhteys. Lopputuloksena oli kattava tietopaketti, joka tarjosi mahdollisuuden tutustua ajankohtaisiin aihepiireihin sekä teoriassa, että käytännössä.

2 PILVILASKENTA

Pilvilaskenta on malli, jolla viitataan Internetin kautta toimiviin hajautettuihin ympäristöihin. Ympäristöt tarjoavat joukon jaettuja palveluita, varustettuna korkealla saatavuudella. Konfiguroitavat palvelut, kuten tietoverkot, palvelimet, tallennustila ja lukuisat sovellukset ovat rakennettavissa ja ylläpidettävissä kohtuullisilla resursseilla. (National Institute of Standards and Technology 2012, 2-1.) Pilvilaskenta on skaalautuva, kustannustehokas tapa rakentaa uusi, tai siirtää olemassa oleva ympäristö – samalla parantaen palveluiden saatavuutta (Amazon 2016, viitattu 11.8.2016). National Institute of Standards and Technologyn laatima kuvaus määrittelee pilvilaskennan varsin hyvin:

> *Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (National Institute of Standards and Technology 2012, 2-2).*

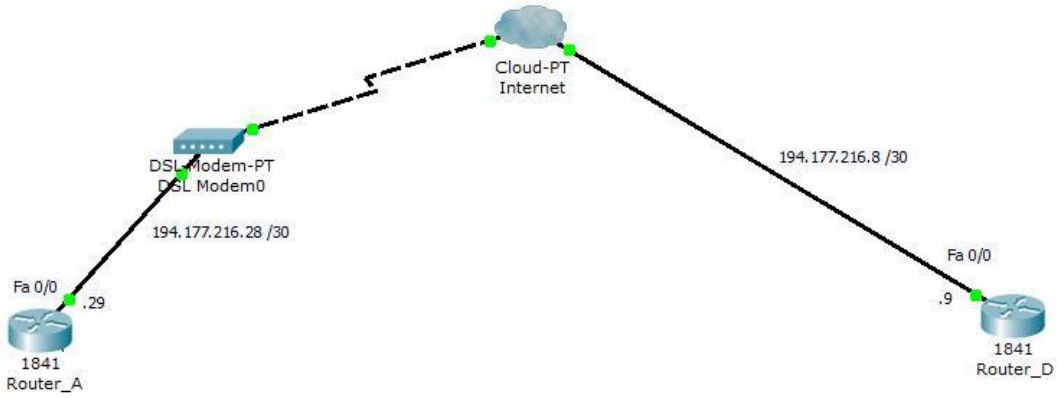
Tässä kappaleessa perehdytään pilvilaskentaan. Kappaleessa käydään läpi pilvilaskennan eri osa-alueita, historiaa, keskeisiä ominaisuuksia ja pilvilaskennan etuja. Microsoft Azure on pilvilaskentaan perustuva palvelu, joten tämä teoriaosuus tukee työn käytännön osuutta.

2.1 Taustatietoa

Varhaisimmat ideat keskitetystä laskennasta esiteltiin jo 1960-luvulla. Tämän aikaiset tietokoneet olivat kuitenkin hyvin alkeellisia ja nykyaikainen Internet oli olemassa vasta kehitysasteella. Ensimmäiset Internetiä hyödyntävät, keskitettyyn laskentaan perustuvat palvelut ilmestyivät vasta 1990-luvulla. Kyseisiä tekniikoita hyödyntävistä palveluista varhaisia versioita tarjosivat nykypäivänäkin käytössä olevien palveluiden sen aikaiset versiot, kuten hakukonepalvelut Google ja Yahoo, sekä sähköpostipalvelut Hotmail ja Gmail. (Erl 2013, 26.)

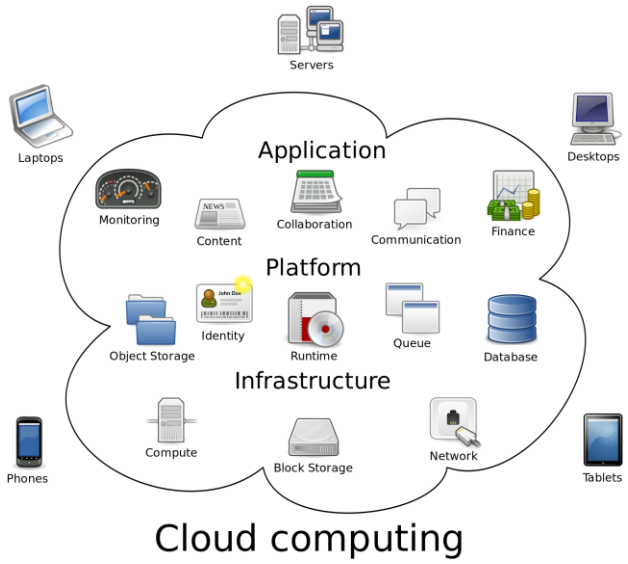
Termi pilvi (englanniksi cloud) esiteltiin 1990-luvulla tietoverkkojen yhteydessä. Verkkorakenne, joka koostui useista sisäverkoista ja julkisesta ulkoverkosta oli yleistymässä. Pilvellä viitattiin julkiseen ulkoverkkoon, jonka läpi data liikkui esimerkiksi lähteen ja kohteen välillä. Termiä pilvi käytettiin siis kuvaamaan WAN (wide area network) -verkkoa. (Erl 2013, 27.) Termiä käytetään tässä

yhteydessä edelleen. Pilven kuvaketta käytetään simuloimaan julkista Internetiä kahden sisäverkon välillä, kuvion 1 osoittamalla tavalla.



KUVIO 1. Pilvi-ikoni kuvastaa Internetiä Cisco Packet Tracer –sovelluksessa (Cisco Support Community 2012, viitattu 26.8.2016)

Pilvilaskennan ja -palveluiden yhteydessä sanaa pilvi käytetään ilmaisemaan tiettyä, tarkemmin rajattua ympäristöä tai alustaa. Sillä viitataan keskitettyyn, skaalautuvaa laskentatehoa tarjoavaan palveluun, joka on laajasti saatavilla verkkoyhteyttä hyödyntämällä. Kuvio 2 kuvastaa nykyaikaista pilveä. (Erl 2013, 33.)



KUVIO 2. Pilvi-ikoni kuvastamassa pilvipalvelua (Wikipedia 2016, viitattu 26.8.2016)

Amazon käytti termiä pilvilaskenta ensimmäisen kerran kaupallisesti vuonna 2006, julkaistessaan EC2:n (Elastic Compute Cloud) – ensimmäisen ulkoistettua laskentatehoa tarjoavan palvelun. Organisaatiot pystyivät hyödyntämään vuokrattavaa laskentatehoa yritystensä sovellusratkaisuihin, käytettäväksi Internetin välityksellä. Pian tämän jälkeen myös Google liittyi mukaan markkinoille Google Apps -sovellusvalikoimallaan. (Collier & Shahan 2015, 27.) Näin ollen vuotta 2006 voidaan pitää nykyaikaisten pilvipalveluiden alkupisteenä.

2.2 Keskeiset ominaisuudet

National Institute of Standards and Technology on Yhdysvaltalainen hallituksen alaisuudessa toimiva virasto, jonka tehtävänä on edistää ja kehittää standardeja, mittaustekniikoita ja tekniikkaa (National Institute of Standards and Technology 2016, viitattu 25.8.2016). Täten sitä voidaan pitää puolueettomana ja luotettavana lähteenä. Heidän mukaan pilvilaskennalla on viisi keskeisintä ominaisuutta. Järjestö listaa seuraavat ominaisuudet, joita voidaan pitää tyypillisen pilvipalvelun määritelmänä. (National Institute of Standards and Technology 2012, 2-1.)

Käyttö itsepalveluperiaatteella. Kuluttaja pystyy käyttämään pilvipalvelun tarjoamia resursseja automaattisesti, tarpeen vaatiessa. Resurssien käyttäminen ei vaadi muiden käyttäjien tai ylläpitäjien toimenpiteitä palvelun tarjoajan puolelta. (sama, 2-1.)

Laaja saatavuus. Palvelut ovat saatavilla internet-yhteyden välityksellä ilman erityisiä toimenpiteitä. Palveluita pystyy käyttämään erilaisilla päätelaitteilla, esimerkiksi tietokoneella tai älypuhelimella. (sama, 2-1.)

Resurssien varastointi. Käytettävissä olevat fyysiset ja virtuaaliset resurssit on jaettu tarpeen mukaan useiden pilvipalvelun käyttäjien kesken. Käyttäjällä ole tarkkaa tietoa mistä kullakin hetkellä hänelle varatut resurssit tarjotaan. Tarvittaessa on kuitenkin mahdollista tarkentaa resurssien alkuperää esimerkiksi maan, tai palvelinkeskuksen tarkkuudella. (sama, 2-1.)

Nopea skaalautuvuus. Resurssit skaalautuvat nopeasti tarpeen kasvaessa. Kun resursseja ei tarvita, skaalautuvat ne vuorostaan alaspäin. Skaalautuvuus on usein toteutettu siten, että käyttäjän näkökulmasta resursseja näyttää olevan käytössä rajaton määrä. (sama, 2-1.)

Valvottu toiminta. Pilvijärjestelmät valvovat, ohjaavat ja optimoivat resursseja automaattisesti. Nämä tiedot ovat sekä palveluntarjoajaa, että palvelun kuluttajaa varten, luoden läpinäkyvyyttä palvelun kaikkien osapuolten kesken. (sama, 2-1–2-2.)

2.3 Palvelumallit

Pilvipalveluille ominainen jaottelutapa ovat palvelumallit. Palvelut jaetaan yleisesti kolmeen palvelumalliin. (sama, 2-1.) Microsoft Azuren tarjonnasta löytyy kaikkien näiden kolmen mallin mukaisia palveluita (Collier & Shahan 2015, 19–20).

2.3.1 Infrastructure-as-a-Service

IaaS (Infrastructure-as-a-Service) tarkoittaa pilvipalveluiden tarjoamia fyysisiä resursseja, kuten laskentatehoa, varastotilaa, tietoverkkoa sekä muita keskeisiä resursseja. Pilvi-infrastruktuurin avulla pystytään asentamaan ja suorittamaan resursseja, kuten käyttöjärjestelmiä ja ohjelmistoja. (National Institute of Standards and Technology 2012, 2-1.)

IaaS-palvelumallissa palvelinkeskusten fyysinen infrastruktuuri on koottu joukoksi palveluita, joita pystytään hallinnoimaan ja automatisoimaan web-käyttöliittymän tai ohjelmointiympäristön kautta. Sovellukset eivät kuulu IaaS-tyyppisten palveluiden valikoimaan; kolmannen osapuolen ratkaisujen asentaminen ja ylläpito ovat sovelluskehittäjien vastuulla. IaaS:n avulla fyysisten resurssien hankinnaista ja ylläpitämisestä päästään kuitenkin eroon. (Kavis 2014, 47.)

2.3.2 Platform-as-a-Service

PaaS (Platform-as-a-Service) -palvelumalli on pilvipalveluympäristö, jonka avulla pystytään kehittämään, julkaisemaan ja käyttämään sovelluksia. Tyypillinen PaaS-infrastruktuuri tarjoaa monipuolisen valikoiman sovelluskehittäjien työkaluja, kuten sovellusalustan, integraatioalustan sekä analysointi- ja monitorointityökaluja. Palvelumalli on muun muassa sovelluskehittäjien suosiossa. (Cloud Standards Customer Council 2015, 7.)

Palvelumallin käyttäjä luo oman sovelluksensa alustalle, käyttäen palveluntarjoajan tarjoamia resursseja kuten ohjelmointikieliä ja työkaluja. PaaS:lle ominaista on, etteivät käyttäjät pysty hallitsemaan fyysisiä resursseja – ne tarjotaan valmiina palveluna. (National Institute of Standards and Technology 2012, 2-2.)

2.3.3 Software-as-a-Service

SaaS (Software as a Service) -palvelumallin mukaiset toteutukset ovat tyypillisesti valmiita, loppukäyttäjille suunnattuja palveluita. SaaS-palvelut vaativat minimaalisen määrän konfigurointia tai ylläpitoa ja tarjotaankin yleensä 'kirjaudu ja käytä' -periaatteella. (Cloud Standards Customer Council 2015, 6.) Valmista sovellusta käytetään sovelluksen tai web-käyttöliittymän kautta halutulla päätelaitteella, kuten älypuhelimella tai tietokoneella (National Institute of Standards and Technology 2012, 2-1–2-2).

2.4 Pilvipalvelun edut

Tullochin mukaan kolme yleisintä pilvipalvelun implementointiin johtavaa tekijää ovat nopeus, skaalautuvuus sekä kustannustehokkuus (Tulloch 2013, 1). Pilveen siirtymiseen on toki muitakin vaikuttavia tekijöitä, ympäristöstä ja käyttötarkoituksesta riippuen. Tässä kappaleessa esitellään kolme edellä mainittua perustelua.

2.4.1 Nopeus

Tyypillisiin palvelinratkaisuihin verrattuna sovelluskehitys on pilvipalvelualustalla huomattavasti helpompaa. Yrityksen omia palvelimia hyödyntävissä kokoonpanoissa sovelluskehitys aloitetaan luomalla palvelinalusta, joka pystyy tarjoamaan riittävästi laskentatehoa, tietokannan, tietoverkon sekä muut tarvittavat ominaisuudet. Pilvipalveluissa nämä resurssit ovat jo olemassa, valmiiksi rakennettuna kokonaisuutena. (sama, 1–2.)

Toinen aspekti liittyy sovellusten testaamiseen. Tyypillisessä palvelinympäristössä uusien järjestelmien ja muutosten simulointia varten käytettävät testiympäristöt ovat harvoin täysin verrattavissa oikean palvelimen toimintaan. Ympäristöjen rakentaminen ja konfiguroiminen testiajoa varten vievät myös aikaa. Pilvipalveluissa uudet palvelut tai muutokset pystytään suorittamaan varsinaista

palvelinta vastaavissa, sisäänrakennetuissa testiympäristöissä. Tämä nopeuttaa sovellustestausta huomattavasti, sekä tekee testituloksista luotettavampia. (sama, 2.)

2.4.2 Skaalautuvuus

Organisaatiolle tarvittavien resurssien kartoittaminen voi olla vaikeaa. Laskentatehon lisäksi on huomioitava myös riittävä verkon kapasiteetti ja tallennustilan määrä. Resursseja pitää olla riittävästi tukemaan organisaation kasvua ja niitä pitää pystyä kasvattamaan tarpeen mukaan. Vastaavasti kysynnän laskiessa, perinteisessä palvelinmallissa resursseja on enemmän kuin tarvetta – investoinnit menevät tässä vaiheessa hukkaan. Pilvipalvelualustat pystyvät vastaamaan tarpeen kasvuun sekä vähenemiseen nopeasti ja luotettavasti. (sama, 2.)

Pilvipalveluiden skaalautuvuus on lyömätön myös isommassa mittakaavassa. Organisaation kasvaessa globaalisti, pilviratkaisuilla pystytään tarjoamaan samat resurssit maailmanlaajuisesti. (sama, 2.)

2.4.3 Taloudellisuus

Edellä käytiin läpi pilvipalveluiden teknisiä etuja. Kolmanneksi yleisin syy pilvilaskentaan siirtymiseen on kustannustehokkuus. Pilvipalveluiden hinnat perustuvat usein käytön mukaiseen veloitukseen. Kappaleeseen 2.3.2 viitaten, resurssien käyttöaste skaalautuu kysynnän mukaan. Tämä vaikuttaa suoraan pilvipalvelun hintaan. Käyttämättä jäävästä tehosta ei tarvitse maksaa. (sama, 2–3.)

Pilvipalveluiden käyttäminen voi olla kannattavaa myös fyysisen laitteiston osalta. Tullochin mukaan etenkin isoilla organisaatioilla on tapana turvautua SAN:iin (Storage Area Network). Kyseinen palvelu tarjoaa tallennuspaikan datalle, kun esimerkiksi yrityksen omat resurssit loppuvat kesken. Varastotilan aiheuttamista hankinta- ja ylläpitokuluista päästään eroon hyödyntämällä pilvipalveluiden tarjoamaa tallennustilaa. (sama, 3.)

3 INTERNET OF THINGS

Termi esineiden internet (Internet of Things, IoT) on verrattain uusi. Sitä käytettiin ensimmäisen kerran vuonna 1999 kuvastamaan järjestelmää, jossa fyysisen maailman objektit yhdistettäisiin sensoreina Internetiin. Sen sijaan nykyaikaisempi määritelmä esineiden internetille on skenaario, jossa erilaiset objektit, laitteet, sensorit tai arkipäiväiset esineet valjastetaan hyötykäyttöön, hyödyntämällä internet-verkkoa ja laskentatehoa. (Internet Society 2015, 7.)

Esineiden internetin suosio on kasvanut teknologisen kehityksen myötä. Entistä useampien ja yhä pienempien laitteiden yhdistäminen Internetiin on nyt mahdollista. Internet Society listaa kuusi syytä IoT:n yleistymiselle. Laitteiden yhtäaikainen yhdistäminen on mahdollista nopeiden verkkoyhteyksien avulla. Etenkin langattomat verkkoyhteydet mahdollistavat lähes minkä tahansa laitteen käyttämisen IoT-laitteena. IP-verkkojen levinneisyys mahdollistaa laitteiden helpon yhdistettävyyden ennalta määritettyjen, tunnettujen verkkotekniikoiden avulla. Laskentatehon taloudellisuus viittaa tehokkaiden ja edullisten komponenttien saatavuuteen. Fyysisten komponenttien pienentyminen on vuorostaan avannut mahdollisuuden käyttää entistä pienempiä ja tehokkaampia komponentteja erilaisissa ratkaisuissa. (sama, 8.)

Kaksi viimeistä kehitysaskelta liittyvät kertyvän datan (big data) käsittelyyn. Datan analysoinnin kehittyminen näkyy kasvaneena laskentatehona, uudenlaisina algoritmeina ja pilvipalveluina. Edellä mainitut mahdollistavat entistä suurempien datamäärien käsittelemisen. Vuorostaan pilvipalveluiden yleistyminen yhdistää IoT-laitteet ja niiden keräämän datan Internetin mahdollistaman laajan saatavuuden avulla laajaan joukkoon palveluita. Nämä palvelut varastoivat, prosessoivat sekä analysoivat saamaansa dataa, jonka perusteella voidaan esimerkiksi ohjata IoT-laitteiden toimintaa. (sama, 8.)

3.1 Tietoturva

IoT-laitteiden suosio on kasvanut voimakkaasti. Lähes jokaisesta kodinkoneesta tai muusta kodin laitteesta on tarjolla tavallisen version lisäksi yhdistettävyydellä varustettu älykäs versio. Vaikka kodin laitteet ovat avautuneet maailmalle, tietoturvan olemassaolo unohdetaan usein. (Barcena & Wueest 2015, 3.)

Tietoturvayhtiö Symantec suoritti tutkimuksen kodin IoT-laitteiden tietoturvasta. Yhtiö analysoi 50 erilaista kodin älylaitetta, kuten älykkäitä termostaatteja, lukkoja, hehkulamppuja, palohälyttimiä, hubeja sekä energiansäästölaitteita. Tutkimuksessa havaitut haavoittuvuudet voivat yhtä hyvin altistaa myös muita verkkoon kytkettyjä laitteita, kuten murtohälyttimiä – tai vaikkapa älytelevisioita. Tutkimuksessa havaittiin puutteita tietoturvassa sekä laitteiden, että laitteiden hallintaohjelmistojen osalta. Haavoittuvuudet altistivat laitteet hyökkäyksille, jotka voivat pahimmassa tapauksessa johtaa koko laitteen hallinnan menettämiseen. (sama, 3, 7.)

Tietoturva on vielä suuremmassa roolissa yritysten osalta. Internet Societyn mukaan IoT-ympäristöstä voidaan luoda turvallinen ottamalla huomioon erilaisia näkökulmia. Lähtökohtaisesti tärkein on laitteen suunnitteluvaihe. IoT-laite on suunniteltava tietoturvalliseksi jo fyysistä laitetta sekä ohjelmistoa rakentaessa. Otetaan esimerkiksi ympäristö, jossa dataa keräämiä laitteita on satoja. Laitteet ovat keskenään lähes identtisiä. Mikäli yhdestä laitteesta löytyy haavoittuvuus, se altistaa potentiaalisesti kaikki muutkin laitteet samalle uhalle. Vuorostaan laitteiden datan lähettäminen pystytään turvaamaan kahdella tavalla. Autentikoinnilla laitteen ja palvelimen välinen yhteys saadaan turvallisemmaksi. Salaamalla lähetetty data voidaan varmistua datan eheydestä. (Internet Society 2015, 23–24.)

Tietoturvan kohtuullisuuden nimissä on kuitenkin tehtävä kompromisseja. On mietittävä mikä on kohtuullinen tietoturvan taso, ja mitä hankaluuksia liiallinen tietoturva tuo mukanaan. Voiko ympäristöä enää kehittää tai laajentaa tarpeen sitä vaatiessa. Entä pystyvätkö laitevalmistajat yhteistyöhön laitteiden lisätyn tietoturvan puolesta. Myös datan salaaminen vaatii lisää prosessointivoimaa ja salatun datan lähettäminen lisää kapasiteettia verkolta. (sama.)

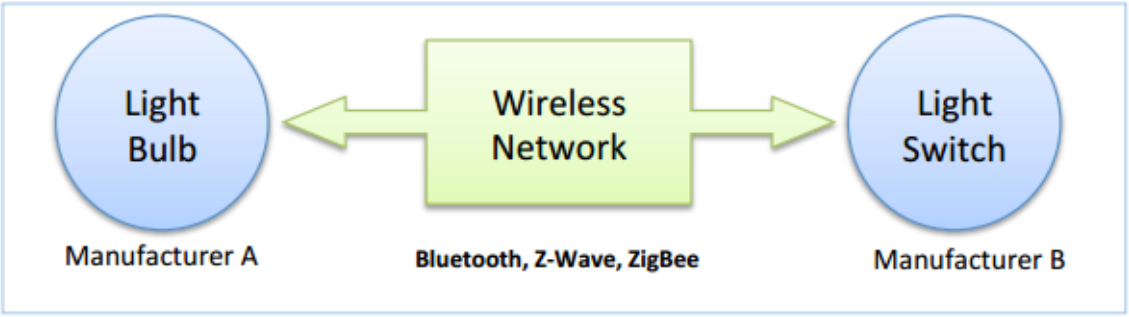
Aiemmin tässä kappaleessa kerrotun valossa voidaan todeta IoT:n tietoturvan tason määrittyvän pitkälti laitteiden käyttötarkoituksen mukaan. Kotiympäristössä kohtuullisen tason tietoturva riittää mainiosti. Organisaation tietoturva tuo puolestaan mukanaan huomattavasti enemmän haasteita tietoturvan kannalta.

3.2 Yhteysmallit

IoT-laitteet ovat yhteydessä Internetiin. Seuraavaksi käydään läpi, miten ja mihin laitteet ovat yhteydessä. Internet Societyn mukaan IoT-laitteet käyttävät kolmea eri yhteysmallia. (Internet Society 2015, 11.)

3.2.1 Device-to-Device -yhteysmalli

Laitteelta laitteelle -yhteysmallissa kaksi tai useampi IoT-laite ovat yhteydessä suoraan toisiinsa, ilman välityssovellusta tai -palvelinta. Laitteet käyttävät kommunikointiin IP-verkkoa tai Internetiä, hyödyntäen yleistyneitä tiedonsiirtoprotokollia, kuten Bluetoothia. Laitteelta laitteelle -yhteysmallia käytetään yksinkertaisuutensa vuoksi paljon esimerkiksi kodin automaatiossa. Kaistavaatimukset eivät myöskään ole korkeat, sillä dataa ei liiku isoja määriä eikä sitä lähetetä paljon kerrallaan. Laitteelta laitteelle -yhteysmalli esitetään kuviossa 3. (sama, 13.)



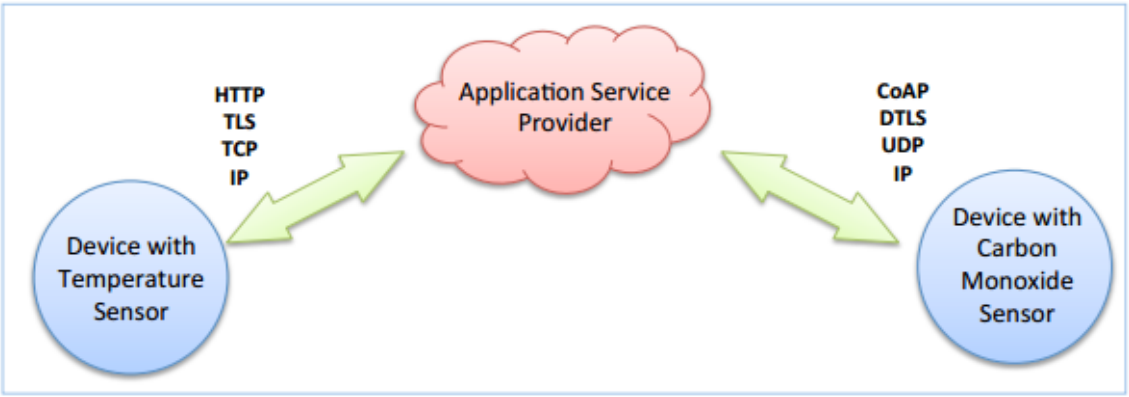
KUVIO 3. Device-to-Device -yhteysmalli (Internet Society 2015, 13)

Laitteelta laitteelle -yhteysmallissa on myös rajoittuvuuksia. Eri laitevalmistajat käyttävät usein omia yhteystekniikoita ja standardeja laitteissaan. Tämä aiheuttaa (useilla teknologian osa-alueilla tunnetun) ongelman; eri valmistajien laitteet eivät ole yhteensopivia keskenään. Tätä yhteysmallia ei myöskään pysty soveltamaan kovin monimutkaisissa toteutuksissa. (sama, 14.)

3.2.2 Device-to-Cloud -yhteysmalli

Laitteelta pilvipalveluun -yhteysmalli käyttää nimensä mukaisesti pilvipalvelua kommunikointiin. IoT-laite muodostaa yhteyden suoraan sovelluspalveluntarjoajaan (Application Service Provider,

ASP) Internet-yhteyden avulla. Tämän tyyppiset laitteet on yleensä rakennettu käyttämällä IP-verkkojen yhteystapoja, kuten Ethernetiä tai langattomia WLAN-verkkoja. Laitteelta pilvipalveluun -yhteysmalli esiintyy kuviossa 4. (sama, 14.)



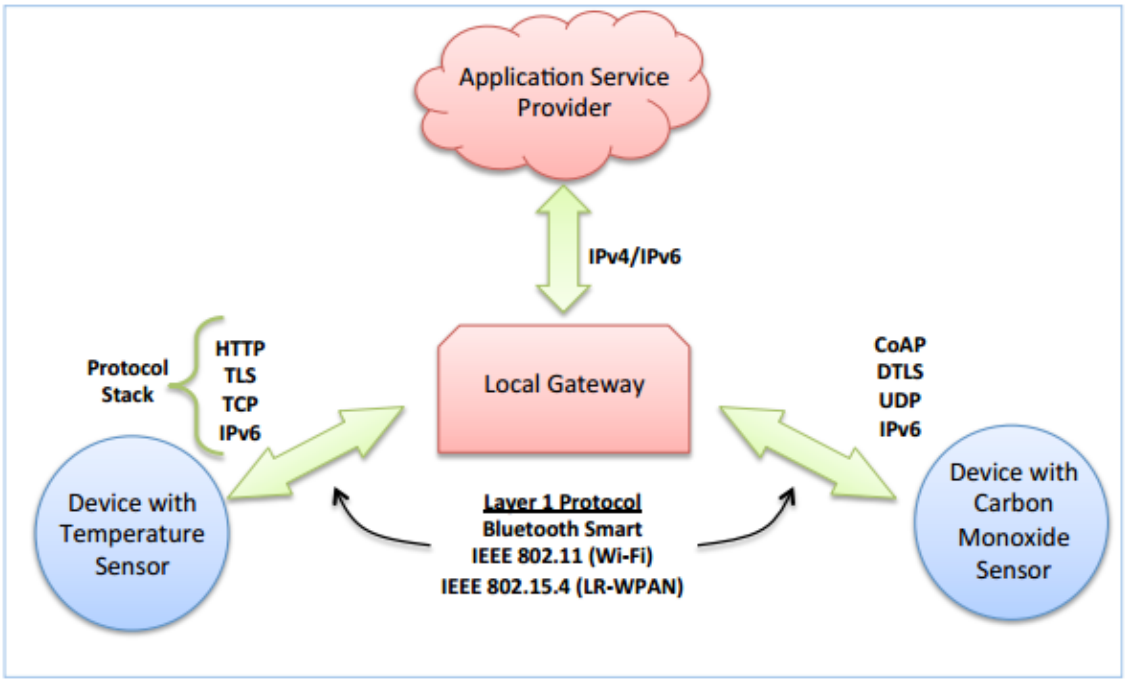
KUVIO 4. Device-to-Cloud -yhteysmalli (Internet Society 2015, 14)

Sovelluspalveluntarjoajaan yhteydessä olevat laitteet pystyvät monipuolisempiin tehtäviin, laitteelta laitteelle -yhteysmallia käyttäviin IoT-laitteisiin verrattuna. Käytännön esimerkkejä Device-to-Cloud -yhteysmallia hyödyntävistä laitteista ovat muun muassa älytelevisiot tai älytermostaatit. Laitteet tallentavat keräämäänsä dataa pilvipalveluun. Ratkaisusta riippuen dataa käytetään eri tavoilla; pelkkänä lokitietona, tai laitteen toiminnan ennakoimiseen ja säätämiseen. Sovelluspalvelimen olemassaolo mahdollistaa myös vaikkapa IoT-laitteiden hallitsemisen etäyhteyden avulla. (sama, 14.)

Myös tässä yhteysmallissa kohdataan aiemmin mainitun laitevalmistajien välillä vallitseva ristiriita. Laitteiden yhteensopivuus eri laitevalmistajien välillä on melko heikkoa. (sama, 15.)

3.2.3 Device-to-Gateway -yhteysmalli

Laitteelta yhdyskäytävään on esitetyistä yhteysmalleista edistynein ratkaisu. Laitteet ovat yhteydessä paikalliseen yhdyskäytävään, joka on vuorostaan muodostaa yhteyden pilvipalveluun. Yhdyskäytävänä toimivan sovelluksen käyttäminen tuo mukanaan lisää tietoturvaa, sekä muita toiminnollisuuksia, kuten toimivuuden useiden eri protokollien kanssa. Laitteelta yhdyskäytävään -yhteysmalli esitellään kuviossa 5. (sama, 15.)



KUVIO 5. Laitteelta yhdyskäytävään -yhteysmalli (Internet Society 2015, 15)

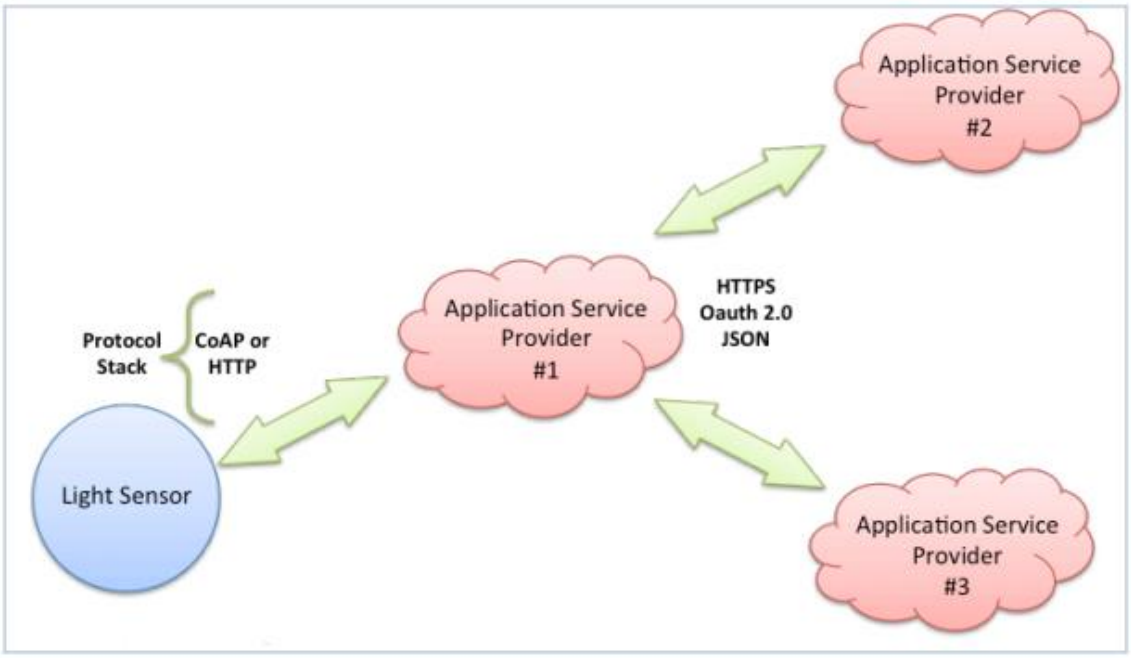
Tämä yhteysmalli on laajalti käytössä kokoonpanoissa, joissa IoT-laitteella ei ole toiminnallisuutta muodostaa yhteyttä suoraan palvelimeen. Hyvä esimerkki kyseistä yhteysmallia hyödyntävästä laitteesta on älyrannekkeet. Tällaiset laitteet ovat yhteydessä älypuhelimeen, johon asennettu sovellus toimittaa yhdyskäytävän virkaa. Älypuhelin vuorostaan kommunikoi pilvipalvelun kanssa Internetin välityksellä. (sama, 16.)

Yhdyskäytävän käyttäminen on myös tuonut ratkaisun aiemmin mainittuun, eri laitevalmistajien välisiin laitteiden yhteensopivuusongelmiin. Yritykset ovat tuoneet markkinoille laitteita, toimivat Device-to-Device tai Device-to-Cloud -tuotteiden kanssa. Laite luo yhdyskäytävän, johon aiemmin keskenään yhteen sopimattomat laitteet kytketään. (sama, 16.)

3.3 Taustajakopalvelut

IoT-laitteiden keräämän datan määrä voi olla todella suuri. Etenkin ympäristöissä, joissa on käytössä lukuisia IoT-laitteita esimerkiksi sensoreina, dataa kertyy valtavasti. Taustajakopalvelut mahdollistavat datan jakamisen, käsittelyn ja yhdistämisen muiden IoT-ympäristöjen kesken. Tätä tekniikkaa voidaan hyödyntää esimerkiksi tilanteessa, jossa yritys haluaa seurata kaikkien käytössä

olevien IoT-laitteidensa toimintaa. IoT-laitteen käyttötarkoituksella ei niinkään ole väliä – tarkoituksena voi olla vaikkapa seurata laitteiden virrankulutusta. Samaa tekniikkaa hyödyntämällä voidaan myös kerätä ja analysoida useiden eri IoT-ympäristöjen dataa yhteen paikkaan, tai verrata keskenään. Taustajakopalvelun toimintaa selkeyttävä visualisointi näkyy kuviossa 6. (sama, 17.)



KUVIO 6. Taustajakopalvelun toiminta (Internet Society 2015, 17)

Taustajakopalveluita avulla on mahdollista rikkoa yksittäisten IoT-ympäristöjen asettamia rajoja. IoT-ympäristöjä ja -laitteita pystytään hyödyntämään tehokkaammin, kun käytettävissä on useampia ympäristöjä. (sama, 17.)

4 BIG DATA

Big datalle on useita määritelmiä. Kuitenkin se kuvastaa – nimensä mukaisesti – erittäin suuria määriä dataa. Big data on käytännössä suuri määrä joko jäsennettyä tai jäsentämätöntä dataa. Sen ominaispiirteisiin kuuluu iso volyymi, jatkuvasti kasvava määrä sekä monipuolisuus. (SAS 2016, viitattu 9.10.2016.) Puolestaan De roos ym. (2012, 3) määrittelee big datan olevan dataa, jota ei pysty luonteensa vuoksi prosessoimaan tai analysoimaan tavanomaisin menetelmin tai työkaluin.

Big dataa kuvastamaan on vakiintunut niin sanottu kolmen V:n malli. Ensimmäisenä on datan suuri volyymi (volume). Dataa kertyy useista eri lähteistä, kuten liiketoiminnasta, sosiaalisesta mediasta, IoT-laitteista – lähes kaikkialta. Datamäärät ovat valtavia. Toisena on datan kiihtyvyys (velocity). Dataa virtaa ennennäkemättömällä vauhdilla ja tähän pitää vastata nopeasti. Monien datalähteiden lähettämä informaatio on sen tyyppistä, että se vaatii lähes reaaliaikaista reagointia. Kolmantena on monipuolisuus (variety); dataa tulee kaikissa formaateissa. Osa datasta on jäsennettyä, kuten taulukoita ja lukuja. Jäsentämätön data puolestaan tarkoittaa dokumentteja, sähköposteja tai vaikkapa videoita. (SAS 2016, viitattu 9.10.2016.)

Lisäksi SAS määrittelee kaksi ulottuvuutta edellä mainittujen lisäksi. Ne ovat vaihtelevuus (variability) sekä kompleksisuus (complexity). Vaihtelevuus tarkoittaa sitä, että dataa ei virtaa koko ajan tasaisesti. Välillä sitä tulee enemmän, välillä vähemmän. Vaihtelevuus käsittää myös, millaista dataa milloinkin tulee. Kompleksisuudella puolestaan tarkoitetaan vaikeutta hallita tulevaa dataa. Sitä tulee useista eri lähteistä, mutta ne voivat silti olla liitoksissa toisiinsa. Kompleksisuus täytyy olla hallinnassa, jotta dataa pystytään hyödyntämään ja käsittelemään tehokkaasti. (sama.)

Kaikessa työläydessään big data on kuitenkin hyvin arvokasta. Datan prosessoinnin jälkeen yritys saa tärkeää informaatiota esimerkiksi epäonnistumisista ja ongelmista lähes reaaliajassa. Dataa voidaan käyttää myös riskikartoitusten tekemiseen ja täten varautua tulevaisuuden uhkiin. Big data on myös erittäin arvokasta myynti- ja markkinointipuolella, esimerkiksi kuluttajien ostokäyttäytymistä tarkkailtaessa. (sama.)

5 MICROSOFT AZURE

Microsoft Azure on Microsoftin luoma pilvipalvelualusta. Azure on yksi Microsoftin kolmesta pilvipalveluvision peruspilareista. Alustan tarkoituksena on auttaa organisaatioita luopumaan perinteisistä palvelinkeskuksista, varmistaa datan laaja saatavuus sekä tarjota moderni alusta järjestelmien kehittämisen tueksi. Lisäksi, alustan tarkoituksena on vahvistaa ylläpidettävyyttä etäkäytön osalta, tinkimättä hallittavuudesta ja tietoturvasta. (Tulloch 2013, Introduction.)

Microsoft Azure tarjoaa pilvipalveluympäristön, jonka avulla on mahdollista rakentaa ja julkaista palveluita lähes kaikilla sovellusalustoilla ja rajapinnoilla. Laaja valikoima heti käytettävissä olevia palveluita mahdollistavat toiminnot aina yrityksen verkkosivustosta monimutkaisten SQL-tietokantojen ylläpitämiseen. Microsoft Azuren palvelimet sijaitsevat Microsoftin palvelinkeskuksissa, joita löytyy ympäri maailman. Siispä kaikki palvelut toimivat ovat aina luotettavasti saatavilla, varustettuna korkealla suorituskyvyllä ja minimaalisilla viiveillä. (sama, 4.)

Palvelun hinnoittelu perustuu käytön mukaiseen, kuukausittaiseen veloitukseen. Pay-as-you-go -politiikkaa noudattava Azure seuraa palveluiden käyttöastetta, ja ilmoittaa 30 päivää ennen maksupäivää, mikäli hinnoitteluun on tehty muutoksia. Hinta määräytyy muun muassa käytössä olevien palveluiden, fyysisen sijainnin ja palveluiden käyttöasteen mukaan. (Microsoft Azure 2016c, viitattu 9.9.2016.)

5.1 Historia

Microsoft julkaisi pilvipalvelualusta Windows Azuren lokakuussa 2008 (Rouse 2016, viitattu 24.8.2016). Palvelu ei vielä silloin tavoittanut suosiota. Vuorostaan Clarke mainitsee artikkelissaan arvostelleensa Windows Azuren ensimmäisen kerran vuonna 2011, eikä pitänyt näkemästään. Hän kuvailee sisältöä puutteelliseksi, sekä web-käyttöliittymää huonosti toimivaksi. Clarke kuitenkin pitää vuotta 2011 käännekohtana, jolloin Scott Guthrie otti haltuun Azuren kehitystyön. Hänen johdollaan web-käyttöliittymästä tehtiin kevyt ja toimiva, hyödyntämällä HTML5-tekniikkaa. Vuodesta 2011 eteenpäin palvelua on kehitetty, paranneltu ja laajennettu jatkuvasti. (Clarke 2015, viitattu 24.8.2016.)

Vuonna 2014 palvelu uudelleen brändättiin Microsoft Azureksi (Rouse 2016, viitattu 24.8.2016). Samana vuonna julkaistiin myös uusi, paranneltu web-käyttöliittymä, Azure Portal. Clarke kutsuu artikkelissaan uutta käyttöliittymää taideteokseksi. (Clarke 2015, viitattu 24.8.2016.)

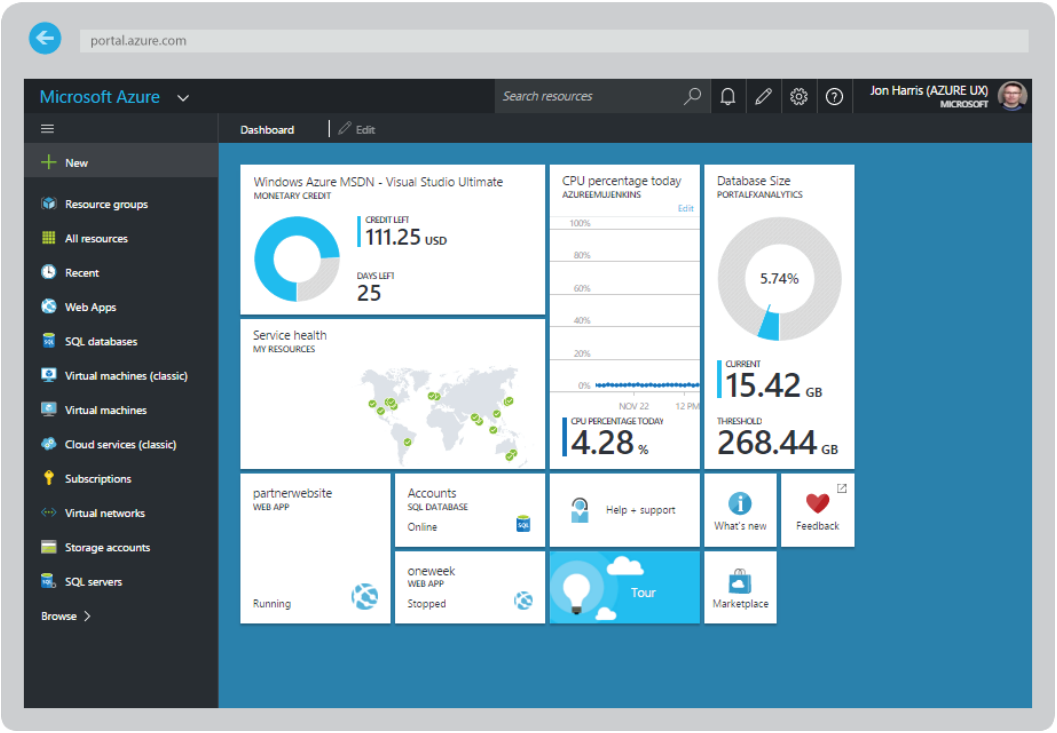
5.2 Ominaisuudet

Azuren tarjoaa lukuisia eri tarpeisiin vastaavia palveluita käyttäjilleen. Palvelut on jaettu neljään pääkategoriaan: pilvilaskentapalvelut, datapalvelut, sovelluspalvelut ja verkkopalvelut. Pilveen siirtymistä suunnitellessa on hyvä tuntea perusteet Azuren eri osa-alueista ja ominaisuuksista, jotta käyttöönotto sujuu vaivattomasti ja tehokkaasti. (Collier & Shahan 2015, 20.)

5.2.1 Azure Portal

Azure Portal on hallintaportaali, jonka avulla pilvipalvelun resursseja on helppo hallita. Sen kautta saa helposti tietoa käytössä olevista palveluista ja palveluiden tilasta. Portaalin avulla pystyy ottamaan palveluita käyttöön sekä navigoimaan paikasta toiseen. (Microsoft Azure. 2016b, viitattu 26.8.2016.)

Hallintaportaali on helppo tapa nähdä perustietoja pilvipalveluympäristöstä. Näkymä koostuu navigointipalkista sekä kustomoitavasta alueesta. Navigointipalkin avulla pääsee helposti käsiksi palvelun ominaisuuksiin. Kustomoitavalle alueelle voidaan lisätä haluttuja toimintoja, kuten käytössä olevien palveluiden tilan, käytetyn laskentatehon tai oman tilauksen tietoja. Kuviossa 7 näkyy Azure Portalin päänäkymä. Azure Portal löytyy osoitteesta https://portal.azure.com/. (sama.)

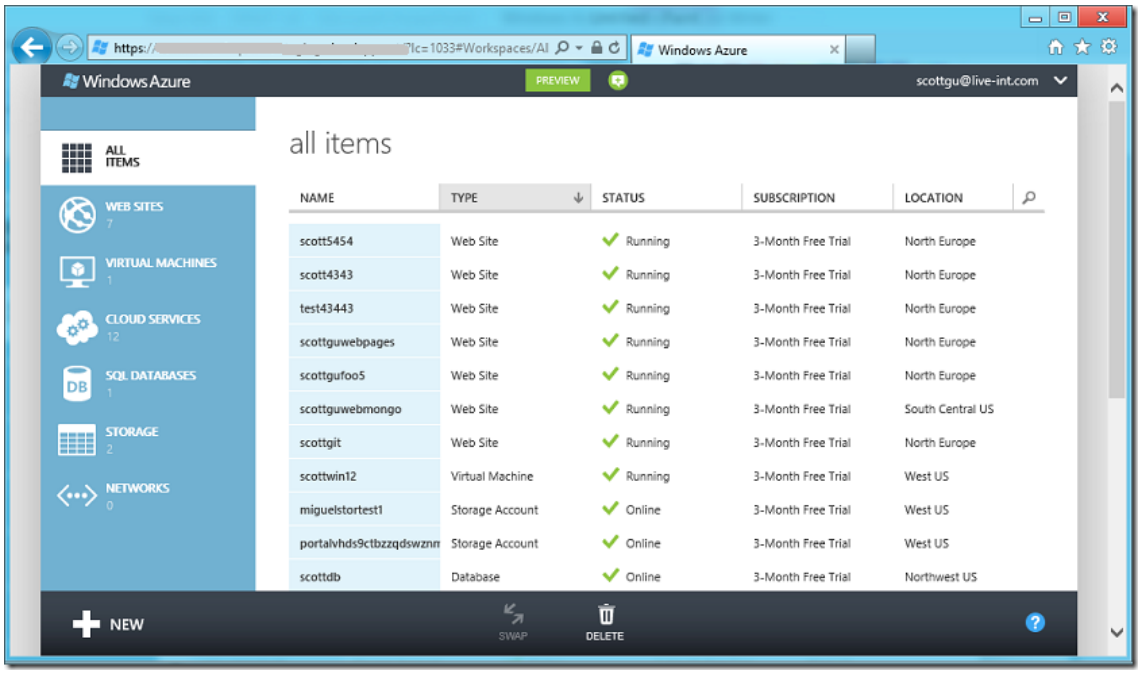


KUVIO 7. Azure Portalin oletusnäkymä (Microsoft Azure 2016b, viitattu 26.8.2016)

Palvelu on tällä hetkellä siirtymävaiheessa uuden Azure Portalin ja aiemman Azure Management Portalin välillä. Kaikkia ominaisuuksia, esimerkiksi Active Directory -palvelua ei pysty vielä täysin hallinnoimaan uuden Azure Portalin kautta. (sama, viitattu 5.9.2016.) Tästä syystä on olennaista käydä läpi myös vanhempi Management Portal.

5.2.2 Azure Management Portal

Management Portal on Azure Portalia edeltävä hallintaliittymä. Sen kautta pystytään luomaan, hallitsemaan ja valvomaan käytössä olevia Azure pilvipalveluratkaisuja. Management Portalin asettelu on saman kaltainen kuin Azure Portalissa. Vasemmassa laidassa on navigointipalkki ja näytön keskiosassa lista käytössä olevista palveluista, kuten virtuaalikoneista. Aivan kuten uuden Azure Portalin kautta, myös Management Portalin avulla pystytään hallitsemaan Azuren tarjoamia palveluita. Management Portal löytyy osoitteesta https://manage.windowsazure.com/. Portaalin päänäkymä esitellään kuviossa 8. (Guthrie 2012, viitattu 9.9.2016.)

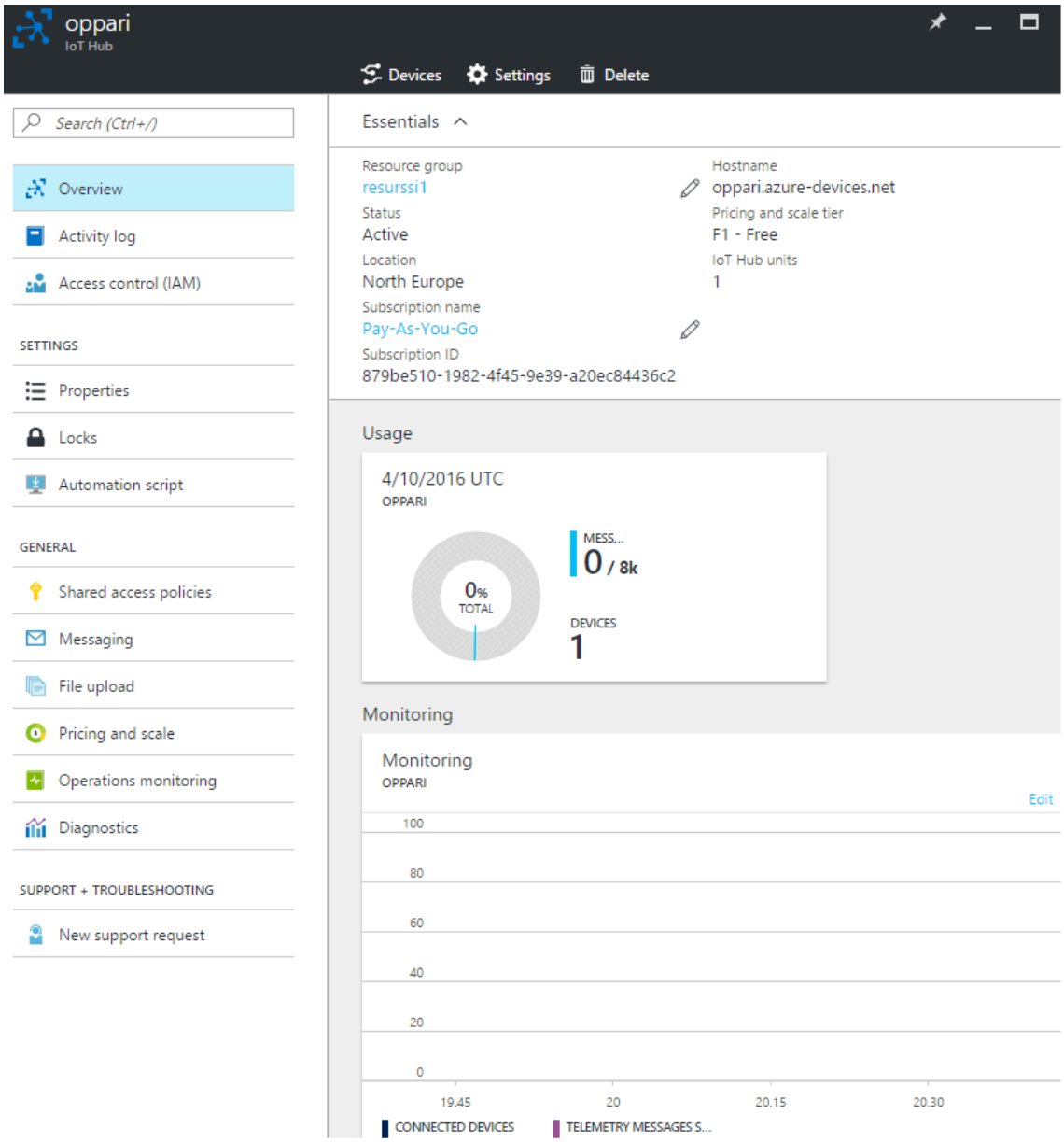


KUVIO 8. Azure Management Portal päänäkymä (Guthrie 2012, viitattu 9.9.2016)

5.2.3 Azure IoT Hub

Azure IoT Hub on Microsoftin pilvipalveluratkaisu esineiden internet -ympäristön toteuttamiseen. Nimensä mukaisesti IoT Hub on joukko IoT-laitteiden hallintaan, valvontaan ja ohjaamiseen tarkoitettuja palveluita. Kyseessä on PaaS-tyypin pilvipalvelu. IoT Hub sisältää laiterekisterin, tietovaraston, tietoturvaominaisuuksia sekä yksinkertaisia valvonta- ja diagnostiikkatyökaluja. Palvelusta löytyy myös palvelurajapinta IoT-ohjelmistokehityksen tueksi. IoT Hubin päänäkymä on nähtävillä kuviossa 9. (Microsoft Azure 2016a, viitattu 4.10.2016.)

IoT Hubia voidaan pitää IoT-laitteiden lähtöpisteenä Azuren palveluissa. Käyttötarkoituksesta riippuen Azure tarjoaa myös useita muita palveluita, joita voidaan tarvittaessa hyödyntää IoT Hubin yhteydessä. (sama.)



KUVIO 9. Azure IoT Hub päänäkymä (Microsoft Azure 2016a, viitattu 4.10.2016)

6 KÄYTÄNNÖN OSUUS

Käytännön osuudessa luotiin IoT-laite- ja hallintaympäristö Microsoft Azureen, sekä erilaisia ympäristön testaamiseen tarkoitettuja ohjelmia. Tarvittavat testaustarkoitukseen käytettävät ohjelmat rakennettiin Microsoft Azuren ohjetietokannan, sekä Microsoft Visual Studion avulla. Näin varmistuttiin koodin oikeellisuudesta ja testiohjelmien toimivuudesta. Ohjelmat luotiin C# -ohjelmointikielellä. Luodut ohjelmat käyttivät Azureen liittyviä komentoja, jotka vaativat toimiakseen pakettivarastoja. Tarvittavat pakettivarastot asennettiin Visual Studion NuGet -pakettienhallinnan avulla.

Testausympäristön alustana käytettiin 64-bittistä Windows 10 Pro -käyttöjärjestelmää, joka oli asennettuna tavalliseen kuluttajatyöasemaan. Työasemassa oli Intel Core i5-2500K -prosessori, 16Gt DDR3 RAM-muistia, sekä SSD-kovalevy. Työasema oli myös yhteydessä Internetiin, jotta yhteys Azureen oli mahdollinen. Suorituskykyongelmia tällä kokoonpanolla ei kohdattu.

IoT-laitetta mallintava ohjelma (kappale 6.3) toteutettiin pelkkänä simulaationa, sillä oikea, anturia mittausarvojen keräämiseen käyttävä laite voisi toimia samalla koodilla. Oikeaa anturia käytettäessä koodia olisi pitänyt muokata siten, että mittaustulokset tulevat anturilta, ei satunnaislukugeneraattorilta. Simulaatio vastasi siis käytännön ympäristöä ainoastaan sillä poikkeuksella, että laitteen tuottamat arvot luotiin ohjelman sisällä. Myöskään datan sisältö ei ollut tämän simulaation kannalta olennaista – tästä syystä mittausanturin käyttöön ei nähty tarvetta.

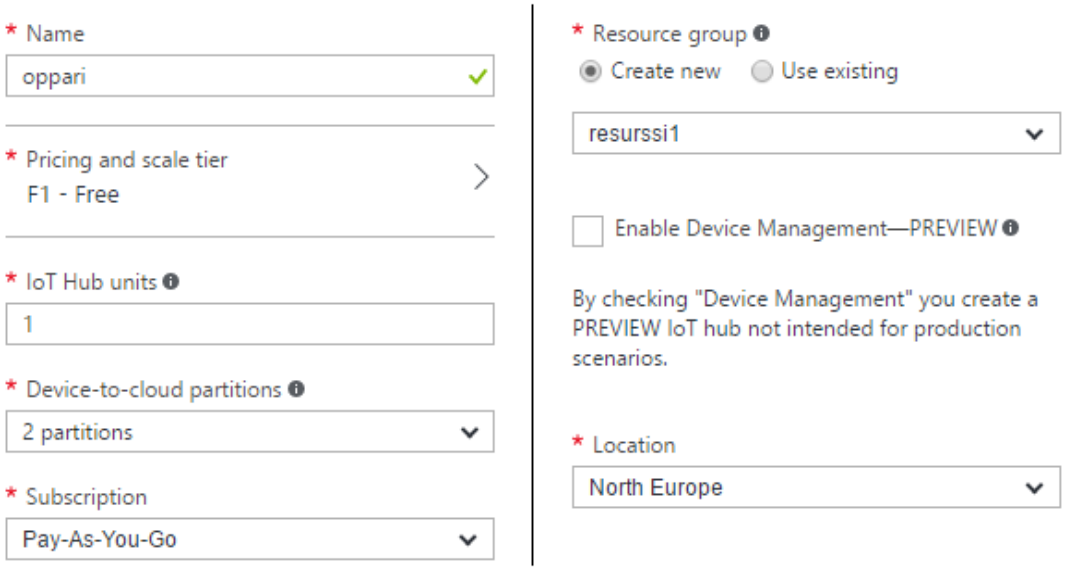
Teoriaosuudessa mainittu tietoturva oli yksi huolenaihe ympäristössä, jossa dataa siirretään laitteelta Internetin yli pilveen. Tässä ympäristössä tietoturvan kannalta olennaisin osuus oli kappaleessa 6.2 käsitelty identiteetin luominen. Identiteetin avulla pilvipalvelun ja konsoliohjelmien välinen yhteys autentikoitiin, luomalla laitteen ja palvelimen välille yksilöllinen käyttöoikeus.

6.1 Azure IoT Hub

Ensiksi käyttöönotettiin Azure IoT Hub (kuvio 10). Tämä tapahtui luonnollisesti Azure Portalin kautta. IoT Hubille annettiin nimeksi oppari. Hinta- ja skaalautuvuusluokaksi valittiin F1 – Free, joka riitti testaustarkoitukseen varsin hyvin. Device-to-cloud -osioiden määrä pidettiin myös nimimissä (2 osiota). Testiympäristössä käytettiin vain yhtä osiota. Olemassa olevaa Recource groupia ei

ollut, joten myös tällainen piti luoda. Fyysiseksi sijainniksi valittiin North Europe, jotta Azuren käyttämä datakeskus oli mahdollisimman lähellä IoT-laitteiden fyysistä sijaintia.

IoT Hubin luomisen yhteydessä IoT-alustalle määritettiin nimi. Tästä nimestä Azure generoi IoT Hubiin yhteydenottoon käytettävän URI-osoitteen. Tässä tapauksessa IoT-alustan yhteysosoitteeksi muodostui siis oppari.azure-devices.net.



KUVIO 10. IoT Hubin luonti

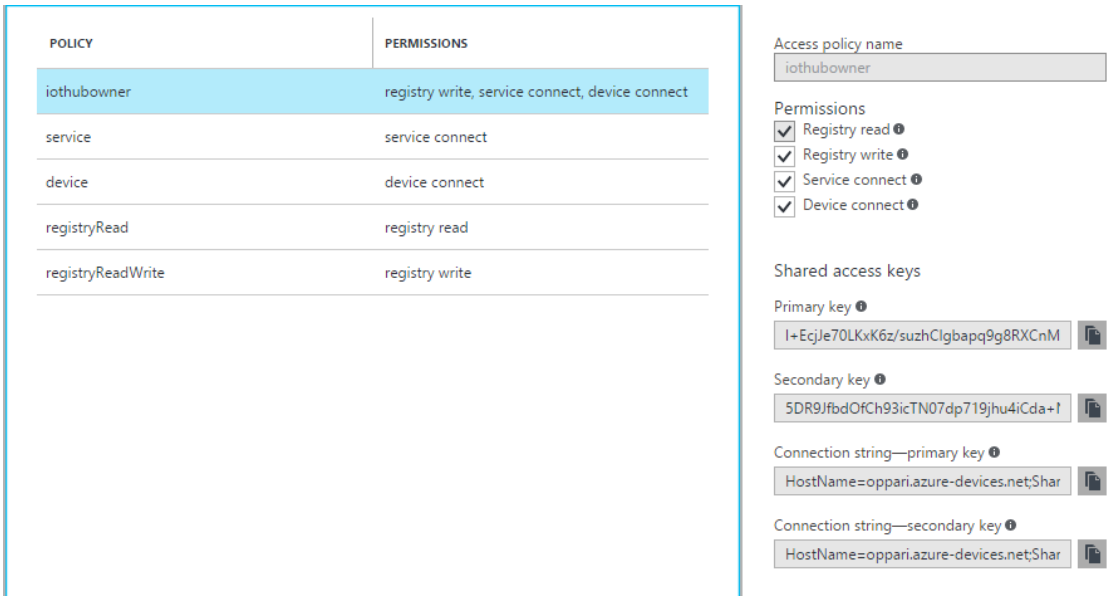
6.2 Laitteen identiteetin luominen

Jokaiselle IoT Hubiin yhteyden ottavalle ohjelmalle piti luoda identiteetti, ennen kuin ne voitiin yhdistää IoT Hubiin. Tähän tarkoitukseen luotiin uusi konsoliohjelma Microsoft Visual Studiolla. Projektin tyypiksi valittiin C#:lla toteutettava konsoliohjelma. Projekti nimettiin toimintansa mukaisesti nimellä CreateDeviceIdentity (liite 1).

Luotuun projektiin lisättiin NuGet pakettienhallinnasta pakettivarasto nimeltä Microsoft.Azure.Devices.Client. Tarvittavat osat pakettivaraston sisällöstä otettiin käyttöön lisäämällä ohjelman alkuun komennot using *Microsoft.Azure.Devices*.Client ja *using Microsoft.Azure.Devices*.Common.Exceptions.

Jotta identiteetin luominen onnistui ja halutut ohjelmat saivat muodostettua yhteyden IoT Hubiin, piti muuttujan connectionString arvoksi määrittää IoT Hubissa määritetty pääsyavain (kuvio 11).

Tämä uniikki avain mahdollisti datan turvallisen siirron palvelimen ja IoT-laitteen välillä. Ainoastaan ne laitteet, joille määritettiin identiteetti tämän pääsyavaimen avulla, pystyivät olemaan yhteydessä IoT Hubiin. Avain haettiin IoT Hubista. Valikosta Shared Access Policies valittiin Azuren automaattisesti luoma iothubowner-niminen käyttäjä. Käyttäjän alta kopioitiin arvo Connection string – primary key. Tämä tekstijono syötettiin arvoksi connectionString-muuttujalle.



KUVIO 11. Pääsyavaimen sijainti Azure IoT Hubissa

Tämän jälkeen ohjelma oli valmis suoritettavaksi. Ohjelman suorittaminen avasi konsoli-ikkunan, jonka tulosteena oli ohjelman generoima laiteavain (kuvio 12).

KUVIO 12. CreateDeviceIdentity-konsoliohjelman generoima laiteavain

6.3 IoT-laitteen simulointi

IoT-alustan testaamista varten piti luoda IoT-laite (liite 2). Laitetta simuloimaan rakennettiin Visual Studiolla SimulateDevice-niminen konsoliohjelma. Projekti luotiin C# konsoliohjelmana. Konsoliohjelman tarkoitus oli simuloida tuulen nopeutta mittaavaa sensoria. Ohjelma arpoi sekunnin välein satunnaisluvun esittämään mittaustulosta ja lähetti sen Azure IoT Hubiin. Lähetettävä data sisälsi mittausajankohdan, laitteen nimen, sekä itse mittaustuloksen. Laitteen toiminnasta otettiin kuvakaappaus, joka on nähtävillä kuviossa 13.

Luotuun projektiin lisättiin NuGet pakettienhallinnasta pakettivarasto nimeltä Microsoft.Azure.Devices.Client. Komennot, joiden avulla otetaan yhteyttä Azureen, tulivat lisätyn pakettivaraston mukana. Pakettivaraston sisältö otettiin käyttöön lisäämällä ohjelman alkuun komento using Microsoft.Azure.Devices.Client.

Jotta ohjelma pystyi lähettämään dataa IoT Hubiin, piti koodiin määrittää kaksi asiaa; aiemmin luotu laiteavain, sekä IoT Hubin URI-osoite. URI-osoitteen avulla ohjelma määritettiin ottamaan yhteyttä haluttuun pilvipalveluun. Laiteavain varmensi ”laitteen” ja IoT Hubin välisen yhteyden. Connection string haettiin IoT Hubin päänäkymästä. Connection string oppari.azure-devices.net määritettiin iotHubUri-nimisen muuttujan arvoksi.

KUVIO 13. SimulateDevice-konsoliohjelman lähettämää dataa

Kun ohjelma muodosti onnistuneesti yhteyden IoT Hubiin, ”laitteelle” muodostettiin laiteavaimen ja laitteen nimen yhdistelmästä tietue IoT Hubin identiteettirekisteriin. Tätä toimintoa voi verrata käyttäjätunnukseen ja salasanaan.

6.4 Datan hakeminen pilvestä

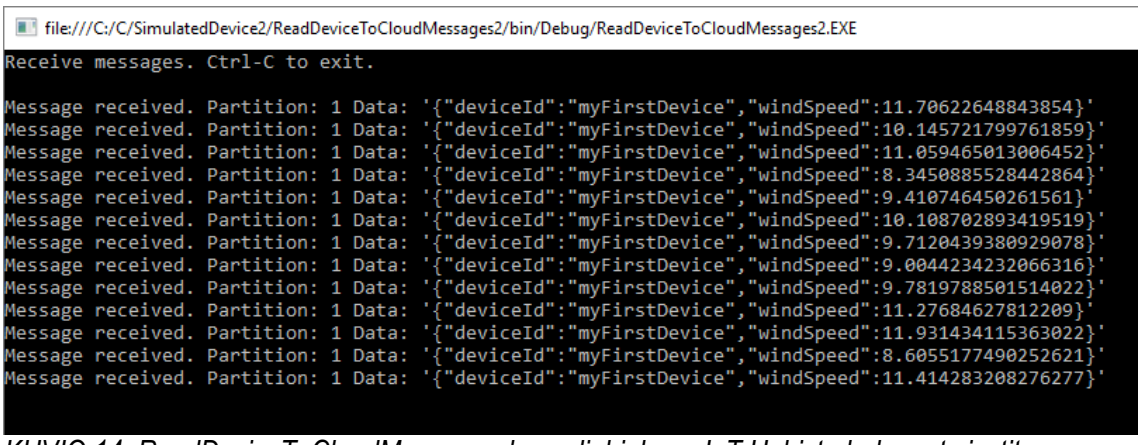
Seuraavaksi testattiin IoT Hubiin tallennetun datan hakemista pilvipalvelusta. Tähän tarkoitukseen rakennettiin C#-konsoliohjelma nimeltä ReadDeviceToCloudMessages (liite 3). Ohjelman tarkoituksena oli hakea ja näyttää IoT-laitteiden tallentamat viestit IoT Hubista, jotta voitiin varmistua tietojen olemassaolosta ja oikeellisuudesta.

Koska kyseessä oli ohjelma, joka tarvitsi enemmän kuin kirjoitusoikeudet IoT Hubiin (vrt. kappale 6.3, SimulateDevice-ohjelma), konsoliohjelmaan piti määrittää tiedot autentikoinnille. Muuttujan connectionString arvoksi piti määrittää URI-osoitteen lisäksi myös käyttäjätunnus ja jaettu avain. Kyseinen merkkijono, Connection string, haettiin IoT Hubista (kuvio 9).

Luotuun ohjelmaan lisättiin NuGet pakettienhallinnasta pakettivarasto nimeltä Microsoft.ServiceBus.Messaging. Komennot, joiden avulla otetaan yhteyttä Azureen, tulivat lisätyn pakettivaraston mukana. Pakettivaraston sisältö otettiin käyttöön lisäämällä ohjelman alkuun komento using Microsoft.ServiceBus.Messaging.

Konsoliohjelma oli rakennettu siten, että se hakee vain ne viestit, jotka lisätään IoT Hubiin konsoliohjelman ollessa käynnissä. Testin aikana SimulateDevice-konsoliohjelma generoi ja lähetti dataa pilveen, kun taas ReadDeviceToCloudMessages-konsoliohjelma haki uusia saapuvia viestejä pilvestä.

Toimintaa testattiin myös muokkaamalla ohjelman hakuehdon aikamääritystä siten, että ohjelma hakee kaikki palvelimella olevat viestit – ei pelkästään suorittamisen aikana lisättyjä. Ohjelma toimi halutulla tavalla molemmissa testeissä (kuvio 14).



KUVIO 14. ReadDeviceToCloudMessages-konsoliohjelman IoT Hubista hakemat viestit

7 POHDINTA

Opinnäytetyön tarkoituksena oli tutkia IoT-laiteympäristön toimintaa hyödyntämällä pilvipalveluratkaisuja. Työn tavoitteena oli luoda toimiva testausympäristö. Aihevalinta perusteltiin työn ajankohtaisuudella. Toteutusalustaksi valittiin Microsoftin tuotteet, yrityksen suuren markkina-aseman ja suosion vuoksi. Edellä mainitut valinnat muodostivat opinnäytetyön aiherajauksen. Teoriaosuuden aiherajaus perusteltiin edellä mainituilla valinnoilla. Työn käytännön osuuden kannalta oli olennaista käydä läpi pilvipalveluiden ja esineiden internetin teoriaa, sekä Microsoft Azurea. Myös mainінta big datasta koettiin tarpeelliseksi.

Teoriaosuus muodosti tarvittavan tietoperustan työn käytännön osuutta varten. Pilvipalveluiden sekä esineiden internetin teoria käytiin läpi tarpeellisilta osilta. Microsoft Azuren teoria, sekä sen ominaisuudet käytiin käytännön osuudessa käytettyjen palveluiden osalta. Big data ei ollut olennainen osa tätä työtä, joskin on huomionarvoista ymmärtää, että IoT-laitteiden tuottama data on käytännössä big dataa. Tästä syystä myös se otettiin osaksi tietoperustaa. Teoriaosuus rajattiin siten, että lukija sai kattavan pohjustuksen työn käytännön osuutta varten.

Käytännön osuudessa rakennettiin testausympäristö IoT-laitteille. Kaikki testit saatiin suoritettua onnistuneesti. Pilvipalvelualusta luotiin ja konfiguroitiin onnistuneesti. Laitteiden identiteetin luonti onnistui kuten pitääkin. IoT-laite saatiin ottamaan yhteyttä pilvipalveluun ja lähettämään dataa. Lähetetty data näkyi pilvipalvelussa ja se saatiin haettua takaisin prosessointia varten.

Varsinaisia ongelmia työssä ei tullut vastaan. Tekstin tuottaminen onnistui vaivattomasti ja työ eteni aikataulussa. Microsoftin tarjoama ohjetietokanta oli kattava, joten käytännön osuuden testausympäristön rakentaminen onnistui myös ilman suuria ongelmia. Itse prosessi kehitti sekä kirjoitus- että tiedonhakutaitoja ja antoi kattavat valmiudet IoT-laitteiden ja pilvipalveluiden maailmaan.

Koulun käydyiltä kursseilta saatu osaaminen korostui opinnäytetyöprosessissa. Työ toimi ikään kuin osaamisen mittarina aiemmin hankitun tiedon soveltamisen, sekä rajattuun aihepiiriin syventymisen muodossa. Työtä ei välttämättä tarjoa lukijalle käyttöä sellaisenaan, mutta sitä voi hyödyntää mallina tai tietolähteenä, vastaavissa käyttötarkoituksissa.

LÄHTEET

Amazon Web Services. 2016. What is Cloud Computing. Viitattu 11.8.2016. https://aws.amazon.com/what-is-cloud-computing/.

Barcena, M. & Wueest, C. 2015. Security Response. Insecurity in the Internet of Things. Versio 1.0. https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-the-internet-of-things-en.pdf.

Cisco Support Community. 2012. Emulate Internet with PT-Cloud in Packet Tracer. Viitattu 26.8.2016. https://supportforums.cisco.com/discussion/11032411/emulate-internet-pt-cloud-packet-tracer.

Clarke, G. 2015. A Brief History of Microsoft Azure. Viitattu 24.8.2016. http://garyclarke.us/technology/a-brief-history-of-microsoft-azure/.

Cloud Standards Customer Council. 2015. Practical guide to Platform-as-a-Service. Viitattu 24.8.2016. http://www.cloud-council.org/CSCC-Practical-Guide-to-PaaS.pdf.

Collier, M. & Shahan, R. 2015. Fundamentals of Azure: Microsoft Azure Essentials. Washington: Microsoft Press.

De roos, D., Deutsch, T., Eaton, C., Lapis, G. & Zikopoulos, P. Understanding Big Data. 2012. Analytics for Enterprise Class Hadoop and Streaming Data. New York: McGraw-Hill Companies.

Erl, T. 2013. Cloud Computing: Concepts, Technology & Architecture. New Jersey: Prentice Hall.

Guthrie, S. 2012. Meet the New Windows Azure. Viitattu 9.9.2016. http://weblogs.asp.net/scottgu/meet-the-new-windows-azure.

Internet Society. 2015. The Internet of Things: An Overview. Understanding the Issues and Challenges of a More Connected World.

Kavis, M. 2014. Architecting the Cloud. Design Decisions for Cloud Computing Service Models. New Jersey: Wiley.

National Institute of Standards and Technology. 2012. Cloud Computing Synopsis and Recommendations. Viitattu 18.8.2016. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf.

National Institute of Standards and Technology. 2016. NIST General Information. Viitattu 25.8.2016. http://www.nist.gov/public_affairs/general_information.cfm.

Microsoft Azure. 2016a. Azure IoT Hub. Viitattu 4.10.2016. https://portal.azure.com/.

Microsoft Azure. 2016b. Azure Portal. Viitattu 26.8.2016. https://portal.azure.com/.

Microsoft Azure. 2016c. Azure Pricing. Viitattu 9.9.2016. https://azure.microsoft.com/en-us/pricing/.

Rouse, M. 2016. Definition: Microsoft Azure (Windows Azure). Viitattu 24.8.2016. http://searchcloudcomputing.techtarget.com/definition/Windows-Azure.

SAS. 2016. What Is Big Data. Viitattu 9.10.2016. http://www.sas.com/en_us/insights/big-data/what-is-big-data.html.

Tulloch, M. 2013. Introducing Windows Azure for IT professionals. Washington: Microsoft Press.

Wikipedia. 2016. Cloud computing. Viitattu 26.8.2016. https://en.wikipedia.org/wiki/Cloud_computing.

```
using Microsoft.Azure.Devices;
using Microsoft.Azure.Devices.Common.Exceptions;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CreateDeviceIdentity
{
    class Program
    {
        static RegistryManager registryManager;
        static string connectionString = "HostName=oppari.azure-de-
vices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=l+EcjJe70LKxK6z/su-
zhCIgbapq9g8RXCnMYe7zl1rY=";

        private static async Task AddDeviceAsync()
        {
            string deviceId = "myFirstDevice";
            Device device;
            try
            {
                device = await registryManager.AddDeviceAsync(new Device(de-
viceId));
            }
            catch (DeviceAlreadyExistsException)
            {
                device = await registryManager.GetDeviceAsync(deviceId);
            }
            Console.WriteLine("Generated device key: {0}", device.Authentica-
tion.SymmetricKey.PrimaryKey);
        }

        static void Main(string[] args)
        {
            registryManager = RegistryManager.CreateFromConnectionString(connec-
tionString);
            AddDeviceAsync().Wait();
            Console.ReadLine();
        }
    }
}
```

```
using System.IO;
using Microsoft.ServiceBus.Messaging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ProcessD2CInteractiveMessages
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Process D2C Interactive Messages app\n");

            string connectionString =
"2ssrQrtKsa11faUhB3i6l5DYKB7A0t9wXN2npR3Joig=";
            QueueClient Client = QueueClient.CreateFromConnectionString(connec-
tionString);

            OnMessageOptions options = new OnMessageOptions();
            options.AutoComplete = false;
            options.AutoRenewTimeout = TimeSpan.FromMinutes(1);

            Client.OnMessage((message) =>
            {
                try
                {
                    var bodyStream = message.GetBody<Stream>();
                    bodyStream.Position = 0;
                    var bodyAsString = new StreamReader(bodyStream, Encod-
ing.ASCII).ReadToEnd();

                    Console.WriteLine("Received message: {0} messageId: {1}",
bodyAsString, message.MessageId);

                    message.Complete();
                }
                catch (Exception)
                {
                    message.Abandon();
                }
            }, options);

            Console.WriteLine("Receiving interactive messages from SB queue...");
            Console.WriteLine("Press any key to exit.");
            Console.ReadLine();
        }
    }
}
```

```
using Microsoft.ServiceBus.Messaging;
using System.Threading;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReadDeviceToCloudMessages
{
    class Program
    {
        static string connectionString = "HostName=oppari.azure-de-
vices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=l+EcjJe70LKxK6z/su-
zhCIgbapq9g8RXCnMYe7zl1rY=";
        static string iotHubD2cEndpoint = "messages/events";
        static EventHubClient eventHubClient;
        private static async Task ReceiveMessagesFromDeviceAsync(string parti-
tion, CancellationToken ct)
        {
            var eventHubReceiver = eventHubClient.GetDefaultConsumerGroup().Cre-
ateReceiver(partition, DateTime.UtcNow);
            while (true)
            {
                if (ct.IsCancellationRequested) break;
                EventData eventData = await eventHubReceiver.ReceiveAsync();
                if (eventData == null) continue;

                string data = Encoding.UTF8.GetString(eventData.GetBytes());
                Console.WriteLine("Message received. Partition: {0} Data: '{1}'",
partition, data);
            }
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Receive messages. Ctrl-C to exit.\n");
            eventHubClient = EventHubClient.CreateFromConnectionString(connec-
tionString, iotHubD2cEndpoint);

            var d2cPartitions = eventHubClient.GetRuntimeInformation().Parti-
tionIds;
            CancellationTokenSource cts = new CancellationTokenSource();

            System.Console.CancelKeyPress += (s, e) =>
            {
                e.Cancel = true;
                cts.Cancel();
                Console.WriteLine("Exiting...");
            };

            var tasks = new List<Task>();
            foreach (string partition in d2cPartitions)
            {
                tasks.Add(ReceiveMessagesFromDeviceAsync(partition, cts.Token));
            }
            Task.WaitAll(tasks.ToArray());
        }
    }
}
```