VAASAN YLIOPISTO

TEKNILLINEN TIEDEKUNTA

AUTOMAATIOTEKNIIKKA

Jaakko Yli-Luukko

IOT-SOVELLUKSEN TOTEUTUS, KÄYTTÖASTEEN SEURANTA

Diplomityö, joka on jätetty tarkastettavaksi diplomi-insinöörin tutkintoa varten Vaasassa 28.11.2017.

Työn valvoja Jarmo Alander

Työn ohjaaja Petri Välisuo

ALKULAUSE

Tämä diplomityö on Vaasan yliopiston teknillisen tiedekunnan diplomi-insinöörin tutkintoani varten. Tämä diplomityö on tehty Lapualle RiimaPlan Oy:lle. Työn tuloksista on jatkokehitetty tuote nimeltä *HitScan* ja tuotetta myymään on perustettu yritys nimeltä Tehotec Oy, jossa olen myös osakkaana.

Kiitän RiimaPlan Oy:n Juha Lemposta ja Pekka Antilaa opastuksesta ja mahdollisuudesta diplomityöhön ja Esa Mäkelää teknisestä tuesta. Kiitän diplomityöni valvojaa professori Jarmo Alanderia ja työni ohjaajaa Petri Välisuota opastuksesta diplomityön kirjoituksen aikana. Lopuksi haluaisin kiittää erityisesti perhettäni kannustuksesta ja ajan järjestämisestä opinnoilleni.

Ilmajoella 28.11.2017

Jaakko Yli-Luukko

SISÄLLYSLUETTELO sivu

SYMBOLI- JA LYHENNELUETTELO

API	Application Programming Interface, ohjelmointirajapinta
CMS	Content Management System, sisällönhallintajärjestelmä
CRM	Customer Relationship Management, asiakkuudenhallinta
CSS	Cascading Style Sheets
ERP	Enterprise Resource Planning, toiminnanohjausjärjestelmä
GPIO	General Purpose Input Output
HDMI	High Definition Multimedia Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Secure Hyper Text Transfer Protocol
IoT	Internet of Things, esineiden Internet
JSON	JavaScript Object Notation, tiedon välityksen tiedostomuoto
NAT	Network Address Translation
NoSQL	Not only SQL
NTP	Network Time Protocol, aikatiedon välitysprotokolla
REST	REpresentational State Transfer
SQL	Structured Query Language
SSH	Secure Shell
SVG	Scalable Vector Graphics
TSV	Tab Separated Values, tiedostomuoto
USB	Universal Serial Bus
UTC	Coordinated Universal Time, koordinoitu yleisaika
VNC	Virtual Network Computing

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Jaakko Yli-Luukko
Diplomityön nimi: IoT-sovelluksen toteutus, käyttöasteen seuranta
Valvoja: Jarmo Alander
Ohjaaja: Petri Välisuo
Tutkinto: Diplomi-insinööri
Koulutusohjelma: Sähkö– ja energiatekniikan koulutusohjelma
Suunta: Automaatiotekniikka
Opintojen aloitusvuosi: 2010
Diplomityön valmistumisvuosi: 2017 **Sivumäärä:** 99

TIIVISTELMÄ: Tuotantoa tehostettaessa on tärkeää mitata laitteiden todellinen käyttöaste. Kun tiedetään, miten koneita käytetään ja mihin aika kuluu, voidaan myös löytää parannuskohteita. Kun saadaan parannettua koneiden käyttöastetta, voidaan lisätä tuotantoa ilman uusien koneiden hankkimista. Pelkkä laitteiden mittaus ei riitä, vaan tieto on hyvä visualisoida helposti ymmärrettävään muotoon.

Työssä kävin läpi erilaisia kaupallisia ja vapaan lähdekoodin IoT-järjestelmiä ja niiden käyttömahdollisuuksia. Vertailin myös eri toimijoiden pilvipalveluja IoT-järjestelmän toteuttamisnäkökulmasta ja palvelimen sijoitusalustana. IoT-järjestelmä voidaankin toteuttaa erilaisilla alustoilla ja tekniikoilla. Laitteiden ja tekniikoiden kehittyessä on tärkeää, että järjestelmä on helposti päivitettävissä.

Toteutettu järjestelmä seuraa laitteiden käyttöä ja esittää visuaalisesti koneiden käyttöasteen. Toteutin järjestelmän modulaarisesti. Järjestelmä on jaettu osiin, jolloin järjestelmän osia voidaan kehittää ja muuttaa toisistaan riippumatta. Rajapinta eri osien välillä onkin oltava mahdollisimman pysyvä ja hyvin tuettu.

Laitteiden tilatiedon lukemiseen käytin Raspberry Pi –korttitietokoneita, jotka lähettävät tiedon palvelimelle. Tiedon visualisoinnin toteutin selainpohjaisesti, jolloin käyttö onnistuu suoraan lähes kaikilla laitteilla, joissa on Internet-selain. Työnjohdon lisäksi visualisoitu reaaliaikainen tieto esitetään myös työntekijöille taukotilassa sijaitsevassa televisiossa.

Toteutettu järjestelmä on asennettu konepajaan, jossa järjestelmällä seurataan hitsauskoneita, särmäyspuristimia, laserleikkuria, levyntyöstökeskusta ja robottisolua.

AVAINSANAT: esineiden Internet, teollinen Internet, datan visualisointi, käyttöaste.

UNIVERSITY OF VAASA
Faculty of technology
Author: Jaakko Yli-Luukko
Topic of the Thesis: Implementation of the IoT application, monitoring capacity utilization
Supervisor: Jarmo Alander
Instructor: Petri Välisuo
Degree: Master of Science in Technology
Degree Programme: Degree Programme in Electrical and Energy Engineering
Major: Automation
Year of Entering the University: 2010
Year of Completing the Thesis: 2017 **Pages:** 99

ABSTRACT: When starting to increse the performance of production, it is important to measure the actual capacity utilization of the equipment. Improvements can be found after detecting when and how the machines are being used. When machines are better used, production of the machines can be increased without the acquisition of new machinery. Measuring the capacity utilization is not enough. The information is still to be visualized in an easily understandable form.

Within this work we present different commercial and open source IoT systems and possibilities to use them. Different cloud services were also compared the perspective of IoT-platform implementation and server placement. IoT-systems can be implemented on different kind of platforms and techniques. While equipment and technologies are developing, it is important to have an easily maintained system

The implemented system is following the utilization of the equipment and visulaizes the utilizations of them. The system has modular structure and it consists of clearly separated parts that can be developed and updated independently. The Interfaces between the parts must be stable and supported.

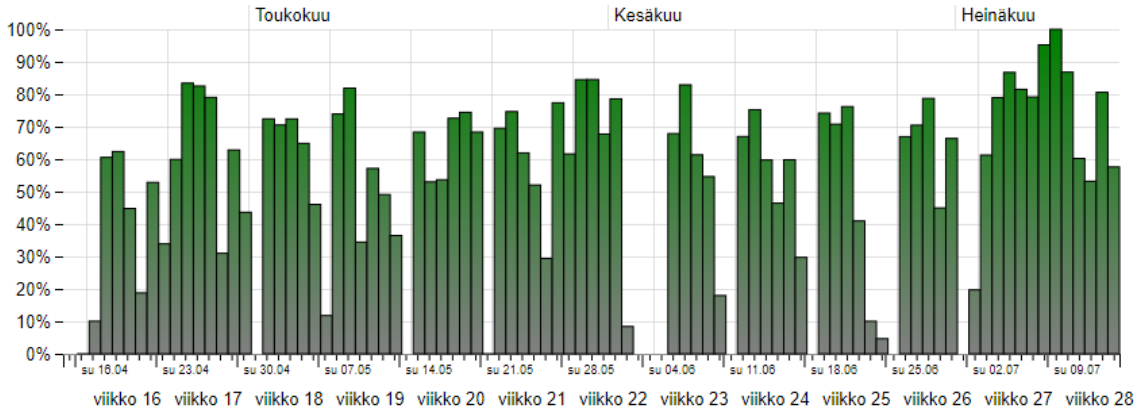
Raspberry Pi single card computer is used for reading the status of the equipment, which is then send to a server. Data visualization is implemented on a browser based user interface, so it can be used directly on almost any device with Internet browser. Besides of work supervision visualized real time data is represented to employees on a television located at break room.

The implemented system is installed in a workshop. The system is used to monitor welding machines, press brakes, laser cutter, turret punch press and robot manufacturing cell.

KEYWORDS: Internet of Things, industrial Internet, data visualization.

1. JOHDANTO

Tässä työssä käsitellään tuotannon koneiden käyttöasteen seurantaan tarkoitettua järjestelmää. Järjestelmän toimittaa RiimaPlan Oy ja se toimitetaan konepaja-alan asiakkaalle. Järjestelmä toteutetaan kuitenkin yleiskäyttöiseksi, jolla voidaan valvoa monen muunkin tehdassalin tai konepajan koneiden käyttöastetta. Järjestelmästä halutaan apua tuotannon seuraamiseen ja tehostamismahdollisuuksien etsimiseen. Kuvassa 1 on erään työstökoneen päivittäisen käyttöasteen kehitys. Kuva on työssä toteutetun järjestelmän eräs visualisointinäkymä. Kesäkuun alussa näkyvä pienempi käyttöaste johtuu siitä, että koneen käyttöastetta oli saatu tehostettua ja koneelta loppuivat tehtävät työt. Heinäkuulle oli taas myyty enemmän töitä.



Kuva 1: Erään työstökoneen käyttöasteen kehitys järjestelmän käyttöönoton jälkeen vuonna 2017.

Työssä on tutustuttu erilaisiin IoT-järjestelmiin ja kartoitettu niiden nykytilanne, sekä valittu työssä käytettävät laitteet, tekniikat ja ohjelmat.

1.1. Konepaja

Tärkeimmät koneet konepajassa on ryhmitelty konetyyppien mukaan. Hitsauskoneet (5 kappaletta) ovat kahdessa eri solussa, mutta niitä voidaan siirrellä tarpeen vaatiessa. Särmäyspuristimet (3 kappaletta) ovat lähellä toisiaan. Laser-leikkuri ja levyntyöstökeskus ovat vähän erillään toisistaan. Robottisolu koostuu robottikäsivarresta, särmäyspuristimesta ja robottikäsivarrella varustetusta hitsauskoneesta.

1.2. Tavoitteet

Asiakasyritys halusi järjestelmän, jolla voidaan seurata koneiden käyttöaikaa ja käyttöaikasuhdetta. Mittaustietoja käydään läpi työntekijöiden kanssa kuukausipalaverissa ja mitattua tietoa voidaan myöhemmin käyttää myös tulospalkkauksen tukena. Asiakasyrityksen tavoitteena on näillä toimilla tehostaa tuotantoa.

Nykytilanteessa koneiden käyttöasteesta on vain työntekijöiden ja työnjohdon oma arvio työkoneen käyttöasteesta. Kun halutaan tehostaa toimintaa, on hyvä olla mitattua tietoa. Toisaalta halutaan myös motivoida työntekijöitä, esittämällä koneiden käyttöaste. Eri tyyppisessä työssä samankin koneen käyttöaste voi olla erilainen, mutta ehkä tärkeämpää onkin verrata samantyyppistä työtä keskenään. Toisaalta tiettyyn työhön käytetty aika ja koneen käyttöaste kertovat työnjohdolle tehtävän työn kustannuksista.

Uutta järjestelmää suunniteltaessa ennakko-olettamuksena on pitää järjestelmä mahdollisimman näkymättömänä työntekijälle, jolloin työntekijän ei tarvitse vaivautua kertomaan järjestelmälle mitä on tekemässä. Ilman erikseen kerrottavaa syytäkin, voidaan nähdä kuinka paljon kone tekee tuottavaa työtä ja voidaan hakea parannuskohteita. Kun järjestelmälle on kerrottu syy, miksi jotain on tapahtunut, on jälkikäteen toki helpompi selvittää mitä on tapahtunut. Onko työntekijöillä sitten tarve selittää tekemisiään, jää nähtäväksi.

Järjestelmän tilaaja ei halua järjestelmään mitään syykoodeja. Lisäksi järjestelmään on

tilattu vain koneen käyttöasteen seuranta ilman häiriöiden tunnistusta. Järjestelmän suunnittelussa kuitenkin pyritään varautumaan siihen, että tulevaisuudessa on mahdollista lisätä erilaisia häiriön tunnistuksia, mittauksia ja syykoodeja.

1.3. Olemassaolevat järjestelmät

Koneiden ja laitteiden seuranta ei ole uusi ajatus. Markkinoilla on useita vastaavia järjestelmiä. Seuraavassa on listattuna muutamia kotimaisia järjestelmiä:

- Control Express Finland Oy, Webrosensor.
- ARROW Engineering Oy, Machine Track.
- Novotek Oy, OEE/KNL-tehokkuudenseuranta.
- Vossi Group Oy, Kone- ja OEE-seuranta.
- TreLab Ltd, Smart Data Mill

Kävimme 8.12.2016 tutustumiskäynnillä toisessa konepaja-alan yrityksessä, jossa on Arrow Engineering Oy:n toimittama Machine Track tiedonkeruujärjestelmä käytössä. Kun järjestelmää oli otettu käyttöön, niin jopa osa työnjohtoa vastusti uutta järjestelmää. Järjestelmä on kuitenkin osoittautunut hyödylliseksi. Esimerkkinä havaittiin kuinka paljon koneen käyttäjiltä kului aikaa materiaalien hakemiseen. Ratkaisuna koneen käyttäjille annettiin käyttöön radiopuhelimet, jolloin he voivat pyytää tarvittaessa varastomiestä tuomaan tarvittavat materiaalit paikalle. Machine Trac mahdollistaa työaikojen suunnittelun, mutta se ei kuitenkaan ole enää käytössä, koska suunnittelemattomat muutokset työajoissa aiheuttivat ongelmia koneen seurannassa. Sen sijaan työntekijöitä on ohjeistettu sammuttamaan virrat koneista työajan päätyttyä, jolloin järjestelmästä näkee, koska koneella on ollut työntekijä. Lisäksi työntekijät merkitsevät järjestelmän päätteelle syykoodin sille ajalle, kun kone ei ole käytössä. He pitivätkin automaattisten syykoodien esteenä anturoinnin vaikeutta, eli kuinka saada selville keskeytyksen syy, esim. materiaalin loppuminen ja toisaalta koneelle tehtävien säätöjen asetteluun kuluva aika.

2. TEKNOLOGIAVERTAILU

Tässä luvussa käyn läpi erilaisia tekniikoita työn toteuttamisen näkökohdista. Työn toteuttamiseen tarvitaan jokin laite, jolla luetaan koneiden käyttötietoa. Käyttötieto on tallennettava jonnekin ja se on pystyttävä esittämään visuaalisesti.

Työn alussa kävin läpi valmiita IoT-järjestelmiä ja niiden käyttömahdollisuuksia.

2.1. Kaupalliset IoT-ratkaisut

Valmiita kaupallisia IoT-ratkaisuja löytyy useammaltakin toimijalta. Kaikki tässä käsitellyt ratkaisut ovat pilvipalveluja. Palveluun sisältyy tietokanta tai tietovarasto, johon tietoa voidaan tallentaa laitteista ja raportointityökaluja. Kaikilla on myös omat ohjelmointirajapinnat laitteille, tiedon käsittelyyn ja raportointiin. Vaikkakin osalla on tarjolla palveluntarjoajan omia laitteita, palveluihin voidaan ydistää myös muita vapaasti ohjelmoitavia laitteita.

Palveluita voi jossain määrin kokeilla ilmaiseksi, mutta rekisteröiminen vaatii lähes aina luottokorttitietojen antamisen. Uskoisin, että suunnitellun järjestelmän rakentaminen onnistuu kaikkien toimijoiden työkaluilla. Nähdäkseni suurin etu kaupallisten IoT-ratkaisujen käytölle on valmiit rajapinnat ja jonkin tasoinen laitteiden hallinta. Lisäksi palvelimen hallinta on ulkoistettu, mikä voi olla sekä hyvä että huono asia.

2.1.1. Amazon AWS IoT

Amazon AWS IoT palvelut jakautuvat kolmeen osaan (Amazon, 2017):

- AWS Greengrass on ohjelmisto, joka laajentaa AWS pilvipalvelun ominaisuuksia paikallisille laitteille.
  AWS Greengrass Core ohjelmiston ominaisuuksia.

  - Sallii funktioiden jakamisen pilvipalvelusta ja suorittamisen paikallisessa IoT-laitteessa. Tuettuja ohjelmointikieliä ovat Python, Node.JS ja Java.
  - Mahdollistaa paikallisen viestinvälityksen IoT-laitteiden välillä.
  - Varmistaa turvallisen yhteyden laitteiden ja pilvipalvelun välillä.
  - Tarjoaa laitteiden pilvipohjaisen hallinnan laiterekisterin kautta ja laitteiden etäpäivityksen.

  Laitteiden tulee olla vähintään Raspberry Pi –tasoisia, varustettuna Linux-käyttöjärjestelmällä.
- AWS IoT on hallittu pilvipalvelualusta, joka mahdollistaa palveluun kytkettyjen laitteiden vuorovaikutuksen pilvipalvelusovellusten ja toisten laitteiden kanssa.
- AWS IoT Button on yksinkertainen Wi-Fi laite, jossa on yksi painike. Painikkeen toiminto ohjelmoidaan pilvipalvelussa ja sitä voidaan käyttää kaukosäätimenä käynnistämään ja sammuttamaan sovelluksia ja laitteita.

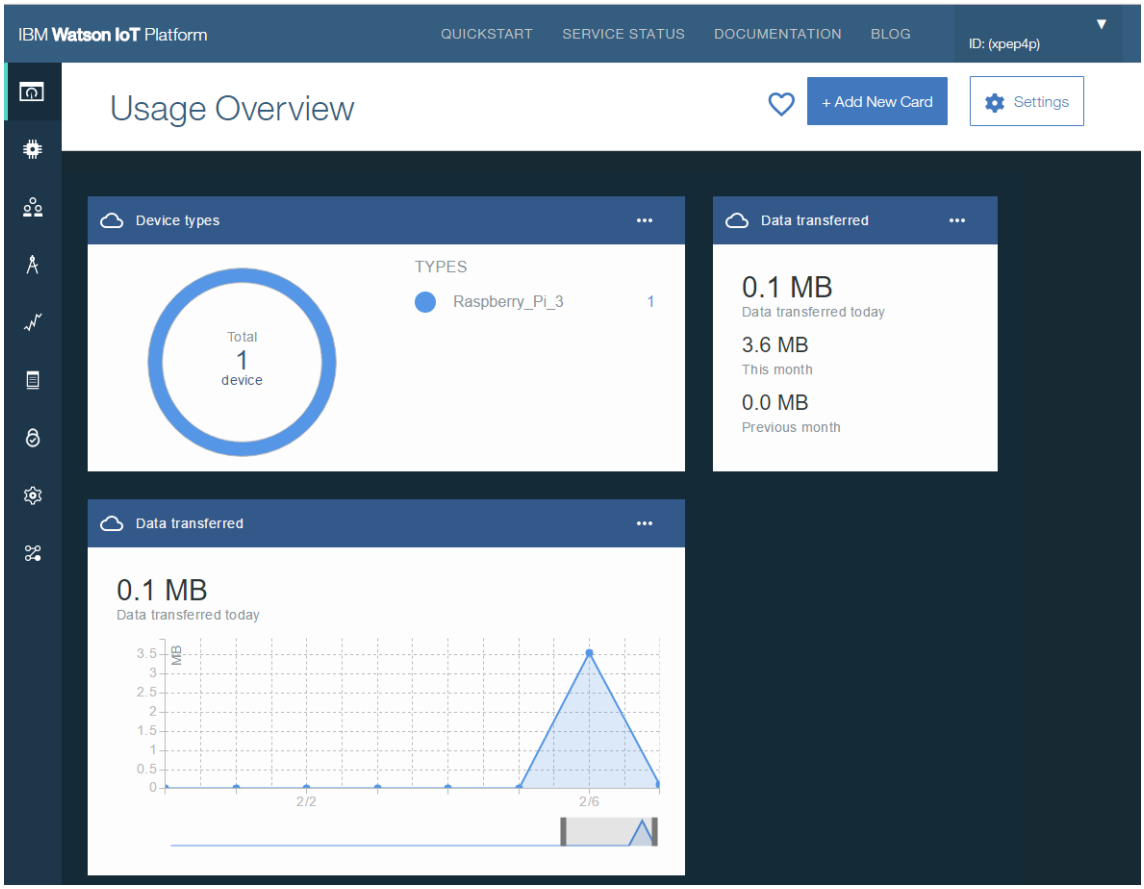
2.1.2. Google Cloud IoT

Google Cloud IoT on kehittynyt tämän työn aikana beta-vaiheessa olevaksi pilvipohjaiseksi IoT-palveluksi. IoT Core on osa Google Cloud Platform pilvipalvelua (Google, 2017c). IoT Core palveluun luodaan yksi tai useampi laiterekisteri, johon itse laitteet lisätään. Viestinvälitykseen käytetään Pub/Sub palvelua, joka on myös osa Google Cloud Platform pilvipalvelua. Muu toiminnallisuus on mahdollista toteuttaa Google Cloud Platform palveluiden avulla.

2.1.3. IBM Watson IoT Platform

IBM Watson IoT Platform on yksi osa IBM Bluemix pilvipalvelua (IBM, 2017b). Visualisointia varten on erilaisia tauluja, johon voidaan liittää kortteja, joihin taas saadaan yksinkertaisia kuvaajia ja mittareita, tai tietoja tekstimuodossa (kuva 2).

Laitteita varten luodaan ensin laitetyyppi, johon voidaan laitekohtaisesti liittää eri tietoja laitteesta. IBM on julkaissut asiakaskirjastot usealle eri ohjelmointikielelle. Asiakaskir-



Kuva 2: IBM Watson IoT Platform.

jaston avulla sovellus tai laite voi kommunikoida Watson IoT Platform –alustan kanssa. Esimerkiksi Python kirjasto asentui suoraan Raspberry Pi:n Raspbian Linuxille.

2.1.4. Wapice IoT-Ticket

En ole päässyt kokeilemaan IoT-Ticket –alustaa, mutta esittelyvideon perusteella se vaikuttaa olevan markkinoiden kehittynein IoT-alusta, mihin olen törmännyt. Erityisesti tiedon visualisointi on todella vaikuttavaa (Wapice, 2017). Alustaan voidaan liittää myös muita laitteita tai sovelluksia IoT-Ticket API rajapinnan kautta, jopa Android matkapuhelinta varten on oma sovelluksensa.

Työn loppuvaiheessa hintatietoja ei enää löytynyt yrityksen Internet-sivuilta, mutta aiem-

min (4.2.2017) oli saatavilla 30 päivän ilmainen kokeiluversio, sekä kolme eri tasoista ratkaisua 349 €/kk aina 4799 €/kk asti. WRM 247+ IoT-laitteen hinta oli 645 €.

2.1.5. Microsoft Azure IoT Hub

Azure IoT Hub on suojattu viestinvälityspalvelu, jolla liitetään IoT-laitteet Azure pilvipalveluun (Microsoft, 2017a). Ohjelmointirajapinta IoT-laitteille on saatavilla .NET, Java, Node.JS, Python ja C –ohjelmointikielille. Tuettuna on useita eri laitteita mm. Raspberry Pi. IoT-laitteiden lähettämää tietoa voidaan visualisoida käyttämällä Power BI ja Web Apps –palveluja.

IoT Hub hinnoittelu on esitetty taulukossa 2.

Taulukko 2: Azure IoT Hub hinnoittelu (24.11.2017).

Laitteiden määrä	**Viestiä päivässä**	**Hinta [€/kk]**
500	8 000	ilmainen
rajoittamaton	400 000	42,17
rajoittamaton	6 000 000	421,65
rajoittamaton	300 000 000	4 216,50

2.1.6. Windows 10 IoT Core

Windows 10 IoT Core on IoT-laitteille tarkoitettu Windows käyttöjärjestelmä (Microsoft, 2017b). Tuettuja laitteita on useita mm. Raspberry Pi. Windows 10 IoT Core käyttöjärjestelmän asennus muistikortille tapahtuu *Windows 10 IoT Core Desktop* –ohjelmalla, joka on saatavilla Windows 10 käyttöjärjestelmälle. Kokeilin Windows 10 IoT Core –käyttöjärjestelmää Raspberry Pi 3:lla. Asentaminen onnistuu helposti micro sd kortille. Asennuksen automaattinen viimeistely Raspberry Pi 3 kortilla kestää hetken. Langaton-

ta verkkoa käytettäessä, Wi-Fi asetukset täytyy asettaa asennuksen jälkeen. Tätä varten täytyy Raspberry Pi:hin kytkeä näyttö ja näppäimistö. Asennetun Windows 10 IoT Core käyttöjärjestelmän käyttö paikallisesti on hyvin rajoitettua. Käytössä on yksinkertainen komentotulkki ja Internet-selain. Windows 10 IoT Core käyttöjärjestelmän etäkäyttö tapahtuu ottamalla yhteys selaimella laitteen porttiin 8080. Selaimella voidaan muuttaa laajemmin asetuksia ja asentaa sovelluksia. Esimerkkiohjelmien asennus sujui helposti ja ne myös toimivat hyvin. Käyttöjärjestelmä käynnistyy vähän hitaammin, kuin Raspbian Linux.

2.1.7. General Electric Predix

General Electric Predix on teollisen Internetin alusta, joka on rakennettu alunperin GE:n omia liiketoiminta-alueita varten. Tarjolla on GE:n omia sovelluksia, kolmannen osapuolen tekemiä sovelluksia ja Predix Developer Network alustapalvelu (GE, 2017).

Predix Developer Network on laaja alustapalvelu sovellusten rakentamiseen. Järjestelmässä on laaja valikoima valmiita palveluita, ohjelmistoja, analytiikkaa ja sovelluksia, joita voidaan hyödyntää sovelluksissa.

2.2. Open-source IoT-ratkaisut

2.2.1. Kaa IoT Platform

Kaa on open source middleware IoT –alusta, jonka taustalla on CyberVision –niminen ohjelmistoyritys (KaaIoT, 2017). Ohjelmisto on ilmainen ja vapaasti ladattavissa. Valittavana on ladattava valmis virtuaalikone ”hiekkalaatikko”, Amazonin pilvipalvelussa toimiva ”hiekkalaatikko”, Linux-asennuspaketit ja lähdekoodipaketti. Kokeiltaessa virtuaalikone toimii ongelmitta. Hallinta tapahtuu selainpohjaisesti. Mukana on useita demosovelluksia, joista yksi on Raspberry Pi:lle, lämpötilanvalvontasovellus. Osa ohjeista ja demo-sovelluksista on vielä vanhemmille versioille tehtyjä. Kehitystyö onkin vielä selvästi

kesken. Palvelimen ohjelmointirajapinnat näyttävät melko valmiilta. SQL-tietokannoista on tuettuna MariaDB ja PostgerSQL, NoSQL-tietokannoista on tuettuna Apache Cassandra ja MongoDB. Datan visualisointia ei ole.

Kaa vaikuttaa toimivan, mutta on kuitenkin vielä sen verran keskeneräinen, että se ei taida olla vielä tuotantokäyttöön soveltuva, koska aiemmistakin versioista on epäyhteensopivia muutoksia.

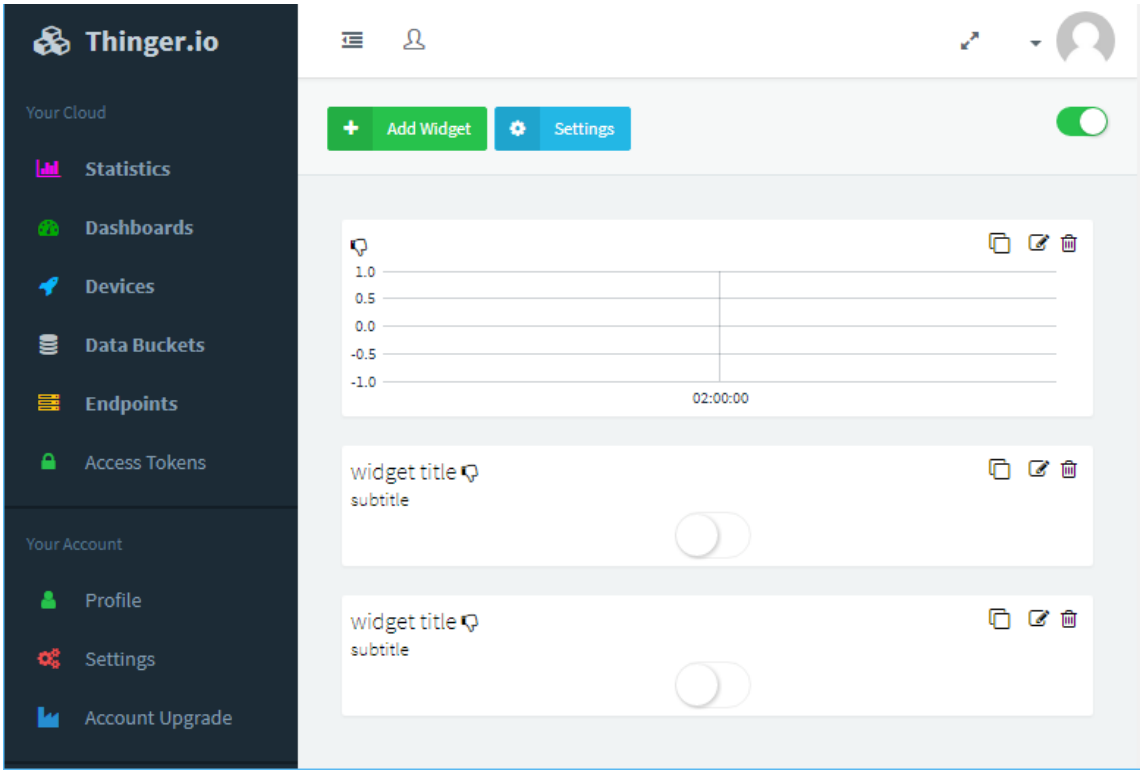
2.2.2. Thinger.io platform

Thinger.io platform on open source IoT-alusta (Thinger.io, 2017). Palvelinohjelmisto on ladattavissa Ubuntu-Linuxille. Amazon AWS pilvipalveluun on saatavilla valmis pilvipalvelinpohja. Myös Raspberry Pi 3:lle on saatavilla Thinger.io palvelin levykuvana. Thinger.io tarjoaa IoT alustaa pieneen käyttöön ilmaisena pilvipalveluna ja laajempaan käyttöön maksullisena palveluna. Palvelussa on hallintapaneelit laitteille, tietovarastoille, käyttöoikeuksille ja ohjattaville tapahtumille. Ohjattavia tapahtumia ovat mm. sähköpostin lähetys, HTTP-pyyntö ja laitekutsu. Visualisointia varten on yksinkertaisia kuvaajia (kuva 3). Palvelimen ohjelmointirajapintojen dokumentaatiossa oli vielä puutteita.

IoT-laitteiden ohjelmointirajapinta löytyy Arduinolle ja Linux-pohjaisille laitteille, myös Max OS:lle.

2.2.3. The thing system

The thing system on open source ohjelmisto, joka on keskittynyt laitteiden väliseen kommunikointiin. Tuettuna onkin useita eri tyyppisiä laitteita. Projektin ytimenä on *steward* –niminen Node.JS ohjelma (Blixa Morgan, 2017). Ohjelman asennus ei kuitenkaan toiminut ja projekti näyttääkin jääneen kesken, koska viimeisin päivitys ohjelmistoon on tehty kesällä 2016.



Kuva 3: Thinger.io IoT pilvipalvelussa on visualisointia varten yksinkertaisia kuvaajia.

2.2.4. ThingSpeak

ThingSpeak on avoin ilmainen open source palvelu ei-kaupalliseen käyttöön, jossa palveluun ladattua dataa voidaan analysoida ja visualisoida MatLab-ympäristössä ilman MatLab-lisenssiä. Käyttö on melko rajattua, tiedon tallennustapaan ei juurikaan pysty vaikuttamaan. Hyvänä puolena on MatLabin käyttö datan analysointiin (Mathworks, 2017).

Kaupalliseen käyttöön on lisenssi, joka maksaa 650 €(9.11.2017). Yksi lisenssiyksikkö sisältää 33 miljoonaa viestiä, joka riittää yhdelle laitteelle vuodeksi, kun lähetetään yksi viesti sekunnissa. Dataa voidaan tallentaa 100 miljoonaa viestiä yhtä lisenssiyksikköä kohden.

ThingSpeak voidaan asentaa myös paikallisesti, mutta silloin ei ole käytössä MatLab – analysointityökaluja. ThingSpeak on paikallisesti asennettuna turhan suppea sellaisenaan.

ThingSpeak-palvelu sopii hyvin demokäyttöön, prototyyppeihin ja pieneen IoT-laitteiden seuraamiseen. Paikallisesti asennettunakin voidaan nopeasti saada yksinkertaisia kuvaajia mittaustiedoista.

ThingSpeak tukee yksinkertaisia kuvaajia, mutta data voidaan myös hakea halutulta aikaväliltä json-muodossa, jolloin voidaan käyttää myös muita analysointityökaluja.

Raspberry Pi demon asennus on liitteessä LIITE 4.

Tein ohjelman, joka lukee painiketiedon Raspberry Pi:llä ja lähettää tiedon palvelimelle. Lähdekoodi on liitteessä LIITE 6.

2.2.5. Matlab

Matlabissa on suora tuki Raspberry Pi:lle laitetasolla ja ThingSpeak-palvelulle. Lisäksi tietoa voidaan hakea eri pilvipalveluista ja tietokannoista. Matlab on varteenotettava vaihtoehto rakennettaessa IoT-järjestelmiä, varsinkin kun tarvitaan monimutkaisempaa datan analysointia, tai Matlab on muuten käytössä.

2.3. Yhteenveto olemassaolevista IoT-ratkaisuista

IoT-ratkaisut ovat tällä hetkellä voimakkaan kehityksen alla. Osa IoT-projektien ja -valmistajien Internet-sivuista oli kadonnut tämän työn tekemisen aikana. Kokeilluista IoT-ratkaisuista vain ThingSpeak oli valmis käyttöön ilman lisäohjelmointia. ThingSpeak sopii erityisen hyvin nopeaan kehitys- ja demokäyttöön. IBM:n palvelu oli kokonaisuudessaan monipuolinen, mutta laajuudestaan johtuen tarvitsee myös paljon perehtymistä. Amazonin palvelusta täytyy nostaa erityisesti esille mahdollisuus päivittää IoT-laitteen ohjelmisto etänä.

IoT-ratkaisujen suora vertailu on hankalaa, koska kaikki ovat erilaisia niin laajuudessaan

kuin toteutustavassaan. Myös samankaltaisia ominaisuuksia löytyy. Google IoT Core ja Azure IoT Hub ovat periaatteeltaan hyvin samankaltaisia. Google ja Azure vaikuttavat hyvin suppeilta verrattuna IBM:n IoT-ratkaisuun. Kuitenkin Googlen ja Azuren pilvipalvelussa on laajat mahdollisuudet tiedon jatkojalostukselle. Ero IoT-ratkaisun ja muun pilvipalvelusovelluksen välillä on häilyvä, jos sitä on tarpeen erottaakaan.

Sovelluksesta riippuen valmiin IoT-alustan käyttö ei välttämättä vähennä tarvittavan työn määrää, koska IoT-alustan opiskelu vie aikaa. Ratkaisevaa on kuitenkin se, mitä IoT-sovelluksen on tarkoitus tehdä. Mikäli on olemassa valmis sovellus jonkin toimijan palvelussa, voi olla tarkoituksenmukaista käyttää myös kyseisen toimijan IoT-ratkaisua. Jos sovellus kuitenkin rakennetaan puhtaalta pöydältä, voi olla helpompi rakentaa myös IoT-alusta itse. Rajoitetussa ympäristössä IoT-laitteiden rajapinnasta saadaan melko yksinkertainen. Yleiskäyttöisen IoT-ratkaisun tekeminen onkin sitten jo haastavampaa.

Kokeilemistani IoT-ratkaisuista vain osassa oli kiinnitetty huomiota tiedon visualisointiin ja niissäkin visualisointi rajoittui muutamiin eri peruskuvaajiin. Monimutkaisempi visualisointi ei valmiilla IoT-ratkaisuilla onnistu suoraan. Monimutkaisempaa tietojen käsittelyä varten kaikissa kokeilemissani alustoissa on kuitenkin jokin rajapinta olemassa.

Se mitä haluaisin IoT-alustoissa nähdä tulevaisuudessa on: Yksi tai kaksi standardia rajapintaa laitteiden ja palvelimen tai tietovaraston välillä. Pienelle datamäärälle REST API (Fredrich, 2016) ja kaksisuuntaiseen tiedonsiirtoon ja suurille datamäärille gRPC (gRPC, 2016) tai jokin sen kaltainen. Lisäksi tarvitaan laitteiden hallinta, tiedon analysointi- ja visualisointimahdollisuus ja tarvittaessa laitteiden ohjauslogiikka ja käyttöliittymä. Tällä hetkellä lähes kaikki taitaa olla valmiina ThingSpeak, Matlab, Simulink –yhdistelmällä.

Usealla laitevalmistajalla on myös omia IoT-alustoja, jotka on tarkoitettu toimivaksi vain valmistajien omien laitteiden kanssa. Mielestäni pidemmän aikavälin ratkaisu tulisi kuitenkin olla avoin, tai ainakin osittain avoin laiteriippumaton alusta. Valmistajia ja laitteita tulee ja menee, mutta toimivat rajapinnat pysyvät kauemmin.

Monet pilvipalvelut, esimerkiksi Google Cloud Platform ja IBM tuntuvat painottavan sovellusten välisiä rajapintoja, mikä mahdollistaa järjestelmän osien riippumattomuuden toisista osista, sekä sovellusten liittämisen toisiin sovelluksiin ja palveluihin. Jopa kilpailevan toimijan palveluihin on omat rajapintansa. Kun rajapinnat ovat sovittuja, voidaan eri osia kehittää erikseen ja vaihtaa tarvittaessa toiseen. Tämä on nähty muuallakin standardoinnin tarpeena ja kehittymisenä, esimerkiksi eri valmistajien Wi-Fi laitteet kommunikoivat toistensa kanssa. Uskonkin, että tulevaisuudessa nimenomaan rajapintojen standardointi tulee helpottamaan ja lisäämään IoT-laitteiden käyttöä.

Pienelle tarpeelle ja pienille yrityksille kaupallisten IoT-ratkaisujen hinta voi tulla käyttöönoton esteeksi. Toisaalta nimenomaan pienessä yrityksessä ei välttämättä ole aikaa eikä tietotaitoa oman IoT-ratkaisun toteuttamiseen.

2.4. Räätälöity IoT-ratkaisu

Useimmat open source –alustat keskittyvät lähinnä kommunikointiin laitteiden välillä, eivätkä datan analysointiin ja visualisointiin. Kaupallisten IoT-ratkaisujen ja valmiiden open source –ratkaisujen sijaan järjestelmä voidaan rakentaa myös itse. Ratkaisuun voidaan yhdistää osia sekä kaupallisista että open-source –ratkaisuista.

2.5. Pilvipalveluiden vertailu

Järjestelmää suunniteltaessa, päädyttiin palvelinten osalta mahdolliseen pilvipalvelujen käyttöön. Kansainvälisiä pilvipalveluita tarjoaa ainakin Google, Microsoft ja Amazon. Lisäksi on useita pienempiä koti- ja ulkomaisia palveluntarjoajia. Valinta pilvipalveluntarjoajan välillä päätettiin tehdä Googlen ja Microsoftin välillä, joihin oli tarkoitus tutustua tarkemmin.

Pilvipalvelinten osalta perehdyin siis kahteen eri vaihtoehtoon, Google Cloud Platform ja

Microsoft Azure, joiden välillä valinta tultaisiin tekemään. Molemmissa voidaan luoda sekä Linux-, että Windows-virtuaalipalvelimia, molemmissa on valmiita työkaluja ja pohjia erilaisille palveluille. Lisäksi tutustuin päällisin puolin IBM:n pilvipalveluihin.

2.5.1. Google Cloud Platform

Google Cloud Platform (Google, 2017b) on kokeiltavissa ilmaiseksi 60 päivän ajan. Lisäksi rekisteröityessä täytyy antaa luottokortin tiedot.

Virtuaalipalvelinta luodessa valitaan sopivin datakeskus, jossa palvelu sijaitsee. Kirjoitushetkellä lähin on St. Ghislain, Belgiassa. Vuonna 2017-2018 pitäisi olla saatavilla myös Haminan datakeskus Suomessa. Virtuaalipalvelimelle valitaan konetyyppi, prosessoriydinten lukumäärä ja muistin määrä. Kirjoitushetkellä (17.11.2017) valittavana olevat käyttöjärjestelmät ovat taulukossa 3.

Taulukko 3: Google Cloud Platform palvelinten käyttöjärjestelmät.

Debian GNU/Linux
CentOS
CoreOS
Ubuntu
ChromiumOS-pohjainen Container-Optimized OS
Red Hat Enterprise Linux
SUSE Linux Enterprise Server
Windows Server

Valittavissa on myös suuri määrä valmiiksi asennettuja kolmannen osapuolen sovelluspalvelimia. Voidaan myös valita oma virtuaalikone tai kopioida jokin käytössä oleva virtuaalikone. Virtuaalikoneen hinta muodostuu valittujen ominaisuuksien mukaan. Virtuaalikoneen luonti kesti pari minuuttia. Palvelussa virtuaalipalvelimia voidaan luoda uusia,

kopioida, poistaa, sammuttaa, käynnistää ja muokata, palvelimia voidaan myös ryhmitellä. Virtuaalipalvelinten kuormitusta voidaan seurata ja palvelimeen voidaan ottaa ssh-yhteys suoraan selaimella, sekä ulkoisella ohjelmalla. Pilvipalvelun hallintaan on myös Android-sovellus, jolla palveluja voidaan hallita suoraan puhelimella. Jokaiselle virtuaalikoneelle on oma sisäinen kiinteä IP-osoite, sekä mahdollinen ulkoinen IP-osoite, joka voi olla kiinteä tai vaihtuva. Virtuaalikoneen luontiin ja hallintaan on myös REST API, kuten monissa muissakin palveluissa. Pienin muistimäärä jaetulla prosessoriytimellä on 0,6 GB. Pienin levytila on 10 GB. Enimmillään virtuaalikoneessa voi olla 64 ydintä ja 416 GB muistia. Maksimi levytila on 65536 GB. Virtuaalikoneen hintaesimerkkejä on taulukossa 6.

Palvelussa voidaan lisätä käyttäjiä Google-tilin mukaan ja hallita käyttäjien oikeuksia eri palvelujen osiin. Taulukko Google Cloud Platform palveluista on liitteessä LIITE 5. Taulukossa 4 on Googlen eri tallennusvaihtoehtojen vertailu (Google, 2017a).

Taulukko 4: Google Cloud Platform eri tallennusvaihtoehtoja (Google, 2017a).

Persistent Disk	Täysin hallinnoitu levytallennustila.
Google Cloud Storage	Skaalautuva hallittu tallennustila mm. multimediatiedostoille ja muulle jäsentämättömälle tiedolle.
Google Cloud Bigtable	Skaalautuva hallittu NoSQL tietokanta, joka soveltuu sekä reaaliaikaiseen käsittelyyn, että analysointiin.
Google Cloud Datastore	Skaalautuva hallittu NoSQL tietokanta web- ja mobiilisovellusten tiedontallennukseen.
Google Cloud SQL	Hallittu MySQL ja PostgreSQL tietokantapalvelu.
Google Cloud Spanner	Globaalisti saalautuva korkean saatavuuden relaatiotietokantapalvelu.
Google BigQuery	Skaalautuva hallittu tietovarasto, josta voidaan tehdä SQL-hakuja. Soveltuu suurten tietomäärien käsittelyyn.
Google Drive	Yhteisöllinen tila tiedostojen tallennukseen, jakamiseen ja muokkaukseen. Soveltuu vuorovaikutukseen loppukäyttäjän kanssa.

Taulukossa 5 on esimerkkinä *Billing*-palvelussa esitetyt tiedot, Google Cloud Platform palvelujen kustannusten kertymisestä.

Taulukko 5: *Billing*-palvelussa esitetyt tiedot, Google Cloud Platform palvelujen kustannusten kertymisestä

Product	**Resource**	**Usage**	**Amount**
Cloud Bigtable	Server Node	344.31 Hours	€211.43
Google Compute	Generic Small instance with 1 VCPU, no scratch disk	5,611 Minutes	€2.65
Google Compute	Storage FD Capacity	1.26 GB-month	€0.05

Yhteenvetona Google Cloud Platform hyvät puolet:

+ helppokäyttöinen hallinta selaimen kautta,
+ virtuaalipalvelinten hallinta,
+ käyttöoikeuksien hallinta,
+ virtuaalikoneiden salasanoja ei tarvitse muistaa.

Ja huonot puolet:

- Kustannusten kertymisen arviointi osassa palveluita, esim. BigTable.

2.5.2. Microsoft Azure

Microsoft Azure (Microsoft, 2017c) on kokeiltavissa ilmaiseksi 30 päivän ajan. Lisäksi rekisteröityessä täytyy antaa luottokortin tiedot.

Microsoft Azure:ssa on valittavana paljon erilaisia valmiita virtuaalikoneympäristöjä ja Windows- ja Linux-käyttöjärjestelmiä. Virtuaalikoneen ominaisuudet (mm. ydinten lukumäärä, muisti ja levytila) voidaan valita muutamasta kokonaisuudesta, yksittäisiä ominaisuuksia ei voi valita. Virtuaalikoneen luominen kestää noin 5 minuuttia. Jokaiseen

virtuaalikoneeseen ja palveluun yhdistetään aina jokin DNS-nimi. Virtuaalikoneen hintaesimerkkejä on taulukossa 6. Virtuaalikoneiden lisäksi Azuressa on laaja valikoima erilaisia sovelluspalveluja, joihin perehtymiseen menisi paljon aikaa. IP-osoitteiden, virtuaaliverkkojen, DNS-palvelujen ja muiden verkkojen hallintaan liittyviin asioihin on omat palvelunsa.

NoSQL tietokantarajapinnaksi voidaan valita DocumentDB ja MongoDB. SQL tietokanta on azuren oma. SQL-tietokannan käyttöönotossa valitaan haluttu maksimi koko, jonka pohjalta laskutus tapahtuu. Tietokantaa ja laskutuspohjaa voidaan myös jälkikäteen muuttaa. SQL-tietokanta vaatiikin laajempaa perehtymistä aiheeseen. Azuressa on myös hallittu MySQL-palvelu.

Azure HDInsight Cluster on pilvipohjainen Apache Spark ja Hadoop palvelu. Valittavia palveluja ovat Hadoop, HBase, Storm, Spark, Interactive Hive, R server ja Kafka. Kokeiltaessa palvelujen käyttöönottoa tuli testikäytön 60 prosessoriytimen raja vastaan. Kun Hadoop ja Spark –palveluista oli valittu resursseiltaan pienimmät vaihtoehdot perjantaina ja palvelut olivat vain päällä ilman käyttöä, oli ilmaiseen testikäytöön varattu 170 €:n raja kulunut loppuun sunnuntaina. Syy yllättäviin kustannuksiin on olettavasti siinä, että Hadoop palvelu luo pienimmilläänkin niin paljon resursseja vaativan virtuaalikoneen, että sen käyttökustannukset ovat suuret. Kustannukset esitetään Azuressa visuaalisesti resurssien mukaan jaoteltuna ja ajan mukaan kertymänä. Kertymäkuvaajassa on mukana myös ennuste kustannusten kertymisestä. Lisäksi kustannukset esitetään taulukkona resurssien mukaan.

Yhteenvetona Microsoft Azure hyvät puolet:

+ virtaalikoneille ja palveluille annetaan aina jokin dns-nimi,

huonot puolet:

- virtuaalikoneiden salasanojen hallinta,
- yllättävät kustannukset palvelun käytöstä, esim. HDInsight.

2.5.3. IBM Bluemix

IBM Bluemix (IBM, 2017a) on kokeiltavissa ilmaiseksi 30 päivän ajan. Luottokorttia ei tarvita kokeilujakson aktivoimiseksi. IBM Bluemix pilvipalvelussa on tarjolla palvelimia erilaisina palveluina, fyysisinä palvelimina ja virtuaalipalvelimina. Palvelimien kokeileminen ei kuitenkaan onnistu ilmaisessa kokeilujaksossa.

Palvelimien lisäksi tarjolla on infrastruktuuripalveluita tiedon tallennukseen, verkkopalveluita ja tietoturvaa. Palveluita voidaan toteuttaa myös sovelluspalveluina ilman erillistä palvelinta. Tuettuina on useita eri ohjelmointikieliä. Sovelluspalvelussa laskutus tapahtuu gigatavutunteina, pilvipalvelussa aktiivisena olevan muistin määrä tuntia kohden. Sovelluspalveluita voidaan yhdistää muihin palveluihin, mm. IoT-palveluun. Rajapintojen luomiseen ja hallintaan on oma palvelunsa.

Data & Analytics –palvelut sisältävät useita erilaisia tietokanta- ja analysointipalveluita ja mm. *IBM Watson Machine Learning* –koneoppimispalvelun.

Watson –palvelut sisältävät monia kognitiivisia sovelluksia ja työkalua, mm. tekstistä puheeksi, puheesta tekstiksi, dokumenttien muunnos, kielenkääntö ja visuaalista tunnistusta.

Internet of Things Platform –palvelu on käsitelty kohdassa 2.1.3.. Tarjolla on myös muita erikoissovelluksia, mm. kuljettajan käyttäytymiseen, elektroniikan laitteisiin ja vakuutusyhtiöille valvontaan.

IBM Bluemix –palvelun suurin vahvuus on mielestäni Watson-tekoälypalvelut, mikäli niille on tarvetta.

2.6. Palvelimen valinta

Päätin toteuttaa järjestelmän siten, että se toimii omalla palvelimella, mahdollisesti jopa Windows-palvelimella, sekä Googlen tai Microsoftin tai jonkin muun palveluntarjoajan pilvipalvelussa. Palvelimen paikkaa voidaan siten haluttaessa vaihtaa. Valinnan vapaus edellyttää sellaisten ohjelmistokomponenttien käyttöä, jotka toimivat kaikissa palvelimissa ja palveluissa. Pilvipalveluiden toimittajakohtaisia ohjelmistorajapintoja ei voida silloin käyttää. Tietokantapalvelimeksi valitsin MySQL ja WWW-palvelimeksi Apache. Sekä Googlen että Microsoftin pilvipalvelussa on suora tuki hallinnoidulle MySQL-palvelulle. MySQL voidan myös ottaa käyttöön suoraan virtuaalipalvelimeen.

Google Cloud Platform ja Microsoft Azure olivat minulle etukäteen tuntemattomia, joten pääsin tekemään vertailua puhtaalta pöydältä, ehkä pieni ennakkoasenne Microsoftia vastaan. Google Cloud Platform tuntui heti aluksi melko selkeältä, kun taas Microsoft Azure tuntui sekavalta. Kustannusten osalta yllätys oli Azuren HDInsight, joka käytti kaiken ilmaisen testijakson käyttöön varatusta budjetista. Samalla lailla Googlen BigTable käytti lähes kaiken ilmaisen testijakson käyttöön varatusta budjetista. Azuressa on monipuolisempi kustannusten seuranta. Siinä tuntui olevan joitain ristiriitaisuuksia, mutta ilmeisesti se ei vain ole täysin reaaliaikainen. Kustannuksen kertymän lisäksi siinä on ennustettu kertymä. Googlen kustannusten seuranta on hyvin minimaalinen ja näyttää lähinnä tuotekohtaisesti summat. Googlen palvelussa voi tehdä kustannuksille budjetin ja hälytyksiä.

Hinnaltaan yleiskäyttöinen virtuaalipalvelin on Googlen palvelussa vähän halvempi, kuin Azuressa (taulukko 6). Googlen palvelussa on mielestäni selkeämpi palvelinten hallinta. Azuren palvelussa annetaan palvelimelle aina jokin dns-nimi, kun Googlen palvelussa dns-nimien hallinta on erillinen palvelu. Kehitysvaiheessa on helpompi, kun virtuaalikoneella on DNS-nimi eikä pelkkä IP-osoite.

Pilvipalvelussa maksetaan sekä käytön mukaan että varatusta kapasiteetista. Kapasiteettia on helppo jälkikäteen lisätä. Palvelun aloitus ja lisäys on myös nopeasti tehtävissä pilvi-

Taulukko 6: Google Cloud Platform ja Microsoft Azure virtuaalikoneiden hintaesimerkkejä (24.11.2017).

Google Cloud Platform					
Ytimiä	**Muisti**	**Levytila**	**Levytyyppi**	**Hinta [$/kk]**	**Käyttöjärjestelmä**
1 jaettu	0,6 GB	10 GB	HDD	4,28	Debian, CentOS, CoreOS ja Ubuntu
1 jaettu	0,6 GB	10 GB	HDD	18,88	SUSE
1 jaettu	0,6 GB	50 GB	HDD	20,48	Windows Server 2016
8	52 GB	512 GB	HDD	262,49	Debian
8	52 GB	512 GB	SDD	329,05	Debian
8	52 GB	512 GB	HDD	496,09	Windows Server 2016
64	416 GB	65536 GB	HDD	4 557,52	Debian
64	416 GB	65536 GB	SDD	13 077,20	Debian
64	416 GB	65536 GB	HDD	6 426,32	Windows Server 2016
Microsoft Azure					
Ytimiä	**Muisti**	**Levytila**	**Levytyyppi**	**Hinta [€/kk]**	**Käyttöjärjestelmä**
1	0,75 GB	32 GB		12,41	CentOS, Ubuntu
8	14 GB	512 GB		250,45	CentOS, Ubuntu
8	14 GB	512 GB		388,73	Windows
8	56 GB	512 GB		620,21	Ubuntu
8	56 GB	512 GB		851,70	Windows
64	256 GB	4096 GB		2 192,63	Ubuntu
64	256 GB	4096 GB		4 148,12	Windows

palvelussa. Omassa fyysisessä palvelimessa omissa tiloissa on aluksi suuri alkuinvestointi, jonka jälkeen maksetaan lähinnä sähköstä ja tietoliikenteestä.

Pilvipalvelussa palvelin tulee nimenomaan palveluna ja palvelusta maksetaan. Omassa fyysisessä palvelimessa maksetaan palvelimesta ja ylläpito ja asennustyö tehdään itse, tai maksetaan ylläpidosta ja asennuksesta. Toisaalta myös pilvipalvelu pitää asentaa ja ylläpitää,

pilvipalvelussa asennus ja ylläpito on kuitenkin nopeampaa ja helpompaa. Pilvipalvelua käytettäessä Internet-liikennettä tarvitaan sovelluksesta riippuen enemmän, kuin yrityksen lähiverkossa sijaitsevalla palvelimella. Tässä tapauksessa näkyvin ero omissa tiloissa olevaan palvelimeen verrattuna on käyttöliittymän käytössä, jonka käytön nopeus riippuukin Internet-yhteyden nopeudesta. Uskon kuitenkin, että siinäkään ei ole merkittävää eroa. Tiedonkeruuyksiköiden kannalta palvelimen sijainnilla ei ole merkitystä. Pilvipalvelimen etuna on toimintavarmuus. Toki pilvipalvelussakin paremmasta toimintavarmuudesta laiterikkojen varalta maksetaan erikseen. Toisaalta Internet-yhteyden toimintavarmuus pilvipalvelua käytettäessä tulee kriittisemmäksi. Yrityksen tietoturvapolitiikalla voi myös olla merkitystä palvelimen valinnassa. Tietoturvan kannalta en kuitenkaan näe merkittävää eroa palvelimen valinnassa.

Lopullinen valita palvelimen käytöstä jätettiinkin asiakkaan valinnaksi.

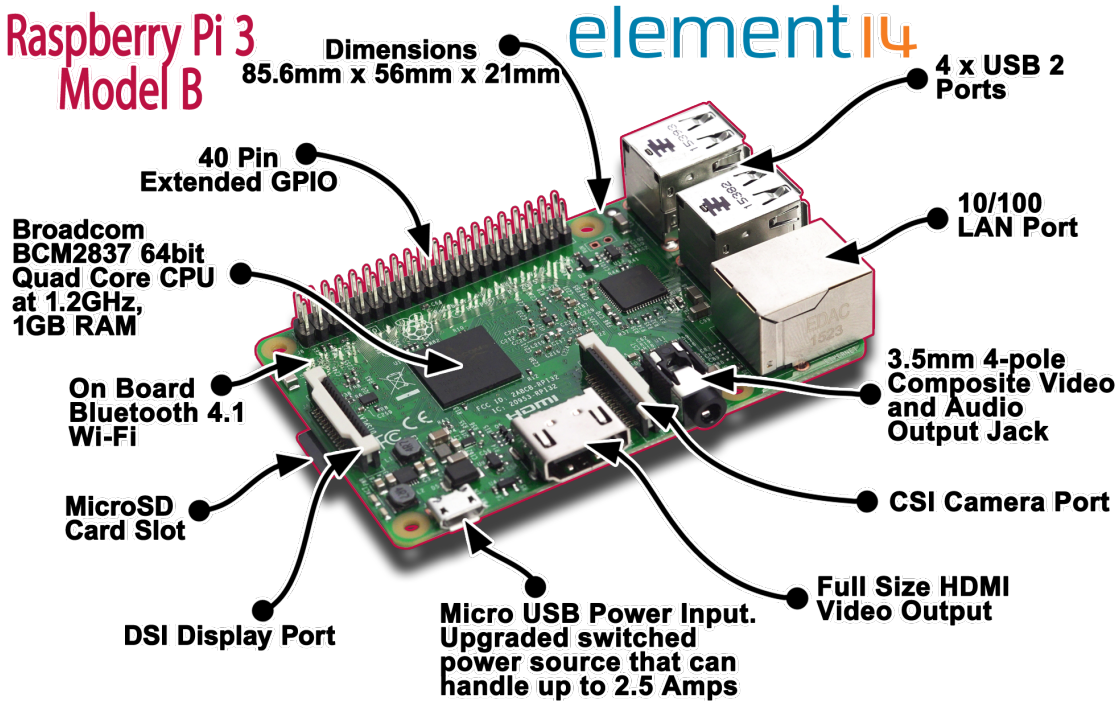
2.7. Laitteisto

Tilatiedon lukemista varten laitevaihtoehtona oli heti alussa Raspberry Pi, koska siitä oli aiempaa kokemusta. Muitakin yhden kortin tietokoneita on markkinoilla. Tarkempaa vertailua en korttien välillä kuitenkaan suorittanut.

2.7.1. Raspberry Pi

Raspberry Pi 3 (kuva 4) on edullinen yhden piirilevyn tietokone (Raspberry, 2016), jossa voidaan ajaa Linux–käyttöjärjestelmää, Windows 10 IoT Core, sekä muutamaa muuta kolmannen osapuolen käyttöjärjestelmää, joita ei tässä dokumentissa käsitellä.

Raspberry Pi ei kuitenkaan sisällä analogisia tuloja eikä –lähtöjä. Mahdolliset analogiset signaalit saadaan kuitenkin luettua elektroniikkaa lisäämällä.



Kuva 4: Raspberry Pi 3 (Element14, 2016)

2.7.2. Virransyöttö

Virransyötössä on huomioitava, että tavallinen puhelimen 500 mA:n USB-laturi ei joka tilanteessa anna riittävästi virtaa Raspberry Pi:lle. Esimerkiksi käynnistettäessä Raspberry Pi saattaa jossain vaiheessa aloittaa käynnistyksen alusta, vaikka virta olisikin näennäisesti päällä. Vähintään 2 A:n laturi onkin suositeltava. Raspberry Pi:n virrankulutus on tietokoneeksi melko pieni, mutta kuitenkin sen verran suuri, että akkukäyttöisyys on poissuljettu tässä sovelluksessa.

Raspberry Pi virransyötöstä on kerrottu enemmän osoitteessa: `https://www.raspberrypi.org/help/faqs/#power`

2.7.3. Reaaliaikakello

Raspberry Pi ei sisällä paristovarmistettua reaaliaikakelloa, joten virran katketessa Raspberry Pi menettää aikatiedon. Käynnistettäessä kello päivitetään Internetin NTP-palvelimelta. Kello päivittyy parissa minuutissa virran palautuessa. Mikäli Internet-yhteyttä ei ole saatavilla, kellonaika ei päivity oikeaksi. Kellonaika tallentuu kuitenkin käyttöjärjestelmätasolla siten, että käynnistettäessä kellonaika on ennen päivitystä lähes sama kuin virran katketessa. Kellonaika ei siis täysin nollaannu.

Valmiita reaaliaikakellomoduuleja on kuitenkin markkinoilla useita.

2.8. Tietovarastot ja niiden käsittely

Kävin läpi tietovarastoja ja niiden käsittelyyn käytettäviä ohjelmia (taulukko 7).

Taulukko 7: Tietovarastoja ja niiden käsittelyyn käytettäviä ohjelmia.

MySQL	Relaatiotietokantaohjelmisto, joka on hyvin yleisesti käytössä erilaisissa web-palveluissa.
MariaDB	Pohjautuu MySQL:ään ja sen pääkehittäjä on toinen MySQL:n alkuperäisistä kehittäjistä.
Apache Hadoop	Luotettava ja vikasietoinen ohjelmakirjastosto suurten tietomäärien hajautettuun käsittelyyn, joka skaalautuu yhdestä palvelimesta tuhansiin koneisiin (Hadoop, 2016).
Hadoop Common	Hadoop moduulien yhteiset osat.
HDFS™	Hadoop Distributed File System, Hadoop hajautettu tiedostojärjestelmä.
Hadoop YARN	Hadoop runko töiden aikataulutukselle ja resurssien hallintaan.

Hadoop MapReduce	YARN-pohjainen järjestelmä suurten tietomäärien rinnakkaiseen käsittelyyn.
Apache Drill	Apache Drill tukee monia tietokantoja ja tiedostomuotoja, josta voidaan suorittaa SQL-syntaksin mukaisia hakuja (Drill, 2016).
Apache HBase	Skaalautuva hajautettu tietokanta, joka tukee jäsennettyä tiedon varastointia suurille taulukoille (HBase, 2016). HBase tarjoaa Googlen BigTable –tyyppisiä ominaisuuksia Hadoop HDFS:n päällä.
Apache Hive	Tietovarasto infrastruktuuri suurten tietomäärien käsittelyyn hajautetussa järjestelmässä. Tukee SQL-syntaksia (Hive, 2016).
Apache Pig	Analysointialusta suurille tietomäärille korkean tason ohjelmointikielellä, joka tukee rinnakkaista käsittelyä (Pig, 2016).
Apache Spark	Nopea ja yleiskäyttöinen laskentamoottori Hadoop datalle (Spark, 2016).
Apache Storm	Hajautettu reaaliaikainen laskentajärjestelmä (Storm, 2016).

2.9. Datan visualisointi ja käyttöliittymä

Järjestelmä päätettiin tehdä siten, että käyttö tapahtuisi selainpohjaisesti. Tarvittiin siis jokin alusta, jonka päälle käyttöliittymä rakennetaan. Myös datan visualisointiin tarvittiin jokin tekniikka.

2.9.1. AngularJS

Harkitsin useita eri sisällönhallintaohjelmia, mutta ne ovat pääasiassa tarkoitettu staattisen sisällön hallintaan. Kuitenkin tässä tapauksessa sisällön ulkoasu pysyy samana, mutta esitettävä tieto on pääasiassa dynaamista. Käytettäväksi alustaksi valikoituikin ilman suurempia vertailuja AngularJS.

AngularJS on JavaScript-kirjasto, joka soveltuu hyvin selainpohjaisiin käyttöliittymiin (AngularJS, 2017). Sivu rakentuu selaimessa HTML-pohjista JavaScriptin avulla. HTML-pohjiin lisätään JavaScript-kirjaston tunnistamia elementtejä, joilla tuodaan sivuun dynaamista sisältöä tai toisia HTML-pohjia. Käyttöliittymää rakennettaessa saadaan ulkoasu pidettyä erillään HTML-pohjissa ja CSS-tyylitiedostoissa. Käyttöliittymän ohjelmallinen toiminnallisuus on JavaScript-tiedostoissa.

AngularJS on varsin laaja ympäristö, jonka opettelu vie paljon aikaa. Tässäkin työssä käytettiin vain pientä osaa kirjaston mahdollisuuksista.

2.9.2. Angular

Angular oli työn alkuvaiheessa vielä beta-vaiheessa, joten sitä ei käytetty työn ensimmäisessä vaiheessa. Siihen siirryttiin vasta jatkokehityksessä varsinaisen diplomityöosuuden päätyttyä. Angular on AngularJS:n tilalle kehitetty uudempi TypeScript-kirjasto ja kehitysympäristö (Angular, 2017). Suurin ero AngularJS:ään verrattuna on TypeScriptin käyttö JavaScriptin tilalla. TypeScriptissä on parannuksia JavaScriptiin verrattuna, mm. mahdollisuus käyttää luokkia sekä muuttujien tyypitys. Pääosin TypeScript on hyvin JavaScriptin kaltainen. Kehitysympäristö myös kääntää TypeScript-sovelluksen JavaScript-kielelle, joka sitten suoritetaan selaimessa.

Angular-ympäristön käyttö myös tuntuu selkeämmältä ja siten helpommalta kuin AngularJS-ympäristön.

2.9.3. Visualisointi

Visualisointiin ja kuvaajien piirtoon oli myös paljon eri vaihtoehtoja, joista kaksi nousi ylitse muiden: Charts.js ja D3.js. Useat vapaat ja kaupalliset työkalut myös pohjautuvat varsinkin D3.js:ään. Molemmat ovat open source JavaScipt-kirjastoja, joilla voidaan luoda HTML-sivulle kuvaajia ladatun datan perusteella. Chart.js on suppeampi ja yksinkertaisempi.

Chart.js luo kuvaajat HTML Canvas –elementiksi (Chart.js, 2017). D3.js mahdollistaa kuvaajien lisäksi laajemminkin datan visualisoinnin HTML- CSS- ja SVG-elementeillä (Bostock, 2017). D3.js onkin varsin laaja kirjasto, jonka opettelu vaatii aikaa. Eri versiot eivät ole täysin yhteensopivia, joten monet vanhemmat esimerkit eivät suoraan toimi uusimmalla versiolla.

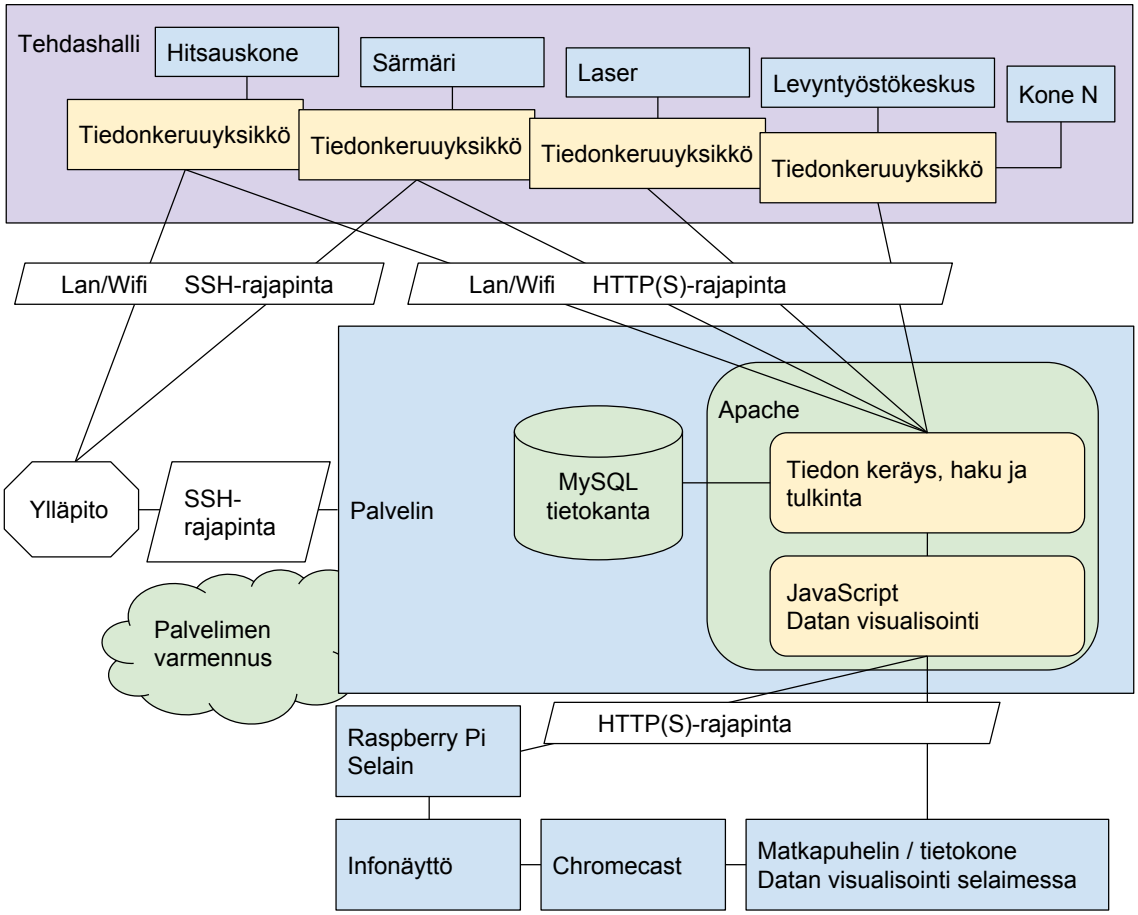
3. TOTEUTETTAVA JÄRJESTELMÄ

Toteutettava järjestelmä koostuu itsenäisistä osista, jotka esitellään tässä dokumentissa tarkemmin erikseen. Järjestelmän loogisia osia ovat (kuvassa 5):

- tiedonkeruuyksiköt,
- palvelin ja tietokanta,
- datan tulkinta ja visualisointi,
- järjestelmän hallinta.

Tiedonkeruuyksiköt ovat itsenäisiä fyysisiä yksikköjä, joissa voi kuitenkin olla useampiakin loogisia yksikköjä. Tiedonkeruuyksikkö tässä työssä tarkoittaa laitetta, jolla luetaan konepajan laitteiden tilatietoa ja lähetetään luettu tilatieto palvelimelle. Muut loogiset yksiköt voivat sijaita yhdessä tai useammassa fyysisessä palvelimessa. Datan visualisointi toimii selain–käyttöliittymällä, joka voi siten sijaita missä tahansa.

Järjestelmässä pyritään käyttämään valmiita standardeja laitteita, ohjelmia ja menetelmiä, jotka pyritään erottamaan toisistaan siten, että yksittäinen laite tai ohjelma voidaan myöhemmin vaihtaa, ilman että muihin tarvitsee tehdä muutoksia. Tavoitteena on laite- ja ohjelmistoriippumattomuus. Laitteet (tiedonkeruuyksiköt) tulee olla vaihdettavissa, siten että muuhun järjestelmään ei tarvitse tehdä muutoksia laitteiden vaihtuessa. Uusia laitteita kuitenkin tulee markkinoille ja vanhoja poistuu. Lisäksi muu järjestelmä ei ota mitään kantaa siihen, miten tiedonkeruuyksiköt keräävät tietoa, eikä tiedon analysointi- ja visualisointivaiheessa oteta kantaa siihen, mistä tai miten tieto on järjestelmään tullut. Tiedonkeruuyksiköt lähettävät tiedon sovitussa muodossa, riippuen siitä mitä mitataan. Ohjelmistoriippumattomuus voi olla vaikeampi toteuttaa. Suunnitteluvaiheessa täytyykin tehdä valintoja joustavuuden ja valmiiden ohjelmien välillä. Lisäksi täytyy tehdä valinta pilvipalvelun ja paikallisen palvelimen välillä. Järjestelmä on mahdollista rakentaa siten, että paikallisesti asennetaan vain tiedonkeruuyksiköt. Tiedon tallennus, analysointi ja datan visualisointi voivat tapahtua muualla sijaitsevassa pilvipalvelussa.



Kuva 5: Järjestelmän kaaviokuva. Kuvassa suorakaide on fyysinen laite, pyöristetty suorakaide on ohjelma, keltaisella pohjalla olevissa lohkoissa on järjestelmään tehtyjä osia ja ohjelmistoja.

3.1. Tiedonkeruuyksiköt

Alkuperäisen tavoitteen mukaan tiedonkeruuyksikön tehtävänä on mitata laitteista digitaalista tilatietoa, koska laite on käynnissä. Tiedonkeruuyksikkö lähettää mittaamansa tilatiedon palvelimelle. Tiedonkeruuyksikön tiedonsiirto tapahtuu salatun langattoman lähiverkon kautta, jolloin vältytään tiedonsiirtokaapeleiden asennukselta tehdashalliin. Tiedonsiirto voidaan myös tarvittaessa toteuttaa kaapelilla. Seurattavan laitteen käyttötieto kytketään tiedonkeruuyksikköön kaapelilla, jonka kautta seurattavasta laitteesta voidaan mahdollisuuksien mukaan ottaa myös virransyöttö tiedonkeruuyksikölle.

3.1.1. Laitteiston valinta

Tiedonkeruuyksikön tulee olla riittävän pieni, että se ei häiritse koneen käyttöä. Useimmat koneet ovat kuitenkin niin suuria, että koteloinnin koko ei ole merkitsevä asia, vain hitsauskoneet ovat siirrettäviä. Useissa laitteissa on myös mahdollisuus saada 24 V DC tiedonkeruuyksikön virransyöttöä varten.

Tiedonkeruuyksikköä ei ollut tarkoitus suunnitella kokonaan tätä tarkoitusta varten, vaan suunnittelussa oli lähtökohtana käyttää mahdollisimman valmiita komponentteja. Laitteen tulisi olla yleisesti saatavilla, edullinen, siinä tulee olla digitaalisia tuloja ja mahdollisuus langattomaan tiedonsiirtoon. Lisäksi tiedonkeruuyksikössä tulee olla sen verran laskentakapasiteettia, että tarvittaessa voidaan suorittaa pienimuotoista signaalinkäsittelyä.

Valinnassa päädyttiin yhden piirilevyn tietokoneeseen, tässä tapauksessa Raspberry Pi 3. Raspberry Pi 3 on yksinkertaiseen tiedon käsittelyyn reilusti ylimitoitettu, mutta sillä voidaan tarvittaessa suorittaa pienimuotoista signaalinkäsittelyä ja lisäksi siinä on valmiina langaton lähiverkkoliitäntä. Kortti on lisäksi edullinen. Toinen varteenotettava vaihtoehto on Raspberry Pi Zero, joka on vielä pienempi ja halvempi. Raspberry Pi Zero:n saatavuus on kuitenkin heikko. Yhden piirilevyn tietokoneita on toki muitakin, Raspberry Pi lienee kuitenkin yksi yleisimmistä. Lisäksi tarvitaan riittävä kotelointi ja elektroniikkaa virransyöttöön sekä signaalin sähköiseen erottamiseen.

Käyttöjärjestelmänä käytetään Linux-pohjaista ’Rasbian Jessie Lite’ –käyttöjärjestelmää, ilman graafista käyttöliittymää. Lisäksi testausvaiheessa on käytössä myös ’Raspbian Jessie with PIXEL’ graafisella käyttöliittymällä, joka on laajempi asennus. Asennusohjeet ovat liitteessä LIITE 1 kohdassa 1.1. ja 1.2..

Mikäli tiedonkeruuyksikköön halutaan tiedon syöttö- ja näyttömahdollisuus, voidaan Raspberry Pi korvata tabletilla. Markkinoilla on olemassa USB- ja Bluetooth-liitäntäisiä tiedonkeruukortteja, joilla tiedon keruu saattaisi olla mahdollista suoraan tabletilla. Raspberry Pi voi toki olla keräämässä tietoa tabletin rinnallakin. Tabletit ovat nykyään melko

edullisia ja suuri osa ihmisistä on tottunut kosketusnäyttöjen käyttöön. Tabletti voidaan lisäksi upottaa koteloon, jolloin se ei haluttaessa edes näytä tabletilta.

Ensimmäisessä vaiheessa seurattavista laitteista luetaan vain digitaalista tilasignaalia. Lisäksi tiedonkeruuyksiköön voidaan laittaa merkkivalot osoittamaan laitteen tilaa — virta päällä ja laite toiminnassa.

Koska tiedonkeruuyksikössä on riittävästi laskentatehoa, voidaan luettava tieto käsitellä valmiiksi oikeaan muotoon tietokantaan tallennettavaksi.

Tiedonkeruuyksikön sisäisen tallennustilan loppumisen estämiseksi, ei tallenneta mitään lokia toiminnasta, tai lokitiedostojen koolle on asetettava yläraja. Lokien hallinta onkin Raspbian Linuxissa valmiina.

3.1.2. Hallinta

Tiedonkeruuyksikön asetusten muuttamista varten tiedonkeruuyksiköön tulee päästä jotenkin käsiksi. Raspberry Pi tiedonkeruuyksikkö on Linux–tietokone, joten tapoja on useita. Massamuistina toimii micro SD korttipaikka, joten micro SD kortille voidaan suoraan muuttaa tarvittavat asetukset (Liite LIITE 1 kohta 1.4.). Laitteeseen voidaan liittää USB-hiiri ja –näppäimistö sekä näyttö HDMI-liitännällä. Laitteeseen voidaan ottaa myös SSH-yhteys kaapelilla ethernet-portin tai langattoman verkon kautta, joka onkin todennäköisin tapa asetusten muuttamiseen.

3.1.3. Tiedonsiirto

Tiedonsiirto tiedonkeruuyksiköltä palvelimelle tapahtuu HTTP– tai HTTPS-protokollaa käyttämällä. Mikäli langattomassa verkossa tulee häiriöitä, laite voi puskuroida kerättyä tietoa, kunnes yhteys on jälleen saatavilla.

Mitattavaa tietoa muokataan vain sen verran, kuin tiedon tallennuksen kannalta on tarpeen.

Tieto tiedonkeruuyksiköltä palvelimelle siirretään HTTP GET, PUT tai POST pyynnöllä. Palvelinpään virheen sattuessa tiedonkeruuyksikkö yrittää tiedonsiirtoa uudelleen määritellyn ajan kuluessa, ei kuitenkaan välittömästi, ettei laajemmassa häiriössä verkko ruuhkaannu. Virheen sattuessa tiedon keruuta kuitenkin jatketaan normaalisti ja tieto tallennetaan puskuriin myöhempää lähetystä varten. Kun luetaan tilatietoa, tiedonkeruuyksikkö lähettää tilan muuttuessa aina uuden viestin, sekä nykyisen tilan määrätyin välein (esim. 30 sekuntia). Vikatilanteen sattuessa, viimeisin lähetetty tilatieto ennen uudelleen käynnistystä on varmistettu tieto, jota voidaan käyttää hyväksi tilatietoja tulkittaessa. Ilman jatkuvaa tilatiedon ja uudelleenkäynnistystiedon lähettämistä, ei voida olla varmoja mittauksen jatkuvuudesta tilatietojen välillä. Lohkokaavio tilatiedon lukemisesta ja lähetyksestä on kuvassa 6.

Käyttämällä standardia HTTP(S) protokollaa, tiedon tallentava palvelin voi periaatteessa sijaita missä tahansa, kunhan tiedonkeruuyksikkö saa siihen yhteyden.

3.1.4. I/O-porttien luku

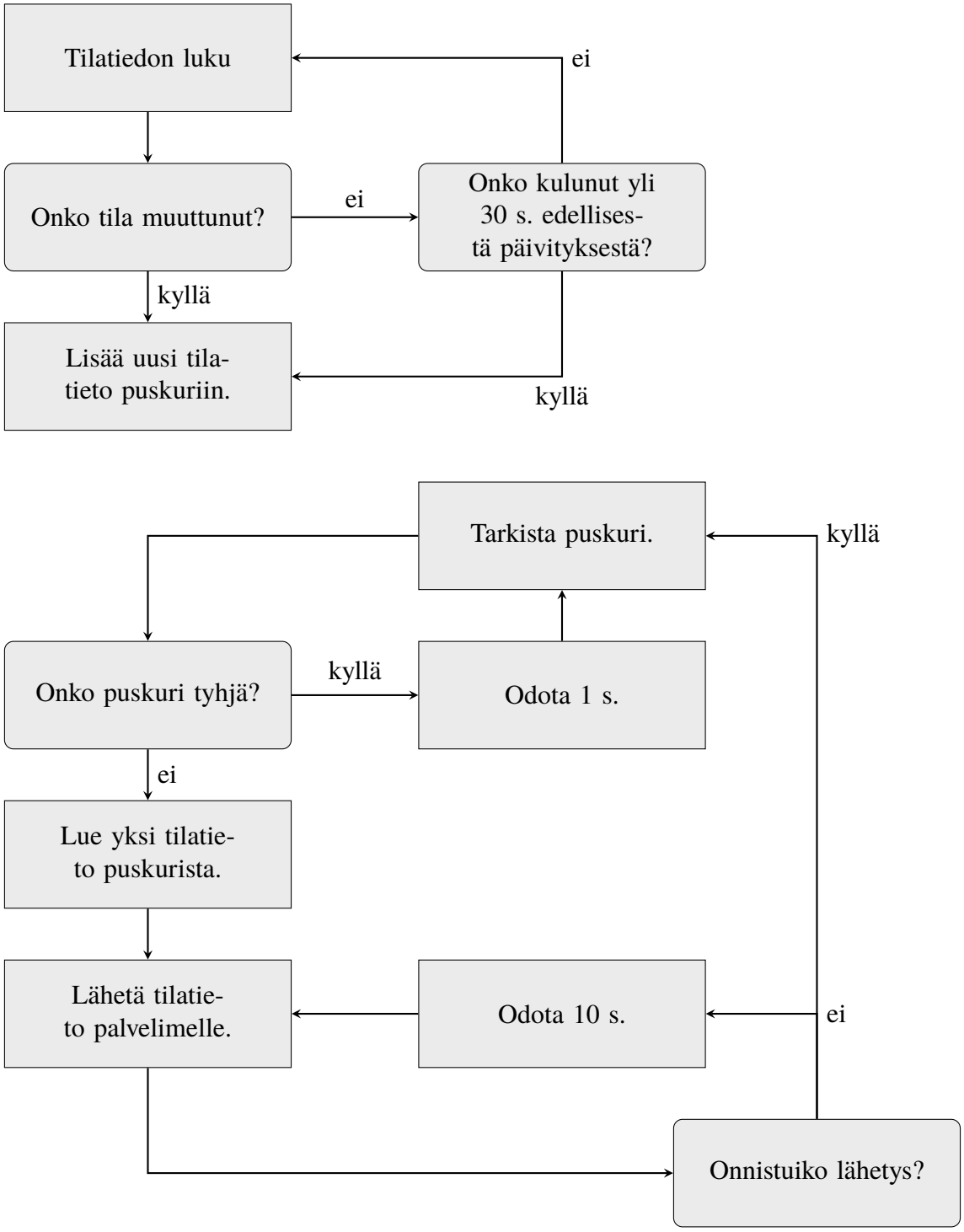
GPIO-porttien luku tapahtuu Python-kielisellä ohjelmalla, joka käyttää *raspberry-gpio-python* –kirjastoa. Kirjaston ohjeet löytyy osoitteesta:

`https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/`

Liitteessä LIITE 6 on esimerkkinä Python-kielinen ohjelma, joka lukee yhden GPIO-portin tilan ja asettaa toisen, sekä lähettää tiedon ThingSpeak-palvelimelle. GPIO-portti voidaan määritellä lähdöksi tai tuloksi. Tuloportille voidaan vielä määritellä käytetäänkö ylös-, alasvetovastusta tai ei mitään.

3.1.5. Tilatiedon lukeminen

Koneilta tallennetaan vain suoraan mitattavissa oleva käyttöaika tilatietona, ei häiriöitä tai odotusaikoja. Tilatiedon luku koneilta pyritään tekemään siten, että koneen toiminta ei mis-



Kuva 6: Lohkokaavio, tilatiedon luku ja lähetys palvelimelle.

sään tapauksessa häiriintyisi. Useissa työstökoneissa on jonkintasoinen tietokoneohjaus, joka lisäksi on yhteydessä yrityksen lähiverkkoon. Mikäli työstökoneissa olisi valmiina

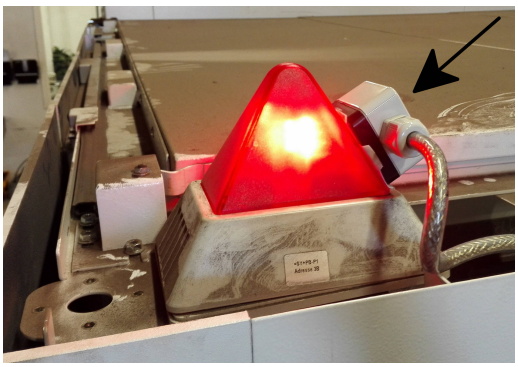
rajapinta, joka lähettäisi tilatiedon haluttuun paikkaan, erillisiä tiedonkeruuyksiköitä ei edes tarvittaisi. Työstökoneet ovat kuitenkin eri ikäisiä ja eri valmistajien tekemiä, joten koneiden toimintaan ei haluttu puuttua millään lailla.

Tilatietoa luetaan seuraavilta koneilta:

- Laser: Tilatiedon mittaus tapahtuu laserin varoitusvalosta (kuva 7), jonka päälläolo tunnistetaan fototransistorilla. Varoitusvalo on päällä, kun leikkuuohjelma on käynnissä. Varoitusvalo on päällä, myös silloin kun leikkuupää siirtyy seuraavaan kohteeseen, eikä laser ole päällä. Levyn vaihdon yhteydessä varoitusvalo ei ole päällä.
- Levyntyöstökeskus: Tilatieto saadaan logiikalta, joka antaa iskukäskyn. Tilatieto on vain lyhyt pulssi. Reikien kierteytystä ei tunnisteta.
- Särmärit: Koneen kytkentäkaapista logiikalta mitataan suoraan koneen käyttöpolkimen tilatietoa. Koneen käyttötavasta riippuen käyttäjä voi yhtä taitosta varten painaa poljinta yhden tai useamman kerran. Työkalu liikkuu alaspäin vain silloin, kun poljinta pidetään alhaalla.
- Hitsauskoneet: Mitataan paloaikaa kaasunsyötön magneettiventtiilistä, tai maajohdosta.
  - Kemppi MasterTig AC/DC 3500W: Tilatieto saadaan kaasunsyötön magneettiventtiilin ohjauksesta (24 V DC). Tiedonkeruuyksikön virransyöttö saadaan muuntajan jälkeen ohjausyksikön virransyötöstä (24 V AC).
  - Kemppi FastMig Pulse 350/FastMig MXF 65: Kaasunsyötön magneettiventtiili (24 V DC) on helposti käsillä langansyöttöyksikössä, mutta langansyöttöyksikön ohjaukselle tulee 50 V DC. Tiedonkeruuyksikön virransyöttö toteutetaan erillisellä muuntajalla.
  - Kemppi Kempact 323A: Laite on muovikuorinen ja kaasunsyötön magneettiventtiiliin käsiksi pääseminen vaatii mm. langansyöttökoneiston irroittamisen. Mittaus tapahtuu maajohdosta reed-releellä. Maajohdosta kierretään kolmen kierroksen kela riittävän suuren magneettikentän aikaansaamiseksi. Reed-rele asetetaan kelan keskelle (kuva 8). Kun tilatieto otetaan maajohdosta, on tie-

donkeruuyksikön asennus huomattavasti helpompi ja nopeampi, kuin hitsauskoneen sisältä otettava tilatieto.

– Aristotig 255: Tilatieto otetaan maajohdosta samalla tavalla kuin Kempact 323A.



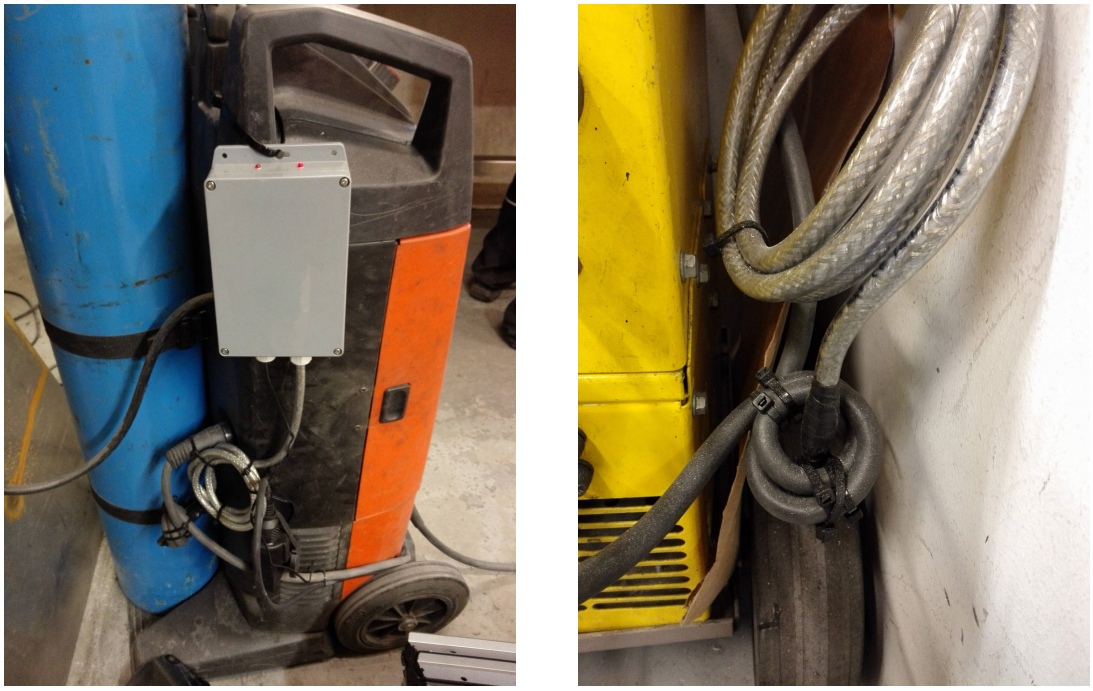
Kuva 7: Tilatiedon mittaus laserin varoitusvalosta. Nuoli osoittaa koteloidun fototransistorin.

Taulukossa 8 on esitetty yhteenvetona laitteiden tilatiedon lukutapa ja tiedonkeruuyksikön virransyöttö.

3.1.6. Robottisolun seuranta

Robottisolu koostuu kappaleita siirtelevästä robottikäsivarresta, särmäristä ja toiseen robottikäsivarteen liitetystä hitsauskoneesta. Robottisolun seurantaan päätettiin kokeilla kameravalvontaa. Kameravalvonta asennetaan Raspberry Pi 3 –kortille, johon on asennettu Raspberry Pi kameramoduuli. Käyttöjärjestelmänä on Linux-pohjainen motionEyeOS, jota ei ole tarkoitettu yleiskäyttöiseksi Linux jakeluksi (Crisan, 2017).

Tilatieto lähtee, kun ohjelma huomaa liikkeen kuvassa. Liikkeen jatkuessa, tai loppuessa tilatietoa ei lähetetä. Asetuksista liikkeen herkkyyttä ja liikkeen tunnistuksen aikaväliä säätämällä voidaan saada riittävä tarkkuus robottisolun seuraamiseen. Kun liikkeen tunnistus tapahtuu riittävän usein tilatiedon voimassaoloajan puitteissa, saadaan robotin liikkuessa



(a) **(b)**

Kuva 8: Hitsiauskoneiden tilatiedon mittaus maajohdosta. Oikean puoleisessa kuvassa musta maajohto on kierretty kolmesti kelalle, jonka keskelle on asettettu koteloitu reed-rele. Tiedonkeruuyksikkö on harmaassa kotelossa, johon reed-rele on kytketty harmaalla kirkkaalla johdolla.

jatkuva tilatieto. Jos robotti kuitenkin liikkuu koko ajan, ei tilatietoa lähetetä uudelleen. Robottisolun seuranta onkin toiminut riittävällä tarkkuudella.

Kuvassa 9a on kamera harmaassa laatikossa kuvaamassa robottisolua. Kuvassa 9b on kameran näkymä robottisolusta.

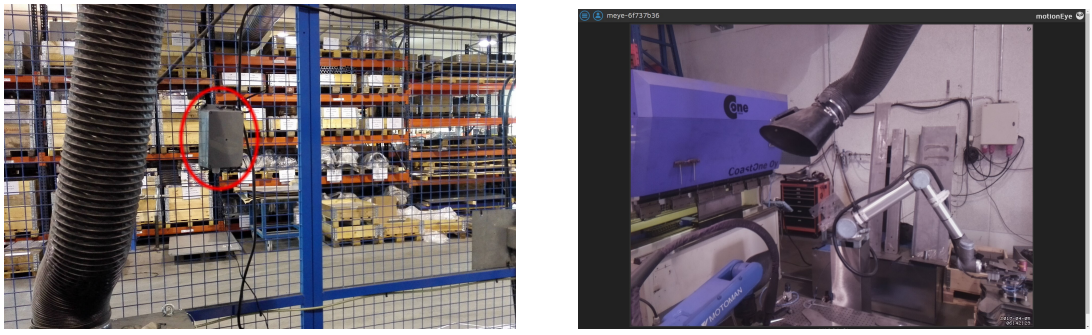
3.1.7. Laitteisto

Raspberry Pi:n GPIO-liityntään ei voida suoraan kytkeä 24 V:n tilatietoa koneilta, vaan väliin tarvitaan jotain suojaamaan tuloliitäntöjä. Tuloliitäntöjen suojana käytetään optoerotinpiiriä sopivasti mitoitetuilla vastuksilla.

Liitteessä LIITE 3 on piirikaavio (LIITE 3.1) kolmen tilatiedon lukua varten. Kolme sär-

Taulukko 8: Laitteiden tilatiedon lukutapa ja tiedonkeruuyksikön virransyöttö.

Laite	Tilatiedon lukutapa	Virransyöttö
Laserleikkuri	Varoitusvalosta fototransistorilla.	Logiikan 24 V:n virransyöttö.
Levyntyöstökeskus	Iskukäsky logiikalta.	Logiikan 24 V:n virransyöttö.
Särmärit	Käyttöpolkimen tilatieto logiikalta.	5 V:n USB verkkomuuntaja.
Hitsauskone Kemppi MasterTig	Kaasunsyötön magneettiventtiili.	Ohjausyksikön 24 V:n virransyöttö.
Hitsauskone Kemppi FastMig	Kaasunsyötön magneettiventtiili.	5 V:n USB verkkomuuntaja.
Hitsauskoneet Kemppi Kempact	Reed-releellä maajohdosta	5 V:n USB verkkomuuntaja.
Hitsauskone Aristotig	Reed-releellä maajohdosta	5 V:n USB verkkomuuntaja.



(a) **(b)**

Kuva 9: Vasemmalla kamera kuvaamassa robottisolua, oikealla kameran näkymä robottisolusta.

märiä ovat niin lähellä toisiaan, että niiden tilatiedot voidaan lukea samalla tiedonkeruuyksiköllä. Virransyöttö oli tarkoitus saada yhdeltä särmäriltä 24 voltin tasavirtalähteestä. Tässä tapauksessa kuitenkin päädyttiin erilliseen USB-virtalähteeseen, koska särmäreistä voidaan katkaista virrat toisistaan riippumatta. Tilatietojen lukua varten kytkennässä on valmiina tulot 5, 12 ja 24 voltin tulojännitteille, jotka voivat olla tasa- tai vaihtovirtaa. Kytkettäessä tiedonkeruuyksikköä valitaan tuloliitin jännitteen mukaan. Vaihtovirtasignaalin

lukua varten kytkennässä on kondensaattori, joka voidaan kytkeä irti, mikäli luetaan tasavirtasignaalia. Kuvassa LIITE 3.5 on ensimmäinen prototyyppi tiedonkeruuyksiköstä, johon on merkitty tulojen kytkentä.

Liitteessä LIITE 3 on piirikaavio (LIITE 3.2) yhden tilatiedon lukua varten. Ensimmäiseen prototyyppiin verrattuna tulossa ei ole useaa eri vastusta tulolle, koska optoerottimen tulossa on melko laaja virta-alue. Näitä käytetään hitsauskoneissa, joista tilatieto otetaan kaasunsyötön magneettiventtiilin ohjauksesta. Kuvassa LIITE 3.6 on toinen prototyyppi tiedonkeruuyksiköstä, johon on myös merkitty tulojen kytkentä.

Liitteessä LIITE 3 on piirikaavio (LIITE 3.3) hitsauskoneen tilatiedon lukua varten, kun mittaus tapahtuu maajohdosta reed-releellä. Tiedonkeruuyksikkö on kuvassa LIITE 3.7. Reed-rele on ohuessa lasiputkessa, joka rikkoutuu todella helposti. Tein alumiinista pienen suojan reed-releelle.

Liitteessä LIITE 3 on piirikaavio (LIITE 3.4) laserleikkurin tilatiedon lukua varten. Tilatiedon luku tapahtuu varoitusmajakasta fototransistorin avulla. Tiedonkeruuyksikkö on kuvassa LIITE 3.8 ja fototransistori koteloituna kuvassa LIITE 3.9.

Osassa työstökoneista otetaan tiedonkeruuyksikön virransyöttö suoraan koneen logiikan 24 V:n virransyötöstä. Jännitteen muuntamiseen käytetään ajoneuvoihin tarkoitettua 24 V - 5 V USB –muunninta. Virransyöttö ei ole aiheuttanut mitään ongelmia.

3.2. Palvelin

3.2.1. Palvelimen vaatimukset

Järjestelmä tulee toteuttaa siten, että se toimii pienellä mittakaavalla hyvin kevyessä tietokoneessa, mutta on myös tarvittaessa skaalattavissa suurempaan tietokoneeseen. Itse käyttöjärjestelmän ylläpitoon ei oteta tässä mitään kantaa. Järjestelmä toimii Windows-

tai Linux-palvelimessa. Järjestelmä voisi toimia myös virtuaalikoneen päällä Windows-tietokoneessa. Olemassa olevien palvelinten toimintaan pyritään puuttumaan mahdollisimman vähän.

3.2.2. Tiedon varmistus

Tietokanta voi olla tallennettuna erikseen varmennetulle verkkolevylle tai paikalliselle koneelle. Lisäksi koko palvelin voidaan varmentaa verkon yli, jolloin palvelimen vikaantuessa on koko palvelin nopeasti palautettavissa.

3.3. Tiedon tallennus

3.3.1. Tallennettavan tiedon määrittely

Jokainen kone ja mitattava tieto on voitava yksilöidä ja tallennus tulee tehdä siten, että tieto on kohtuullisella vaivalla luettavissa. Tilatietojen mittauksessa tilatietoja on odotettavissa korkeintaan keskimäärin 10 sekunnin välein, jolloin vuositasolla jatkuvalla mittauksella saadaan kaikilta koneilta yhteensä suuruusluokkaa 3 000 000 mittaustietoa. Tallennustilan kannalta määrä ei ole vielä liian suuri, kunhan tieto tallennetaan tiiviissä muodossa.

Miten koneelle/mitattavalle kohteelle annetaan täysin yksilöllinen tunniste? Tunniste ei saa myöskään olla riippuvainen käytettävästä tiedonkeruuyksiköstä, koska vikatilanteessa tiedonkeruuyksikkö tulee olla vaihdettavissa.

1. Ratkaisu: Tieto tallennetaan hierarkisessa hakemistorakenteessa (yritys/paikka/koneen.nimi/kohde), jossa paikka on konesalin nimi, koneen.nimi on koneen lyhyt tunniste ja kohde mitattavan kohteen tunniste.
    - Koneen tunnistetietoa ei tarvitse tallentaa jokaiselle mittaustiedolle erikseen.
    - Miten mitattavan kohteen tyyppi (digitaali, analogi) tunnistetaan?
2. Ratkaisu: Tieto tallennetaan yhteen SQL-taulukkoon.

- Koneen tunnistetietoa joudutaan tallentamaan jokaiselle mittaustiedolle erikseen.

3. Ratkaisu: Jokaiselle mitattavalle kohteelle annetaan tietokannassa oma numeerinen tunniste. Tunnisteen avulla luodaan jokaiselle kohteelle myös oma taulunsa tietokantaan. Nyt jokaisesta mittaustiedosta ei tarvitse tallentaa kohteen tunnisteita.

3.3.2. Tietokanta

Tiedon tallennukseen käytetään MySQL tietokantaa. MySQL:n valintaan vaikutti lähinnä laaja tuettavuus.

Kaikki tekstipohjainen tieto tallennetaan UTF-8 –koodauksella, ellei toisin mainita. Sama koskee myös tiedon esitystä.

Tietokannan luontitiedosto on liitteessä LIITE 7. Tietokanta sisältää seuraavat taulut:

- *settings* Taulussa on yleiset ja ryhmäkohtaiset asetukset ja autentikointiavaimet.
- *groups* Taulussa on koneryhmän tiedot.
- *sensors* Taulussa on koneen tiedot.
- *msgStatus_nn* Taulussa on yhden koneen tilatiedot, taulun nimessä *nn* on anturin numero. Tämä taulu luodaan koneen lisäyksen yhteydessä.

3.3.3. Tallennettava tieto

Seurattavasta laitteesta tallennetaan kerralla vähintään seuraavat tiedot:

- koneen tai mitattavan kohteen tunnistetieto
- aikaleima (32-bittinen tai 64-bittinen)
- tilatieto (64-bittinen)

Koneen tilan oletetaan pysyvän muuttumattomana, kunnes tulee uusi tilatieto. Kaksi peräk-

käistä samaa tilatietoa eri aikaleimalla sallitaan, tilan pysyessä samana. Tilatietoja lisäksi odotetaan saapuvan tietyin määrävälein (30 s), jolloin voidaan varmistaa tiedon jatkuva mittaus. Mikäli kahden mittauksen välinen aika on riittävän suuri (> 60 s), voidaan olettaa mittauksen keskeytyneen.

Tietokantaan tallennettua tietoa ei enää muuteta ja tieto tallennetaan mahdollisimman muokkaamattomana. Kaikki tieto tulkitaan vasta luettaessa.

Lisäksi tiedonkeruuyksiköltä voidaan tarvittaessa tallentaa muuta myöhemmin määriteltävää tietoa, esimerkiksi kuvaa, ääntä tai mittaustietoja.

Koneen tunnistetietoa vastaa lisäksi lyhyt tekstipohjainen otsikkokenttä.

3.4. Datan visualisointi

Datan visuaalinen esittäminen tapahtuu selainpohjaisesti. Palvelimella data tuotetaan sellaiseen muotoon, että se on esitettävissä lähes millä tahansa laitteistolla, jossa on Internet-selain. Dataa voidaan siten esittää mobiililaitteilla tai isolla näytöllä. Datan visuaalisen esityksen toimivuus tulee todeta mobiililaitteella, joka ei välttämättä ole ihan huipputason laite, sekä tietokoneen isolla näytöllä.

3.4.1. Ohjelmointirajapinta

Tutustuttuani erilaisiin visualisointirajapintoihin, päätin käyttää tässä työssä D3.js JavaScript kirjastoa. D3.js on melko laajasti käytössä ja sillä pystyy tekemään näyttäviäkin grafiikoita. Kuvaajat luodaan käyttämällä HTML, SVG ja CSS –elementtejä, joten kuvaajat voidaan helposti luoda myös ilman D3.js kirjastoa. Tarve käyttää D3.js kirjastoa tulee kuitenkin siinä vaiheessa, kun halutaan luoda grafiikkaa dynaamisesti datan vaihtuessa. Ihan pieneen tarpeeseen D3.js rajapintojen käyttöä ei kannatakaan opiskella, vaan on nopeampaa ja helpompaa ohjelmoida sellainen itse JavaScriptillä. Grafiikan ohjelmointiin

D3.js kirjaston avulla löytyy useita esimerkkejä. Lisäksi käytän tässä työssä AngularJS –JavaScript kirjastoa, joka on tarkoitettu dynaamisten verkkosivujen luontiin.

Harkitsin myös *gnuplot* kuvaajienpiirto-ohjelman käyttöä, jolloin grafiikan luonti tapahtuisi palvelimella. Järjestelmän rajapintojen kannalta pidin kuitenkin parempana, että grafiikan luonti tapahtuu selaimessa ja data siirretään sopivassa muodossa palvelimelta.

3.4.2. Näytettävä tieto

Tavoitteena on näyttää kustakin seurattavasta kohteesta koneen absoluuttinen käyttöaste, käyttöaste verrattuna viitearvoon, aamu ja iltavuoron välinen ero ja ero viikonpäivien välillä.

Tiedon tulkinta eri kohteille:

- Laser ja levyntyöstökeskus: Laser ja levyntyöstökeskus voivat olla käynnissä pitkiäkin aikoja, myös öisin ja viikonloppuisin. Näytettävä tieto on käyttöaikasuhde, tai suora tilatieto pienellä aikavälillä. Normaali käyttöaikasuhde onkin 100 %.
- Särmäri: Lasketaan iskujen määrää aikavälillä (iskua/tunti). Iskujen määrä tunnissa voi olla prototyypin teossa 1 - 30 ja pienillä kappaleilla jopa 300.
- Hitsauskone: Lasketaan paloaikasuhde. Normaali paloaikasuhde odotetaan olevan käsinhitsauksessa alle 25 %, ja robottihitsauksessakin alle 50 %.

Käyttöaikasuhteen ja paloaikasuhteen yksi mittaustieto lasketaan tarkasteltavan aikavälin kahden pois-päällä –tilamuutoksen väliseltä ajalta.

Datan visualisoinnissa tietoja verrataan keskiarvoon, edelliseen päivään, viikonpäivien välillä ja aamu- ja iltavuorojen välillä. Tieto ryhmitellään koneryhmittäin. Näytettäviä aikavälejä ovat: viimeiset 8 työtuntia, viikko ja kuukausi. Näytettävä tieto voidaan hakea selaimella ajankohdittain.

3.4.3. Infonäyttö

Koneiden käyttöaikaseuranta näytetään työntekijöille taukotilassa olevalla televisiolla. Kuva voidaan siirtää televisioon suoraan tietokoneesta, Google Chromecastin avulla kauempana olevasta tietokoneesta, yksi Raspberry Pi liitettynä televisioon tai käytetään älytelevisiota, jossa on selain itsessään. Asiakkaalla oli kuitenkin valmiina televisio, jota tultaisiin käyttämään infonäyttönä. Tarvitaan siis jokin tietokone, jossa on selainikkuna auki. Asennusohjeet infonäytölle ovat liitteessä LIITE 1 kohdassa 1.8..

3.4.4. Intel Compute Stick

Vaikka Raspberry Pi soveltuukin hyvin infonäytön ohjaukseen, saatavilla oli Intel Compute Stick –laite, jossa oli Windows 10 asennettuna. Intel Compute Stick on fyysisiltä mitoiltaan jopa Raspberry Pi:tä pienempi ja valmiiksi koteloitu tietokone, varustettuna yhdellä HDMI-liittimellä ja yhdellä USB-liittimellä. Laite sisältää 2 GB käyttömuistia ja 32 GB sisäistä tallennustilaa.

3.5. Rajapinnat

Hyvin toteutetut rajapinnat järjestelmän eri osien välillä mahdollistavat järjestelmän eri osien suunnittelun ja testauksen toisistaan riippumatta. Järjestelmän eri osat ovat myös tarpeen vaatiessa vaihdettavissa, muiden osien pysyessä muuttumattomina.

3.5.1. Tietokanta

Tietokantana käytetty MySQL sisältää lisäyksiä standardiin SQL:ään verrattuna, joten tietokannan vaihtaminen ei onnistune suoraan. Kuitenkin tietokannan vaihtuessa muutokset eivät välttämättä ole kovin suuria. Tietokantaa käyttää suoraan vain palvelimella ajettava sovellus. Tietokannan luonti ja samalla tietokannan rakenne on liitteessä LIITE 7.

3.5.2. Autentikointi

Ensimmäisessä vaiheessa autentikointi suoritetaan käyttämällä *APIKey* metodia. Kaikissa REST API –kutsuissa lähetetään *Key* –niminen argumentti, joka sisältää avaimen kyseisen toiminnan toteuttamiseen. Avain on 16-64 merkkiä pitkä merkkijono. Tiedon kirjoitukselle on *WriteKey*, jota anturiyksiköt käyttävät tiedon päivittämiseen. *WriteKey* on ryhmäkohtainen. Tiedon lukemiseen on *ReadKey*, jota käytetään datan visualisoinnissa. Myös *ReadKey* on ryhmäkohtainen. Hallintaa varten on *AdminKey*, jota käytetään koneiden tietojen hallinnassa.

Tietoturvan kannalta tämä ei ole kaikkein turvallisin ratkaisu. Se on kuitenkin yksinkertainen ja helppo toteuttaa. Koska avain lähetetään salaamattomana, kaikki liikenne tuleekin olla salattua. Avain ei ole ulkopuolisen luettavissa, kun käytetään salattua HTTPS-protokollaa. Eri toiminnoille käytetään myös eri avainta, joten tiedon lukemiseen käytetyllä avaimella ei ole oikeuksia tehdä mitään muutoksia.

3.5.3. REST API

Palvelin ja REST API –rajapinta ovat koko järjestelmän kannalta keskeisin osa. Samaa REST API rajapintaa käyttävät sekä Raspberry Pi anturiyksiköt tiedon lähetykseen palvelimelle, että selain tiedon hakemiseen palvelimelta JavaScriptillä datan visualisointia varten.

Rajapinnan toteutustapa on REST API (Fredrich, 2016; Fielding, 2000). REST API on käytössä monessa paikassa ja on yksinkertaisuudessaan todella nerokas.

REST API toimii HTTP(S) -protokollan päällä. Esimerkiksi GET-pyyntö

```
http://192.168.1.107/iot/v1_alpha/sensors/1?
    usage=0&stop=2017-01-16T12:30:38.1&
    start=2017-01-16T08:18:00.0Z
```

palauttaa JSON-muodossa koneen numero 1 tilojen suhteet halutulla aikavälillä. Palve-

limen osoitteen jälkeen tulee rajapinnan nimi (`iot`) ja rajapinnan versio (`v1`, `v2`), jonka jälkeen tulee varsinainen rajapintapyyntö. Rajapinta on kuvattu tarkemmin liitteessä LIITE 2.

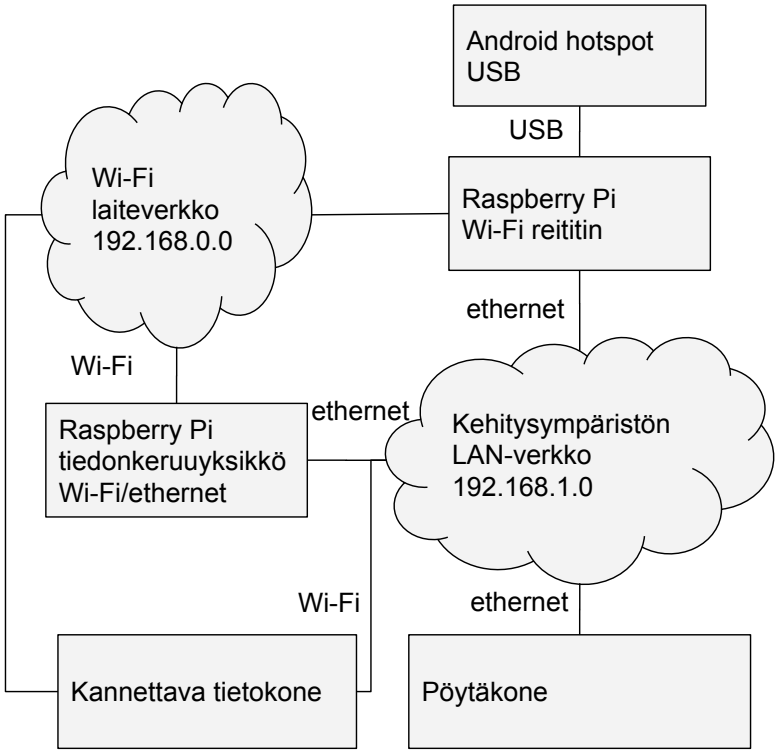
3.6. Kehitys- ja testausympäristö

Kehitysympäristönä käytin Windowsille asennettavaa *Windows Subsystem for Linux* – Linux-ympäristöä (Microsoft, 2017d). Kehitysympäristöön asensin palvelinsovellukset ja kaikki kehitettävät ohjelmat on linkitetty suoraan Windows-hakemistoon, jossa kehitys tapahtuu. Näin palvelimen ohjelmistoa kehitettäessä muutokset ovat suoraan nähtävillä, ilman tiedostojen siirtoa. Vain tiedonkeruuyksikön ohjelmistoa kehitettäessä testaus tapahtui itse Raspberry Pi –laitteella.

3.6.1. Verkko

Tiedonkeruuyksiköiden on tarkoitettu toimivan omassa Wi-Fi-verkossaan. Tiedonkeruuyksiköt voivat käyttää dynaamista IP-osoitetta tai kiinteää IP-osoitetta. Koska tiedonkeruuyksiköt lähinnä lähettävät tilatietoa, ei kiinteä IP-osoite ole välttämätön. Kuitenkin etähallinnan kannalta kiinteä IP-osoite voi olla helpommin hallittavissa.

Tiedonkeruuyksiköiden kannalta verkon halutaan pysyvän samanlaisena kehitys-, demo-, ja testausvaiheessa. Kuvassa 10 on kuvattu kehitysympäristön lähiverkko. Tiedonkeruuyksiköt ovat omassa langattomassa lähiverkossaan, joka on kehitysympäristöstä täysin erillään. Yksi Raspberry Pi on asetettu reitittimeksi verkkojen välille, sekä se toimii samalla palvelimena, johon tiedonkeruuyksiköt lähettävät tietonsa. Demokäytössä Android-puhelimesta jaetaan Internet-yhteys Raspberry Pi –reitittimen kautta. Kannettava tietokone toimii kaikissa verkoissa. Tiedonkeruuyksiköiden langattomasta lähiverkosta pääsee kehitysympäristön lähiverkkoon, mutta ei toisinpäin. Kehitysympäristöstä saadaan tiedonkeruuyksiköihin yhteys kytkemällä ne kehitysympäristön ethernet-lähiverkkoon.



Kuva 10: Kehitysympäristön kuvaus.

Reitittimen asennusohjeet ovat liitteessä LIITE 1 kohdassa 1.9..

3.6.2. Testaus

Käyttöliittymän testaukseen käytin pääasiassa Google Chrome-selainta, jossa on hyvin toimiva debuggeri.

Tietokannan luontia varten tein useamman pienen ohjelman, jotka luovat tietokantaan koneet ja satunnaista tilatietoa.

3.7. Ohjelmisto

Järjestelmää varten on tehty kolme eri ohjelmistoa: tiedonkeruuyksiköille, palvelimelle ja selainpohjainen käyttöliittymä.

3.7.1. Palvelin

Tiedon tallennus ja haku toimii Apache palvelimessa. Ohjelma on toteutettu Python-kielellä. Asennusohjeet palvelimelle ovat liitteessä LIITE 1 kohdassa 1.7..

Palvelinohjelmisto toteuttaa anturin ja palvelimen välisen rajapinnan, joka on kuvattu kohdassa 3.5.3.. Koska palvelinohjelmisto toimii Apache palvelimen päällä, palvelin ei suorita mitään ajastettuja toimintoja, eikä suorita mitään toimenpiteitä tausta-ajona. Ohjelmisto on myös suunniteltu siten, että ajastetulle tai jatkuvalle toiminnalle ei ole mitään tarvetta. Mikäli kuitenkin tulevaisuudessa on tarvetta ajastetulle tai taustalla ajettavalle toiminnalle, sellainen voidaan toteuttaa erillisenä prosessina. Kaikki muuttuva tieto on kuitenkin tallennettuna tietokantaan, jota voidaan käsitellä myös useammalla sovelluksella.

Tilatieto tallennetaan siten, että jokaiselle tilatiedolle on tilan alkuaika, tilan numero ja tilan voimassaoloaika. Jos tilatiedon voimassaoloaikana päivitetään muuttumaton tila, niin tilatiedon voimassaoloaikaa päivitetään vastaavasti. Tilan muuttuessa, edellisen tilatiedon voimassaoloajaksi päivitetään uuden tilatiedon aloitusaika (kuva 11).

Palvelimelle tulevat kutsut ovat asynkronisia ja palvelinprosesseja voi olla käynnissä useampiakin. Kun edellisen tilatiedon voimassaoloaikaa muutetaan, on mahdollista syntyä tilanne, jossa kaksi tilatietoa saapuu palvelimelle lähes samanaikaisesti. Tällöin voi syntyä kilpatilanne edellisen tilatiedon päivitykseen. Kuitenkaan ei ole tarkoituksenmukaista päivittää yhden kohteen tilatietoja useammasta paikasta. Myöskään tiedonkeruuyksikön ei ole tarkoituksenmukaista lähettää uutta tilatietoa, ennen kuin edellinen tilatiedon lähetys on valmistunut. Tässä tapauksessa ei siis ole tarpeen ottaa huomioon kilpavaraustilannetta. Tilatietoja luettaessa ei haittaa, vaikka edellisen tilatiedon voimassaoloaikaa muutetaan,

ennen uuden tilatiedon lisäämistä.

Mikäli tilatiedon voimassaoloaika päättyy, niin tila katsotaan määrittelemättömäksi. Määrittelemätön tila vastaa tilannetta, jolloin tiedonkeruuyksikkö ei ole päällä lähettämässä tilaa, tai lähetyksessä tulee pidempi katkos, esimerkiksi verkkohäiriön sattuessa. Tilatieto päivittyy palvelimelle toteutuneen mukaisesti, kun tiedonkeruuyksikkö toipuu verkkohäiriöstä. Tilatiedon voimassaoloajan säännöllinen päivitys nopeuttaa ja helpottaa koneen tilan seurantaa, sekä mahdollistaa mahdolliset palvelimelta tiedonkeruuyksikölle säännöllisin väliajoin lähetettävät viestit. Tiedonkeruuyksikön ja palvelimen välillä ei ole pysyvää yhteyttä, eikä palvelimelta tiedonkeruuyksikölle voida lähettää muuten viestejä.

Laskettaessa käyttöaikasuhdetta halutulta aikaväliltä, lasketaan kunkin tilan voimassaoloajat yhteen ja jaetaan aikavälin pituudella.

Kun halutaan laskea esimerkiksi särmäriltä, montako iskua tietyllä aikavälillä on saavutettu, lasketaan vain yhteen iskutilat, jossa aloitusaika osuu kyseiselle aikavälille.

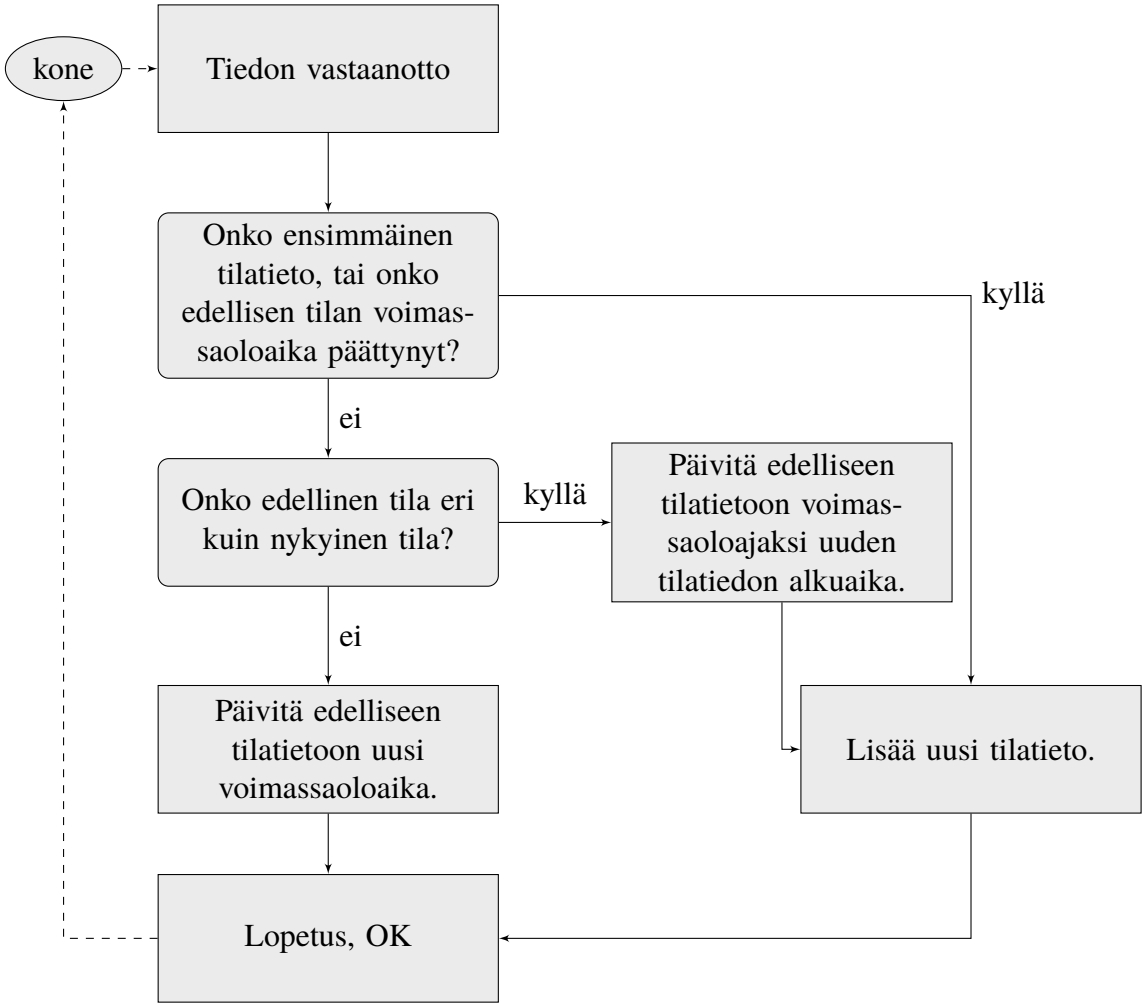
3.7.2. Tiedonkeruuyksiköt

Raspberry Pi tiedonkeruuyksiköissä on yksi ajettava ohjelma jokaista mittauskohdetta kohden. Ohjelma on toteutettu Python-kielellä. Asennusohjeet tiedonkeruuyksikölle ovat liitteessä LIITE 1 kohdassa 1.5..

Ohjelma on jaettu kahteen säikeeseen. Ensimmäinen säie huolehtii fyysisistä liitännöistä ja tiedon lukemista. Toinen säie huolehti tiedon lähettämisestä palvelimelle. Tieto kahden säikeen välillä puskuroidaan, mikäli tiedonsiirto palvelimelle keskeytyy.

3.7.3. Autentikointi

Tiedonkeruuyksiköiden autentikointi tapahtuu avaimen avulla, jolla tiedon lähetys palvelimelle sallitaan. Myös käyttöliittymän autentikointi tapahtuu avaimen avulla. Hallintaa ja



Kuva 11: Tilan päivitys tiedonkeruuyksiköltä palvelimelle. Katkoviivalla on esitetty tiedonsiirto tiedonkeruuyksikön ja palvelimen välillä.

datan visualisointia varten on oma avaimensa.

3.7.4. Käyttöliittymä

Käyttöliittymä on toteutettu selainpohjaisesti. Sivusto voi sijaita periaatteessa missä tahansa, mistä selain voi sivuston ladata. Kaikki toiminnallinen osa on toteutettu JavaScriptillä, jolla myös muuttuva tieto haetaan palvelimelta. JavaScript-ohjelma ladataan kerran palvelimelta, jonka jälkeen näkymän vaihtuessa sivustoa ei ladata palvelimelta uudelleen. Näkymän vaihto tapahtuu JavaScript-ohjelman sisäisellä reitityksellä. Aivan kuten palvelinkin, myös selaimella ajettavan JavaScript-ohjelmiston sisäinen reititys on toteutettu

soveltaen REST API:n henkeä. Selaimen osoiterivillä näkyvä osoite vastaa aina tiettyä näkymää. Osoiteriville voidaan suoraan syöttää haluttu näkymän tyyppi, valittu koneryhmä ja kone. Selaimen osoiterivillä oleva linkki toimii aina samalla tavalla, mikäli tallennettu tieto ei muutu. Kaikki tiedon haku palvelimelta tapahtuu asynkronisesti, eikä selaimen tilaa tallenneta mitenkään. Poikkeuksena on autentikointiavain, joka tallennetaan selaimen muistiin.

Autentikointiavain syötetään joko sivulla olevaan avain-kenttään tai se annetaan osoiteriville *Key* –argumentilla, joka poistuu osoiteriviltä autentikoinnin jälkeen. Autentikointiavaimen syöttö osoiteriviltä on tarkoitettu vain infonäytölle.

Käyttöliittymän rakenne on toteutettu AngularJS JavaScript kirjaston avulla. Käyttöliittymän yksi osa koostuu yhdestä html-sivusta, johon upotetaan eri toiminnallisuudet JavaScriptillä.

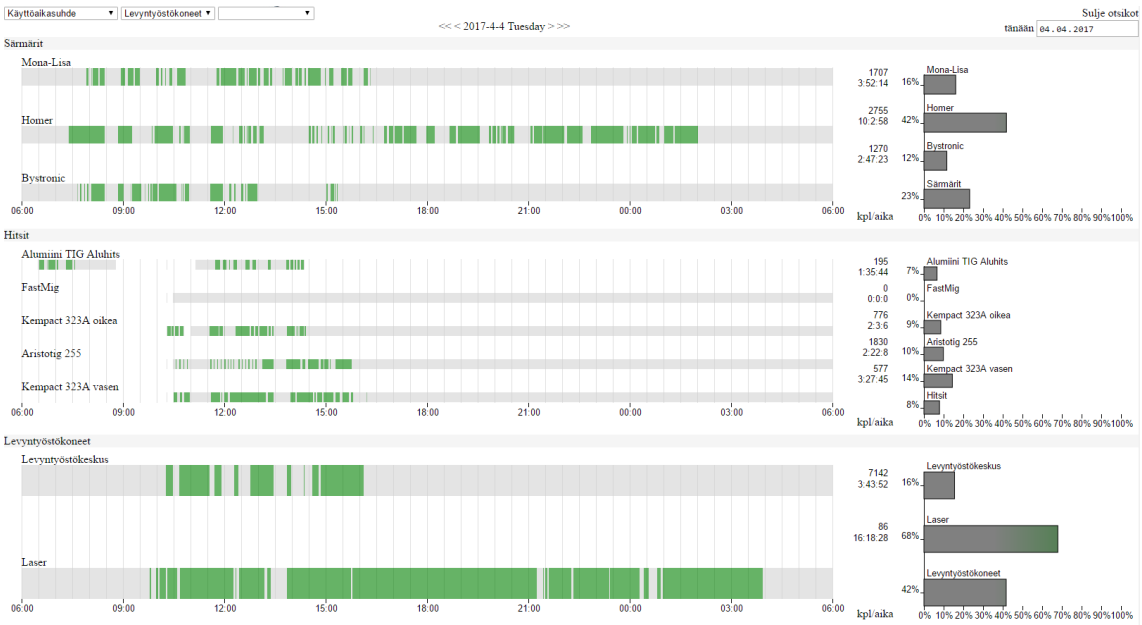
Käyttöliittymä on jaettu kahteen osaan, hallintaan ja datan visualisointiin. Hallinta-osassa voidaan mm. lisätä koneita järjestelmään.

Datan visualisointi on toteutettu D3.js JavaScript kirjaston avulla. Kuvaajat luodaan suoraan HTML-sivuun SVG-grafiikalla.

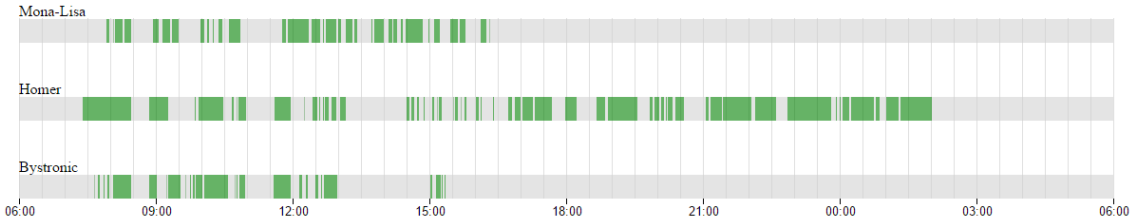
Käyttöliittymä on tarkoitettu käytettäväksi erikokoisilla näytöillä, joten se on suunniteltu responsiiviseksi, eli näytettävä tieto mukautuu käyttöliittymän kokoon. Selaimet osaavat toki rivittää tekstit näytön koon mukaan. Haastavaksi suunnittelu tulee siinä vaiheessa, kun esimerkiksi kuvaajien halutaan täyttävän näytöltä kaiken käytettävissä olevan tilan. Vielä haastavampaa on saada useita kuvaajia näytölle siten, että kaikki ovat saman kokoisia. Monet ongelmat johtuvat siitä, että selain piirtää sivua ja samalla Javascript-ohjelmat luovat uutta sisältöä sivuun. Javasript-ohjelmassa saadaan kyllä selville minkä kokoinen elementti sivusta on varattu, mutta sivun piirtämisen ollessa kesken, samankokoiseksi tarkoitetut elementit voivatkin olla eri kokoisia Javascript-ohjelman suoritusajankohdasta ja -järjestyksestä riippuen. Ongelma esiintyy lähinnä silloin, kun halutaan sovittaa kuvaaja

käyttämään sille varatun pystysuuntaisen tilan. Kun kuvaajalle annetaan vakio korkeus ongelma poistuu. Kun sivulla on useita Javascript-ohjelmalla toteutettuja elementtejä ja niiden välillä on keskinäisiä riippuvuuksia, suoritusjärjestyksellä on suuri merkitys.

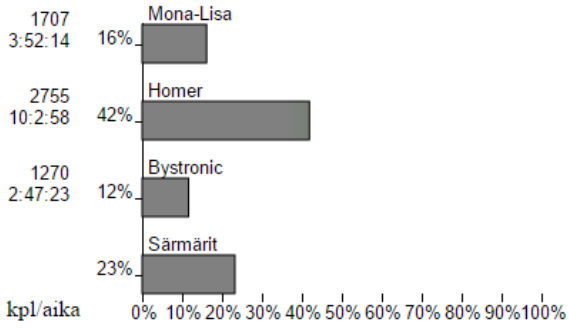
Olen myös aiemmin tehnyt selainpohjaisia käyttöliittymiä ja voin todeta, että toiminnallisuuden siirtäminen palvelimelta selaimelle helpottaa käyttöliittymän kehitystä huomattavasti. Tarvitaan vain rajapinta tiedon siirtoon palvelimen ja selaimen välillä. Lisäksi käyttöliittymän toiminta voi olla paljon nopeampaa, kun koko sivua ei tarvitse ladata uudelleen kaikille toiminnoille. Suuria tietomääriä visualisoitaessa selaimen toiminta käy kuitenkin hitaaksi, minkä takia oletuksena näytetäänkin tilanäkymä, jossa ei ole kaikkia yksityiskohtia. Tarvittaessa voidaan tarkastella tilanäkymää, jossa näkyy jokainen tilatiedon muutos sekunnin tarkkuudella. Kuvassa 12 on yleiskuva käyttöliittymästä. Tämä on oletusnäkymä, joka on myös taukotilassa sijaitsevassa infonäytössä. Kuvissa 13 ja 14 on yhden koneryhmän tilatieto ja käyttöaste yhdeltä päivältä. Tilatieto ja käyttöaste ovat myös erillisiä näkymiä, joita voidaan tarkastella koneryhmittäin tai konekohtaisesti.



Kuva 12: Yleiskuva käyttöliittymästä, Kuvassa vasemmalla näkyy laitteiden tilatieto, vihreät palkit ovat päällä-tilatietoja. Oikealla näkyy koneiden käyttöaste.



Kuva 13: Särmärien tilatieto yhdeltä päivältä, vihreät palkit ovat päällä-tilatietoja.



Kuva 14: Särmärien käyttöaste yhdeltä päivältä. Vasemmalla olevista luvuista ylempi on luettujen polkimen painallusten määrä ja alempi laskettu koneen käyttöaika (tunnit:minuutit:sekunit)

4. TULOSTEN TARKASTELU

Kaikenkaikkiaan järjestelmä on otettu hyvin vastaan asiakasyrityksessä sekä johdon että työtekijöiden osalta. Sunnitelluista ominaisuuksista jäi toteuttamatta käyttöasteen vertailu viitearvoon, vertailu aamu- ja iltavuoron välillä, sekä viikonpäivien välinen vertailu. Usean eri näytettävän aikavälin tilalle valikoitui pelkkä vuorokauden näkymä, sekä päivittäisen käyttöaikasuhteen näyttö halutulta aikaväliltä.

Vaikka kaikkia suunniteltuja ominaisuuksia ei toteutettu, on yksinkertainen järjestelmä helppokäyttöisyydessään osoittautunut hyödylliseksi. Järjestelmän käyttöönoton jälkeen laserin ja levyntyöstökeskuksen käyttöastetta on saatu nostettua niin paljon, että koneilta alkoivat loppua työt, koska koneiden kapasiteettia ei oltu myyty riittävästi. Sittemmin myyntiä on voitu lisätä. Työntekijöiden taukotilassa olevassa infonäytössä esitetty ajantasainen tilanne tuotantolaitteiden käyttöasteesta on aikaansaanut jopa pientä kilpailua työntekijöiden välillä.

4.1. Todettuja ongelmia

Ohjelmointivirheiden lisäksi käytössä ilmeni myös joitain laiteongelmia. Ongelmia tiedonkeruuyksiköiden kanssa on aiheuttanut joidenkin muistikorttien ilmeisesti heikosta laadusta johtuvat muistivirheet, jolloin käyttöjärjestelmä ei enää käynnisty. Infonäyttöä ohjaava Intel Compute Stick vaihdettiin Wi-Fi ongelmien takia pois ja tilalle otettiin Raspberry Pi 3.

4.2. Parannuskohteita

Tekniset perusratkaisut osoittautuivat pääosin toimiviksi. Varsinaisen diplomityöosuuden valmistumisen jälkeen järjestelmää on jatkokehitetty. Jatkokehityksessä AngularJS vaihtui uudempaan Angular 4 kirjastoon.

Suurilla tietomäärillä tietokannan käsittely käy hitaaksi. Haettaessa suuria tietomääriä tietokannasta Python-MySQL –rajapinta aiheuttaa hitautta. Myös tietokannan kasvaessa osa MySQL-tietokannan toiminnoista hidastuu selvästi. Osa tietokannan nopeusongelmista on ratkaistavissa ohjelmallisesti, kenties käyttämällä eri tietokantarajapintaa. Suurilla tietomäärillä MySQL:n tilalle tulisi mahdollisesti jokin HBase-pohjainen ratkaisu.

5. YHTEENVETO

Kokeilujeni perusteella en voi suositella mitään yksittäistä valmista IoT-ratkaisua tai pilvipalvelua ensisijaisena vaihtoehtona. IoT-ratkaisun voi toteuttaa monilla eri tekniikoilla. Olennaisempaa on, mitä erikoistarpeita järjestelmältä vaaditaan, sekä mitä vaatimuksia mahdollisesti olemassa oleva infrastruktuuri edellyttää. Käytetyt menetelmät voi valita tarpeen ja osaamisen mukaan. On eri asia tehdä prototyyppi, kuin laaja vuosia käytössä oleva järjestelmä. Prototyypin teossa voi käyttää jotain helppoa ja yksinkertaista. ThingSpeak esimerkiksi sopii erinomaisesti prototyyppi ja demoalustaksi.

Tässä rakennetussa järjestelmässä ei ole mitään uutta tekniikkaa, eikä mitään yksittäistä osaa mitä ei olisi jo jossain muualla tehty. Kokonaisuuden rakentaminen vaatii kuitenkin ohjelmointia. Ohjelmoinnin ja koko työn kannalta suurin työ on ollut datan visualisoinnissa ja käyttöliittymän tekemisessä.

Vaikka ohjelmointi vaatiikin paljon työtä, vaati paljon aikaa myös käytettävien tekniikoiden ja järjestelmien vertailu ja opiskelu.

Mielestäni tärkeimpiä asioita IoT-ratkaisun rakentamisessa on hallittava kokonaisuus, joka voidaan saavuttaa hyvin toteutetuilla rajapinnoilla. Järjestelmää ei kuitenkaan saada kerralla valmiiksi, joten järjestelmän eri osioita on pystyttävä päivittämään toisistaan riippumatta. Myös skaalautuvuus on otettava huomioon. Järjestelmän rakenteessa on parempi varautua nykyistä hieman suurempaan järjestelmään. Vaikka onkin hyvä varautua tuleviin lisäyksiin, liian yleiskäyttöinen järjestelmä voi kuitenkin tuoda turhaa lisäkuormaa koko järjestelmään.

Javascript-pohjainen selainkäyttöliittymä on mielestäni hyvä vaihtoehto erilliselle tietokoneessa ajettavalle sovellukselle. Sovelluksesta ei tarvita eri versioita eri alustoille ja järjestelmä saadaan toimimaan kaikilla alustoilla, joissa vain on riittävän uusi selain. Tosin näytön koko ja eri käyttötavat on otettava huomioon. Monet sovellukset ovatkin viimeaikoina siirtyneet verkossa toimiviksi palveluiksi.

Tietokannan suorituskykyyn en ole täysin tyytyväinen, vaikkakin sain sitä rakennetta muuttamalla parannettua huomattavasti, alkuperäisiin suunnitelmiin verrattuna. Selaimen suorituskyky suurilla tietomäärillä aiheutti myös ongelmia.

Tiedonkeruuyksikkönä on ollut vain Raspberry Pi 3, mutta palvelimen olen asentanut ongelmitta usealle eri alustalle.

Työstä oli posteri esillä Suomen automaatioseuran järjestämässä Automaatiopäivät22 – tapahtumassa 23.-24.3.2017 Vaasassa.

LÄHDELUETTELO

Amazon (2017). *AWS IoT - Amazon Web Services*. [Online; siteerattu 22.11.2017]. URL: http://www.chartjs.org/.

Angular (2017). *Angular*. [Online; siteerattu 14.11.2017]. URL: https://angular.io.

AngularJS (2017). *AngularJS*. [Online; siteerattu 14.11.2017]. URL: https://angularjs.org.

Blixa Morgan, Brendan T. Hill (2017). *the thing system*. [Online; siteerattu 23.11.2017]. URL: http://thethingsystem.com/.

Bostock, Mike (2017). *D3.js*. [Online; siteerattu 14.11.2017]. URL: https://d3js.org/.

Chart.js (2017). *Chart.js*. [Online; siteerattu 14.11.2017]. URL: http://www.chartjs.org/.

Crisan, Calin (2017). *motionEyeOS*. [Online; siteerattu 31.3.2017]. URL: https://github.com/ccrisan/motioneyeos/wiki/FAQ.

Drill (2016). *Apache Drill*. [Online; siteerattu 19.12.2016]. URL: http://drill.apache.org.

Element14 (2016). *Raspberry Pi 3 Model B Technical Specifications*. [Online; siteerattu 27.11.2016]. URL: https://www.element14.com/community/docs/DOC-80899/l/raspberry-pi-3-model-b-technical-specifications.

Fielding, Roy Thomas (2000). *Representational State Transfer (REST)*. [Online; siteerattu 18.1.2017]. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

Fredrich, Todd (2016). *REST API Tutorial*. [Online; siteerattu 19.12.2016]. URL: http://www.restapitutorial.com/lessons/whatisrest.html.

GE (2017). *Predix | Industrial Cloud-Based Platform (PaaS) | GE Digital*. [Online; siteerattu 22.11.2017]. URL: https://www.ge.com/digital/predix.

Google (2017a). *Choosing a Storage Option | Google Cloud Platform*. [Online; siteerattu 23.11.2017]. URL: https://cloud.google.com/storage-options/.

Google (2017b). *Google Cloud Platform*. [Online; siteerattu 7.2.2017]. URL: https://cloud.google.com/.

Google (2017c). *IoT-Ticket, the Office Suite for Internet of things*. [Online; siteerattu 22.11.2017]. URL: https://cloud.google.com/solutions/iot/.

gRPC (2016). *gRPC*. [Online; siteerattu 19.12.2016]. URL: http://www.grpc.io.

Hadoop (2016). *Apache Hadoop*. [Online; siteerattu 19.12.2016]. URL: http://hadoop.apache.org.

HBase (2016). *Apache HBase*. [Online; siteerattu 19.12.2016]. URL: http://hbase.apache.org.

Hive (2016). *Apache Hive*. [Online; siteerattu 19.12.2016]. URL: http://hive.apache.org.

IBM (2017a). *IBM Bluemix*. [Online; siteerattu 7.2.2017]. URL: https://console.ng.bluemix.net/.

IBM (2017b). *IBM Watson Internet of Things (IoT)*. [Online; siteerattu 22.11.2017]. URL: http://www.ibm.com/internet-of-things/.

KaaIoT (2017). *Kaa Open-Source IoT Platform 2017*. [Online; siteerattu 22.11.2017]. URL: https://www.kaaproject.org/.

Mathworks (2017). *IoT Analytics - ThingSpeak Internet of Things*. [Online; siteerattu 23.11.2017]. URL: https://thingspeak.com/.

Microsoft (2017a). *Microsoft Azure*. [Online; siteerattu 24.11.2017]. URL: https://docs.microsoft.com/fi-fi/azure/iot-hub/.

Microsoft (2017b). *Microsoft Azure*. [Online; siteerattu 24.11.2017]. URL: https://developer.microsoft.com/en-us/windows/iot.

Microsoft (2017c). *Microsoft Azure*. [Online; siteerattu 7.2.2017]. URL: https://azure.microsoft.com.

Microsoft (2017d). *Windows Subsystem for Linux Documentation*. [Online; siteerattu 10.11.2017]. URL: https://msdn.microsoft.com/en-us/commandline/wsl/about.

Pig (2016). *Apache Pig*. [Online; siteerattu 19.12.2016]. URL: http://pig.apache.org.

Raspberry (2016). *Raspberry Pi 3 model b*. [Online; siteerattu 15.11.2016]. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

Raspberry (2017). *GPIO: Models A+, B+, Raspberry Pi 2 B and Raspberry Pi 3 B*. [Online; siteerattu 3.2.2017]. URL: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/README.md.

Spark (2016). *Apache Spark*. [Online; siteerattu 19.12.2016]. URL: http://spark.apache.org.

Storm (2016). *Apache Storm*. [Online; siteerattu 19.12.2016]. URL: http://storm.apache.org.

Thinger.io (2017). *Thinger.io - Open Source IoT Platform*. [Online; siteerattu 23.11.2017]. URL: https://thinger.io/.

Wapice (2017). *AWS IoT - Amazon Web Services*. [Online; siteerattu 25.11.2017]. URL: https://www.iot-ticket.com/.

LIITE 1 Asennusohjeet

1.1. Linux Raspbian Jessie Lite –asennus

Raspbian on ladattavissa osoitteessa:

`https://downloads.raspberrypi.org/raspbian_lite_latest`

Levykuva voidaan kirjoittaa sd-kortille käyttämällä *Win32DiskImager* nimistä Windows-ohjelmaa. Ohjelma on ladattavissa osoitteessa:

`https://sourceforge.net/projects/win32diskimager/`

Samalla ohjelmalla voidaan lukea sd-kortista levykuva, jolloin asetusten tekemisen jälkeen sd-kortti voidaan helposti monistaa käyttövalmiiksi muita tiedonkeruuyksiköitä varten.

Ennen Wi-Fin asennusta, on käyttöjärjestelmään kirjauduttava paikallisesti. Raspbian oletus käyttäjätunnus on `pi` ja salasana on `raspberry`. Konfiguroi Raspberry Pi suorittamalla komento:

```
sudo raspi-config
```

Aseta käyttöjärjestelmä käyttämään koko SD korttia, valitsemalla '`1 Expand Filesystem`'. Aseta maa-asetukset, aikavyöhyke ja Wi-Fi maa-asetus. Aktivoi myös SSH-palvelin kohdasta '`7 Advanced Options`'.

Wi-Fi asennus, kun IP-osoite haetaan automaattisesti DHCP-palvelimelta:

`https://thepihut.com/blogs/raspberry-pi-tutorials/83502916-how-to-setup-wifi-on-raspbian-jessie-lite`

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Muokkaa tiedostoon oikea SSID ja salasana.

```
country=FI
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
ssid="Pi_AP"
psk="Raspberry"
}
```

Käynnistä uudelleen:

```
sudo reboot
```

Tämän jälkeen järjestelmään voidaan kirjautua verkon yli SSH:lla.

Asennetaan käyttöjärjestelmän päivitykset:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

1.2. Linux Raspbian Jessie with PIXEL –asennus

Käyttöjärjestelmä käynnistyy suoraan graafiseen käyttöliittymään, jolloin Wi-Fi asennus voidaan hoitaa helposti oikean yläkulman kuvakkeista. Kun Wi-Fi on asennettu, voidaan käyttöjärjestelmän päivitykset tehdä, kuten Raspbian Jessie Lite –asennuksessakin.

Graafista etähallintaa varten katso kohta 1.8.2..

1.3. Wi-Fi testaus

Wi-Fi signaalin voimakkuutta voidaan seurata ohjelmalla nimeltä *wavemon*. Asennus ja ohjelman käynnistys:

```
sudo apt-get install wavemon
wavemon
```

Myös komennolla `iwconfig`, saadaan nopeasti hetkellinen signaalin voimakkuus.

Kokeiltaessa, kahden Raspberry Pi 3:n välillä yhteys katkeaa aiemmin, kuin Raspberry Pi 3:n ja kannettavan tietokoneen välillä, joten Raspberry Pi 3:n sisäinen Wi-Fi antenni ei ole paras mahdollinen, mutta on kuitenkin käyttökelpoinen.

1.4. Levykuva

Kuten aiemmin mainitsin, sd-kortista voidaan lukea levykuva. Levykuva voidaan avata Linux-käyttöjärjestelmässä, jolloin siihen voidaan suoraan tehdä muutoksia. Ensin täytyy selvittää Linux-osion sijainti levykuvassa.

```
$ sudo fdisk -l  RaspberryClientImage.img

Disk RaspberryClientImage.img: 14.4 GiB, 15489564672 bytes,\
    30253056 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb4cd76bb

Device                     Boot  Start      End  Sectors
Size Id Type
RaspberryClientImage.img1        8192   137215   129024
63M  c W95 FAT32 (LBA)
RaspberryClientImage.img2
137216 30253055 30115840 14.4G 83 Linux
```

Tässä Linux-osion aloitussektori on 137216, joka kerrotaan 512:lla, jolloin saadaan osion aloituskohta (512 * 137216 = 70254592). Nyt osio voidaan liittää.

```
sudo mount -o offset=70254592 RaspberryClientImage.img \
    /mnt/loop
```

Liitettyyn osioon voidaan suoraan tehdä tarvittavat asetukset, jolloin tiedonkeruuyksikkö on valmis käyttöön sd-kortin asennuksen jälkeen.

Levykuvan liitämisen sijaan, tiedostoja voidaan myös muokata sen jälkeen, kun levykuva

on tallennettu sd-kortille.

Windowsille on mahdollista asentaa Ubuntu BASH konsoli, joka toimii kuten Ubuntu Linux Windowsin konsoli-ikkunassa. Tällä hetkellä (6.2.2017) ei kuitenkaan vielä ole mahdollista liittää Linux-osioita levykuvasta tai sd-kortilta.

Linux-osioiden käsittely Windowsilla onnistuu kuitenkin asentamalla Windowsille Linux virtuaalikone.

1.5. Tiedonkeruuyksikön ohjelmiston asennus

Kopioi tiedonkeruusovellus Raspberry Pi kortille *IoTClientV1Alpha.py*.

Ohjelman käynnistys komentoriviltä:

```
python IoTClientV1Alpha.py -i 11 -o 16 -a 1 \
    -s 192.168.0.1:443 -d 1 -k "kirjoitus"
```

Ohjelma tarvitsee käynnistettäessä seuraavat argumentit:

- *i* GPIO tuloliittimen numero.
- *o* GPIO lähtöliittimen numero tilalediä varten.
- *a* Onko tulo aktiivinen alhaalla (0), vai ylhäällä (1).
- *s* Palvelimen osoite ja portti.
- *d* Anturin numero.
- *k* Kirjoitusavain.

Kokeile toiminta ja lopeta sovellus <*Ctrl-C*>

Aseta ohjelma käynnistymään automaattisesti.

```
sudo nano /etc/rc.local
```

Lisää seuraava rivi */etc/rc.local* tiedostoon.

```
...
python IoTClientV1Alpha.py -i 11 -o 16 -a 1 \
    -s 192.168.0.1:443 -d 1 -k "kirjoitus" &

exit 0
```

Mikäli yhdellä kortilla luetaan useampaa tilatietoa, käynnistetään kutakin tilatietoa varten oma sovellus, joka lisätään myös */etc/rc.local* tiedostoon.

1.6. MotionEyeOs asennus

Käyttöjärjestelmän levykuva löytyy osoitteesta:

`https://github.com/ccrisan/motioneyeos/releases`

Levykuvan asennus tapahtuu samalla tavalla kuin Raspbian levykuvan asennus. Kun levykuva on asennettu sd-kortille ja motionEyeOS käynnistetään ensimmäisen kerran, käyttöjärjestelmä tekee alustuksia muutaman minuutin. Alustuksien jälkeen motionEyeOS on valmis käyttöön ja ethernetin kautta päästään käyttöjärjestelmään kiinni. MotionEyeOS kuuntelee porttia 80, josta päästään selaimella muuttamaan asetuksia. Oletuksena myös SSH-palvelin on päällä. Käyttäjätunnuksia on kaksi, kameravalvontaan *user* ilman salasanaa ja hallintaa varten *admin* myös ilman salasanaa. Kun selaimella kirjaudutaan motionEyeOS palvelimelle *admin* tunnuksilla, voidaan ottaa myös WiFi käyttöön, muuttaa kameran asetuksia ja monia muita asetuksia.

Tilatiedon lähetystä varten asetetaan *Motion Notifications* osioon *Run A Command* päälle ja asetetaan komennoksi:

```
curl --connect-timeout 2 -m 5 -k -X PUT \
"https://192.168.0.1/iot/v1_alpha/sensors/11?
    Key=kirjoitus&status=1"
```

Rajapinta on kuvattu tarkemmin kohdassa 3.5.3..

1.7. Palvelimen ohjelmistot

Normaalin Debian tai Raspbian –Linux asennuksen jälkeen tarvitaan vain Apache ja MySQL asennus, sekä kopioida palvelimen ohjelmisto ja käyttöliittymäsivusto. Asennus toimii suoraan myös Windowsin Bash shell:ssä.

1.7.1. Apache ja MySQL asennus

Asennus Debian-Linuxiin, Python 2:

```
sudo apt-get install mysql-server python-mysql.connector
sudo apt-get install apache2
sudo apt-get install python-pip libapache2-mod-wsgi
sudo pip install pymysql webapp2 WebOb Paste
sudo a2enmod cgi
sudo a2enmod ssl
sudo a2ensite default-ssl
sudo mkdir /etc/apache2/ssl
sudo openssl req -x509 -nodes -days 36500 -newkey rsa:2048 \
    -keyout /etc/apache2/ssl/apache.key \
    -out /etc/apache2/ssl/apache.crt
sudo chmod 600 /etc/apache2/ssl/*
```

Muokkaa Apache asetustiedostoja:

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
sudo nano /etc/apache2/sites-enabled/default-ssl.conf
```

Lisää molempiin rivi:

```
WSGIScriptAlias /iot/v1_alpha \
/var/www/Server/IoT/ServerIoT/IoTServerV1Alpha.py
```

Ja vaihda `DocumentRoot`:

```
DocumentRoot /var/www/Server/www
```

Uudelleenkäynnistä Apache:

```
sudo /etc/init.d/apache2 restart
```

Kopioi ohjelmistot hakemistoon `/var/www`.

Alusta tietokanta:

```
mysql -u root < /var/www/Server/IoT/ServerIoT/InitDatabase.sql
```

1.8. Infonäytön asennus

1.8.1. Raspberry Pi:n käyttö infonäytön ohjauksessa

Raspberry Pi:stä täytyy kytkeä näytönsäästäjä pois päältä ja saada Chromium-selain käynnistymään käynnistettäessä. Näytönsäästäjän poistamiseksi muokkaa tiedostoa:

```
sudo nano /etc/kbd/config
```

Muuta seuraavat asetukset:

```
BLANK_TIME=0
BLANK_DPMS=off
POWERDOWN_TIME=0
```

Muokkaa vielä tiedostoa:

```
sudo nano /etc/lightdm/lightdm.conf
```

Ja lisää `[SeatDefaults]` –osioon seuraava rivi:

```
[SeatDefaults]
xserver-command=X -s 0 -dpms
```

Selaimen käynnistämiseksi muokkaa tiedostoa:

```
nano /home/pi/.config/lxsession/LXDE-pi/autostart
```

Ja lisää seuraava rivi loppuun:

```
@/usr/bin/chromium-browser -kiosk -fullscreen --incognito \
http://192.168.0.1/
    RiimaIoT/visualization.html#!/status-view?Key=luku
```

Hiiren kursorin piilottaminen:

```
sudo apt-get install unclutter
sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

Lisää seuraava rivi loppuun:

```
unclutter -display :0 -noevents -grab
```

Hiiren kursori pysyy piilossa käynnistyksen jälkeen, mutta hiirtä liikutettaessa kursori tulee kuitenkin näkyviin.

1.8.2. Etäyhteys

Infonäytön Raspberry Pi –tietokoneeseen voidaan ottaa normaali SSH-yhteys, mutta tarvittaessa voidaan myös ottaa haltuun infonäytön näkymä. Ensin tulee sallia VNC-yhteys Raspberry Pi:n asetuksista (`raspi-config`). VNC-yhteys voidaan ottaa suoraan VNC-asiakasohjelmalla, tai se voidaan tunneloida SSH:n kautta. VNC-etäyhteyden näkymä on sama kuin infonäytössä.

Ohje VNC-yhteyden tunnelointiin SSH:lla: `https://crl.ucsd.edu/handbook/vnc/` Ohjeesta poiketen, VNC-palvelin on jo käynnissä ja näytön numero on 0.

1.8.3. Windows-tietokoneen käyttö infonäytön ohjauksessa

Infonäyttöä varten selainikkunan täytyi vielä saada käynnistymään tietokoneen käynnistyessä. Chrome-selaimen pikakuvake kopioidaan hakemistoon:

```
C:\Users\User\AppData\Roaming\Microsoft\Windows\Start Menu\
    Programs\Startup
```

Pikakuvakkeen ominaisuuksista lisätään vielä käynnistystiedoston perään seuraavat parametrit:

```
"http://192.168.0.1/
    RiimaIoT/visualization.html#!/status-view?Key=luku"
    --kiosk --fullscreen --incognito
```

Kun Chrome-selainta ei suljeta kunnolla, mm. virran katkeamisen takia, selain kysyy aina, palautetaanko edellinen istunto. Infonäytössä tämän kaltainen toiminto on epätoivottavaa, koska tietokoneeseen ei ole tarkoitus liittää hiirtä eikä näppäimistöä. Parametrilla *–incognito* selain ei kysy edellisen istunnon palauttamista, selain kuitenkin käynnistyy *incognito*-tilaan, eikä tallenna mitään tietoja. Selaimen kuitenkin täytyy muistaa autentikointiavain. Autentikointia varten sivusto hyväksyy osoiteriville *Key*-parametrin, jolla saadaan autentikointiavain välitettyä. Tietoturvan kannalta tämä ei ole hyvä ratkaisu. Kuitenkin *Key*-parametri häipyy osoiteriviltä välittömästi autentikoinnin jälkeen ja selainikkuna käynnistetään kokonäytölle, jolloin osoiterivi ei ole edes näkyvillä.

1.9. Reitittimen asennus Linux-ympäristöön

Ohjeet soveltuvilta osin:

`https://learn.adafruit.com/`
`setting-up-a-raspberry-pi-as-a-wifi-access-point/install-software`

Asenna tarvittavat ohjelmat

```
sudo apt-get install hostapd isc-dhcp-server
sudo apt-get install iptables-persistent
```

Sammuta wlan

```
sudo ifdown wlan0
```

Aseta DHCP-laitteet

```
sudo nano /etc/default/isc-dhcp-server
```

Aseta DHCP:lle vain `wlan0`.

```
INTERFACES="wlan0"
```

Aseta DHCP-asetukset

```
sudo nano /etc/dhcp/dhcpd.conf
```

Muuta rivit

```
# option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server
# for the local network,
# the authoritative directive should be uncommented.
#authoritative;
```

Seuraavan kaltaisiksi

```
# option definitions common to all supported networks...
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server
# for the local network,
# the authoritative directive should be uncommented.
authoritative;
```

Lisää vielä loppuun rivit

```
subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.100 192.168.0.199;
option broadcast-address 192.168.0.255;
option routers 192.168.0.1;
default-lease-time 600;
max-lease-time 7200;
#option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Aseta staattinen ip-osoite

```
sudo nano /etc/network/interfaces
```

Muokkaa wlan0 asetuksia seuraavanlaiseksi

```
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.0.1
netmask 255.255.255.0
#iface wlan0 inet manual
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Luo uusi tiedosto

```
sudo nano /etc/hostapd/hostapd.conf
```

Lisää seuraavat rivit

```
interface=wlan0
ssid=Pi_AP
country_code=FI
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1
```

Määritellään NAT. Muokkaa tiedostoa:

```
sudo nano /etc/default/hostapd
```

Lisää rivi.

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Muokkaa tiedostoa:

```
sudo nano /etc/sysctl.conf
```

poista kommentointi

```
net.ipv4.ip_forward=1
```

Kirjoita seuraavat komennot:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state \
--state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE
sudo iptables -A FORWARD -i usb0 -o wlan0 -m state \
--state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o usb0 -j ACCEPT
sudo sh -c "iptables-save > /etc/iptables/rules.v4"
```

Uudelleenkäynnistä

```
sudo reboot
```

LIITE 2 REST API

Eri tyyppiset rajapintakutsut erotetaan HTTP verbeillä POST, GET, PUT, PATCH, ja DELETE. Eri verbit on selitetty seuraavassa osoitteessa:

`http://www.restapitutorial.com/lessons/httpmethods.html`

Rajapinta palauttaa HTTP tilakoodin (*status code*), esim. `200 OK` onnistuneessa rajapintakutsussa. Eri koodit on määritelty seuraavassa osoitteessa:

`http://www.restapitutorial.com/httpstatuscodes.html`

REST API –rajapinta toimii asynkronisesti. Rajapinnan yhteyksiä tai tiloja ei tallenneta mitenkään. Rajapinta on siis 'tilaton'.

Erilaiset rajapintakutsut:

- PATCH `/groups/<gid>` Muokkaa ryhmän numero <*gid*> nimeä. Argumentit:
  - `name`: Ryhmän nimi (pakollinen).

  Toiminto sallittu avaimilla:
  - `AdminKey`
- POST `/groups` Luo uuden koneryhmän. Argumentit:
  - `name`: Ryhmän nimi (pakollinen).

  Palauttaa tekstimuodossa luodun ryhmän numeron.

  Toiminto sallittu avaimilla:
  - `AdminKey`
- GET `/groups` Palauttaa JSON-muodossa listan ryhmistä. Avaimella *AdminKey* listaa kaikki ryhmät. Avaimella *ReadKey* listaa vain ryhmät, mihin on lukuoikeus.

```
[
{"gid": 1, "name": "ryhmä 1"},
{"gid": 2, "name": "ryhmä 2"}
]
```

gid on ryhmän numero ja *name* on ryhmän nimi.

Toiminto sallittu avaimilla:

- `AdminKey`
- `ReadKey`

- GET `/groups/<gid>` Kuten edellinen, mutta palauttaa vain ryhmän numero <*gid*> tiedot.

```
{"gid": 1, "name", "ryhmä 1"}
```

Toiminto sallittu avaimilla:

- `AdminKey`
- `ReadKey`

- POST `/groups/<gid>/sensors` Luo uuden koneen ryhmään <*gid*>. Argumentit:
  - `name`: Koneen nimi (pakollinen)

Palauttaa tekstimuodossa luodun koneen numeron.

Toiminto sallittu avaimilla:

- `AdminKey`

- GET `/groups/<gid>/sensors` Lista ryhmän <*gid*> koneista. Palauttaa JSON-muodossa listan ryhmän koneista.

```
[
{"sid": 1, "name": "kone 1"},
{"sid": 2, "name": "kone 2"}
]
```

jossa *sid* on koneen numero ja *name* on koneen nimi.

Toiminto sallittu avaimilla:

- `AdminKey`
- `ReadKey`

- GET `/groups/<gid>/sensors/<sid>` Ryhmän numero <*gid*> koneen numero <*sid*> tilatiedot. Argumentit:
  - `start`: Aikavälin aloitus UTC ISO 8601 –muodossa. esim. `2017-01-16T08:18:00.0Z` viimeinen `Z` ja sekunnin desimaalit ovat vapaaehtoisia.
  - `stop`: Aikavälin lopetus UTC ISO 8601 –muodossa.
  - `filter=1`: Jos tämä argumentti on annettu, tilatiedoista suodatetaan pois lyhyet tilatiedot ”1”-tilojen välistä.

Palauttaa TSV-muodossa koneen numero <*sid*> tilatiedot aikaväliltä.

```
date    status  active
2017-01-16T14:11:20.448372Z 0 2017-01-16T14:11:20.571789Z
2017-01-16T14:11:20.571789Z 1 2017-01-16T14:12:20.571789Z
```

jossa *date* on tilatiedon alkamisaika, *status* on numeerinen tilatieto ja *active* on tilatiedon voimassaoloaika. Aika on UTC ISO 8601 –muodossa.

Toiminto sallittu avaimilla:

- `ReadKey`

- GET `/groups/<gid>/sensors/<sid>` Ryhmän numero <*gid*> koneen numero <*sid*> ”1”-tilojen lukumäärä. Argumentit:
  - `start`: Aikavälin aloitus UTC ISO 8601 –muodossa. esim. `2017-01-16T08:18:00.0Z` viimeinen `Z` ja sekunnin desimaalit ovat vapaaehtoisia.
  - `stop`: Aikavälin lopetus UTC ISO 8601 –muodossa.
  - `count=1`: Pakollinen, lasketaan tilojen ”1” lukumäärä. Voidaan antaa myös muun laskettavan tilan numero.

  Palauttaa tilojen lukumäärän JSON-muodossa.

  ```
  {"count": 123}
  ```

  Toiminto sallittu avaimilla:

  - `ReadKey`

- GET `/groups/<gid>/sensors/<sid>` Palauttaa ryhmän numero <*gid*> ja koneen numero <*sid*> nimen JSON-muodossa.

  ```
  {"sid": 1, "name": "kone 1", "groupName": "ryhmä 1"}
  ```

  jossa *sid* on koneen numero, *name* on koneen nimi ja *groupName* on ryhmän nimi.

  Toiminto sallittu avaimilla:

  - `ReadKey`

- PATCH `/groups/<gid>/sensors/<sid>` Siirrä kone numero <*sid*> ryhmään numero <*gid*>.

  Toiminto sallittu avaimilla:

  - `AdminKey`

- GET `/groups/<gid>/sensors` Kaikkien ryhmän <*gid*> koneiden tilojen suhteet halutulla aikavälillä. Argumentit:

  - `start`: Aikavälin aloitus UTC ISO 8601 –muodossa. esim. `2017-01-16T08:18:00.0Z` viimeinen `Z` ja sekunnin desimaalit ovat vapaaehtoisia.
  - `stop`: Aikavälin lopetus UTC ISO 8601 –muodossa.
  - `usage=0`: Tyyppi (pakollinen)

  Palauttaa JSON-muodossa kaikkien koneiden tilojen suhteet halutulla aikavälillä.

```
[
{
"kone 1":
{
"1": 0.07007375679168666,
" ": 0.9236281837417151,
"0": 0.006298059466598278}},
{
"kone 2":
{
"1": 0.07218262150858996,
" ": 0.9236303241155247,
"0": 0.0041870543758852985
}
}
]
```

  Jokaisesta koneesta tulee ensin koneen nimi, ja sen jälkeen koneen tilatiedot ja tilan suhde aikavälillä. Tila ” ” tarkoittaa tuntematonta tilaa, eli koneelta ei aikavälin alussa ole ollut mittaustietoja.

  Toiminto sallittu avaimilla:

  - `ReadKey`

- PUT `/sensors/<sid>` Koneen numero <*sid*> tilatiedon lähetys. Argumentit:
  - `status`: Tilatieto kokonaislukuna.
  - `time`: Tilatiedon aikaleima UTC ISO 8601 –muodossa. esim. `2017-01-16T08:18:00.0Z` viimeinen `Z` ja sekunnin desimaalit ovat vapaaehtoisia. Mikäli tilatiedon aikaleima puuttuu, nykyinen aika annetaan tilatiedon aikaleimaksi.

  Palauttaa JSON-muodossa tietoja koneesta.

```
{
"timeout": 60
}
```

  - `timeout: nn`. Tilatiedon voimassaoloaika sekunneissa, kun tila pysyy muut-

tumattomana.

Toiminto sallittu avaimilla:

- `WriteKey`

- DELETE `/sensors/<sid>` Koneen numero <*sid*> poisto. Poistaa myös kaikki tilatiedot kyseiseltä koneelta.

  Toiminto sallittu avaimilla:

  - `AdminKey`

- PATCH `/groups/<gid>/keys/<kid>` Muokkaa ryhmän numero <*gid*> avaimen numero <*kid*> tietoja. Argumentit:

  - `KeyName`: Avaimen nimi (pakollinen)
  - `Value`: Avaimen arvo (pakollinen)

  Toiminto sallittu avaimilla:

  - `AdminKey`

- DELETE `/groups/<gid>/keys/<kid>` Avaimen numero <*kid*> poisto ryhmästä numero <*gid*>.

  Toiminto sallittu avaimilla:

  - `AdminKey`

- POST `/groups/<gid>/keys` Luo uuden avaimen ryhmään numero <*gid*>. Argumentit:

  - `KeyName`: Avaimen nimi (pakollinen)
  - `Value`: Avaimen arvo (pakollinen)

  Toiminto sallittu avaimilla:

  - `AdminKey`

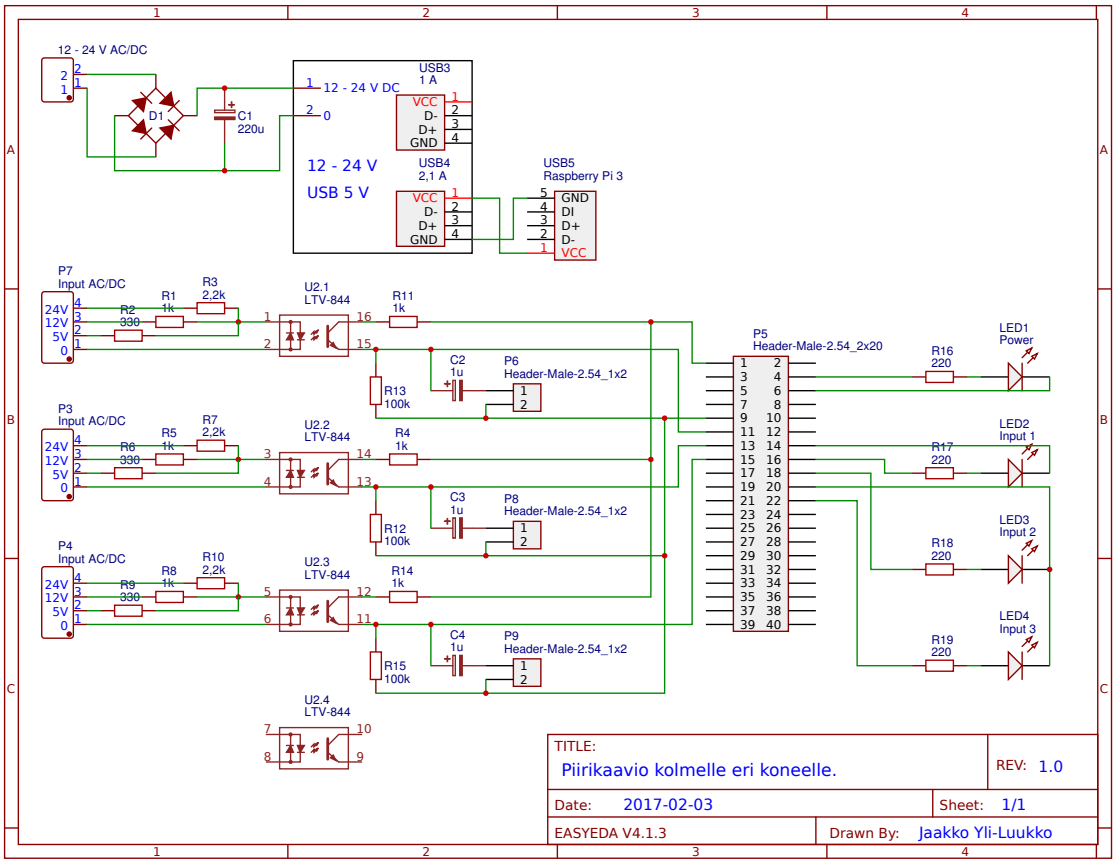
- GET `/groups/<gid>/keys` Palautta JSON-muodossa kaikki ryhmän numero <*gid*> avaimet.

```
[
{
"key": "AdminKey",
"value": "admin",
},
{
"key": "ReadKey",
"value": "luku",
}
]
```

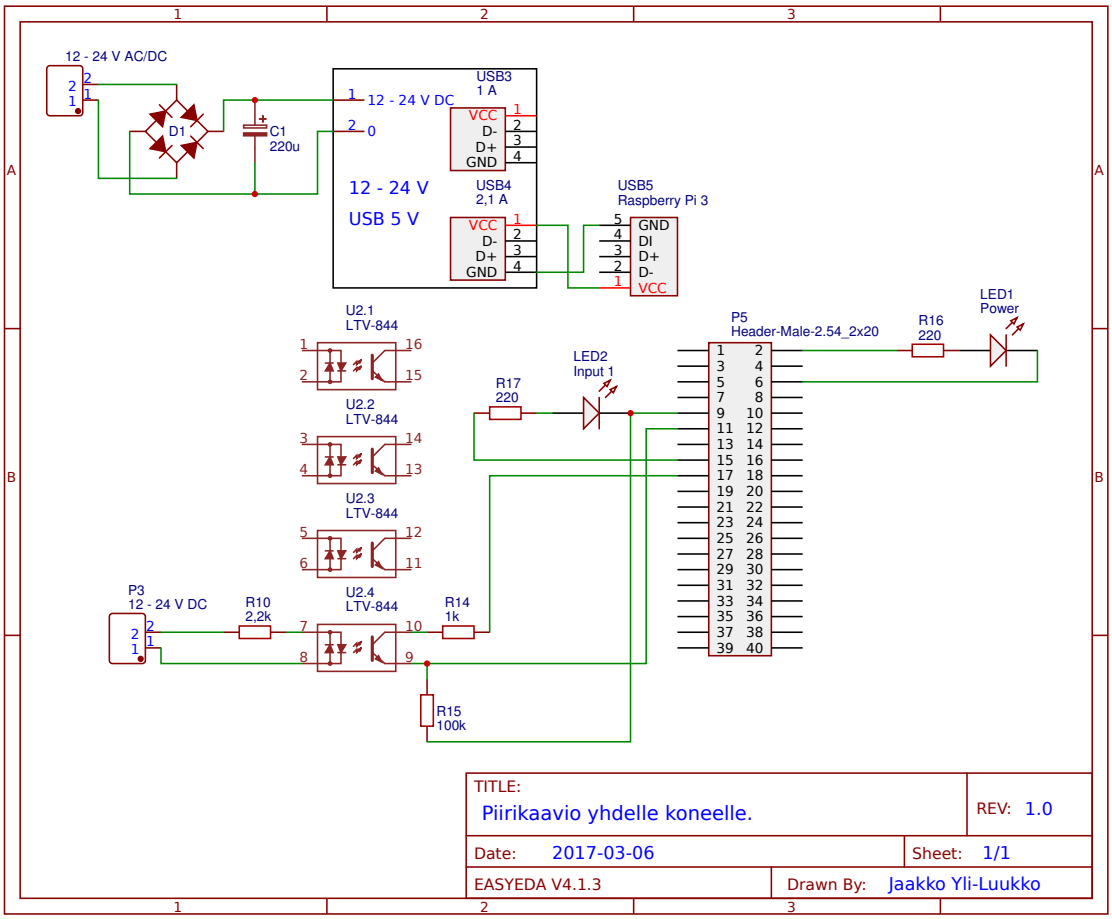
Toiminto sallittu avaimilla:

- `AdminKey`

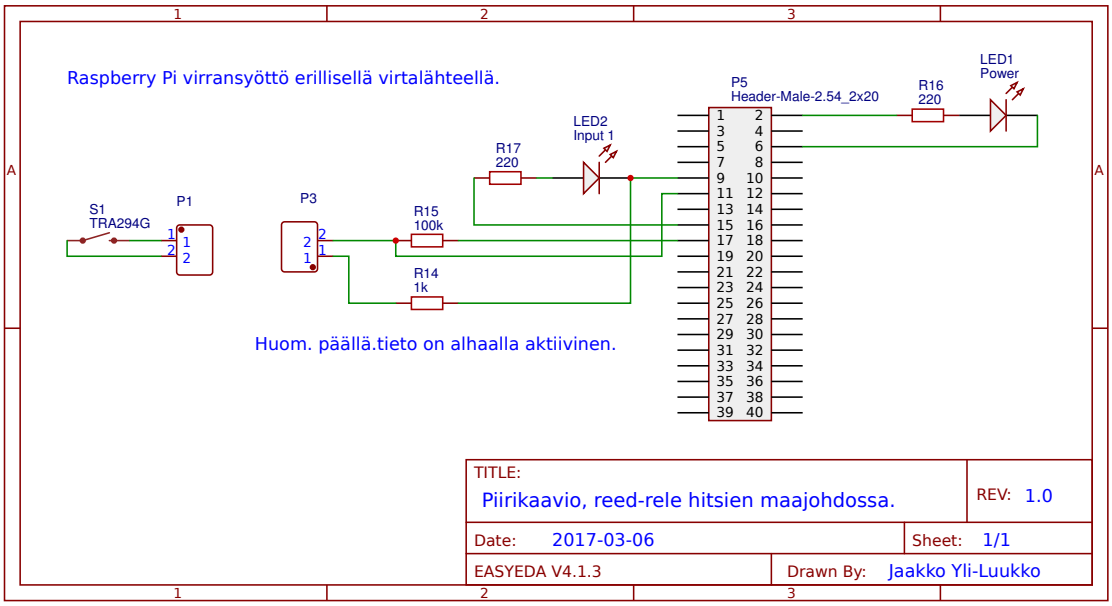
LIITE 3 Tiedonkeruuyksikkö



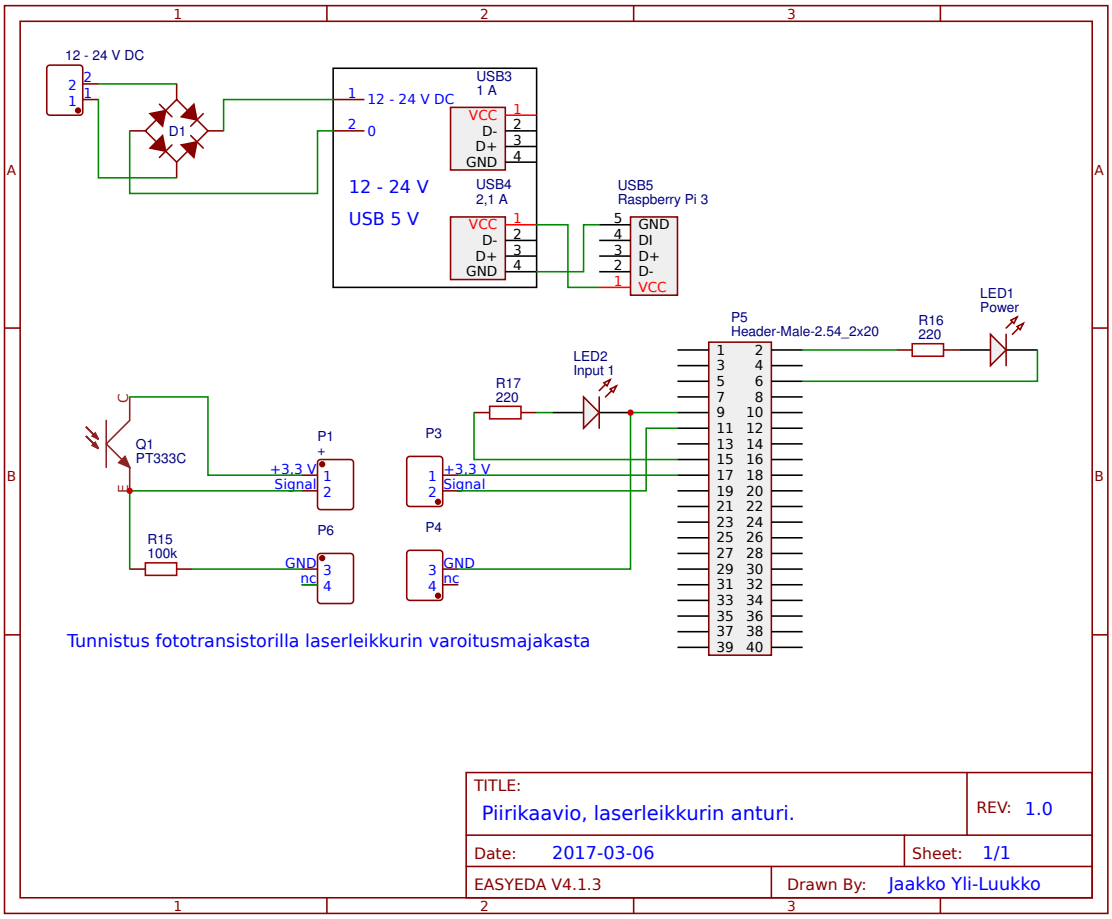
Kuva LIITE 3.1: Piirikaavio kolmelle eri koneelle.



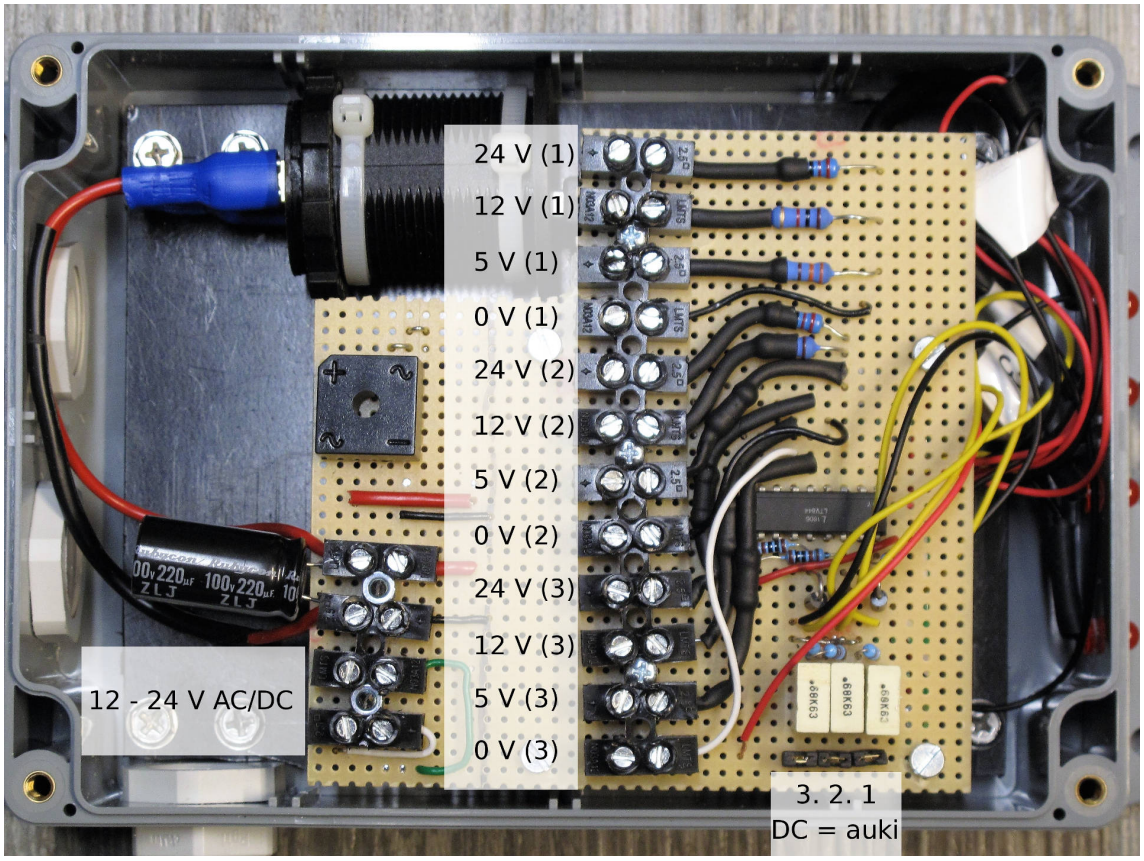
Kuva LIITE 3.2: Piirikaavio yhdelle koneelle.



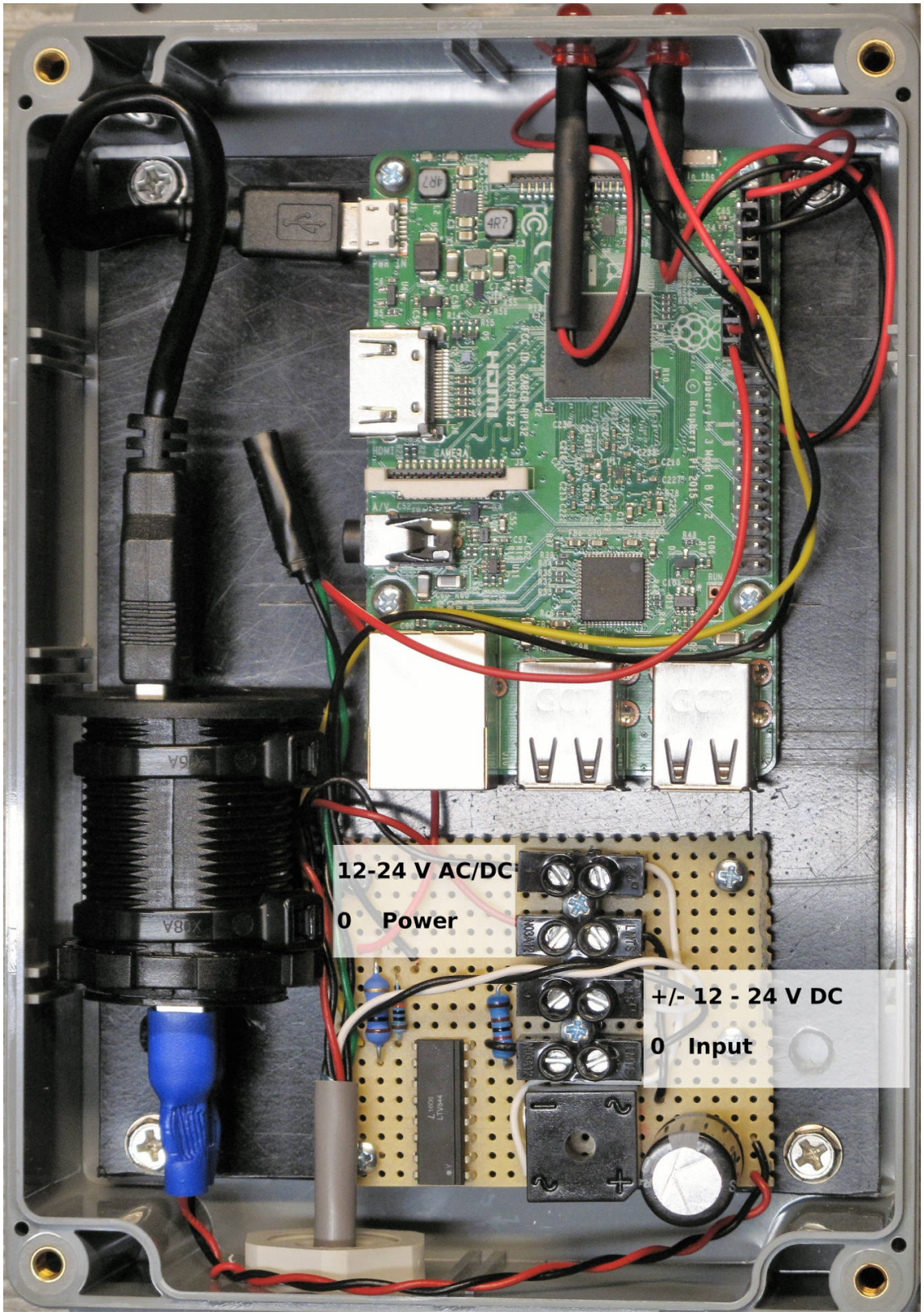
Kuva LIITE 3.3: Piirikaavio, hitsauskoneen käyttötieto maajohdosta reed-releellä.



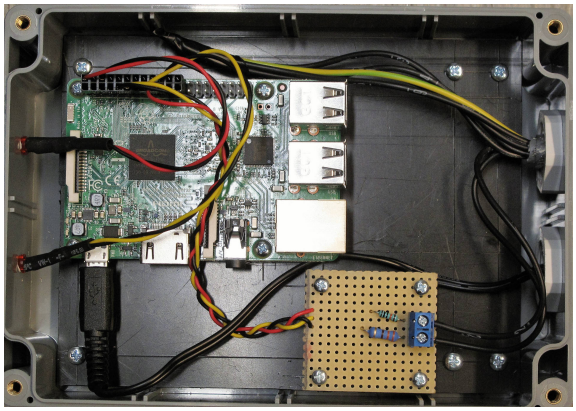
Kuva LIITE 3.4: Piirikaavio, laserleikkurin tunnistus varoitusmajakasta fototransistorilla.



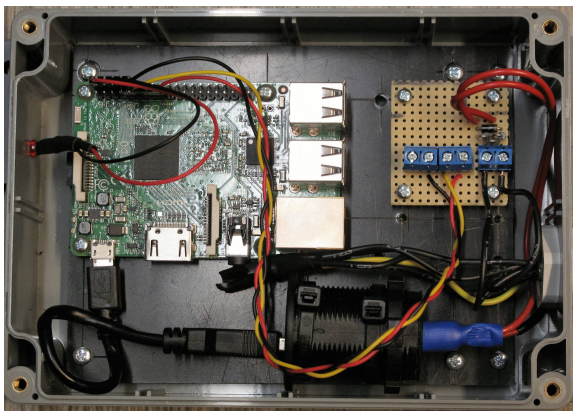
Kuva LIITE 3.5: Johdotus, tiedonkeruuyksikkö kolmelle eri koneelle.



Kuva LIITE 3.6: Johdotus, tiedonkeruuyksikkö yhdelle koneelle.



Kuva LIITE 3.7: Johdotus, tiedonkeruuyksikkö hitsauskoneille.



Kuva LIITE 3.8: Johdotus, tiedonkeruuyksikkö laserleikkurille.

Kuva LIITE 3.9: Johdotus, fototransistori kotelossa laserleikkurin varoitusmajakkaan.

LIITE 4 ThingSpeak asennus

Raspberry Pi demon asennus:

`https://thingspeak.com/channels/new`

`https://github.com/nicholaswilde/thingspeak-raspberry-pi`

```
sudo apt-get install python-psutil python-pip
pip install thingspeak
cd /opt
sudo git clone \
https://github.com/nicholaswilde/thingspeak-raspberry-pi.git
cd thingspeak-raspberry-pi
sudo chmod +x thingspeak_cpu_loop.py
sudo nano thingspeak_cpu_loop.py
```

Muokkaa oikea 'Write API Key', ja kokeile toiminta.

```
python thingspeak_cpu_loop.py
```

Aseta ohjelma käynnistymään automaattisesti.

```
sudo nano /etc/rc.local
```

Lisää seuraava rivi ennen `exit 0` –komentoa `/etc/rc.local` –tiedostoon.

```
...
python /opt/thingspeak-raspberry-pi/thingspeak_cpu_loop.py &

exit 0
```

Uudelleenkäynnistä Raspberry Pi:

```
sudo reboot
```

LIITE 5 Google Cloud Platform

Taulukko LIITE 5.1: Google Cloud Platform palvelut lyhyesti.

API Manager	Palveluiden käytön seuranta.
Billing	Google Cloud Platform palvelun laskutuksen hallinta.
Cloud Launcher	Nopeasti käyttöönotettavia palveluita. *Cloud Launcher* sisältää valmiita ratkaisuja, joista suuri osa voidaan ottaa käyttöön *Compute Engine* virtuaalikoneina. Ratkaisuja ovat mm. erilaiset tietokannat, käyttöjärjestelmät, sisällönhallintajärjestelmiä (CMS), tuotannonohjausjärjestelmiä (ERP), asiakkuudenhallintaratkaisuja (CRM) ja kehitystyökaluja.
Support	Käytön tuki.
IAM & Admin	Käyttöoikeuksien hallinta.
App Engine	Sovelluspalveluiden hallinta.
Compute Engine	Virtuaalikoneiden hallinta.
Container Engine	Sovellusten ja palveluiden paketointijärjestelmä.
Networking	Verkkorajapintojen hallinta.
Bigtable	Googlen NoSQL tietokanta. Skaalautuu suureksi, ei kustannustehokas pieneen käyttöön, kuten tuli kokeiltaessa huomattua.
SQL	Googlen ylläpitämä MySQL tietokanta.
Datastore	Googlen NoSQL tietokanta, jossa käytetään SQL:n kaltaista GQL-kieltä. Soveltuu hyvin pieneen käyttöön, mutta skaalautuu myös suuremmaksi.

Storage	Jäsentämättömän tiedon tallennus, ”data bucket”.
Monitoring	Palvelujen valvonta ja raportointityökalu.
Debug	App Engine debuggeri.
Trace	Analysointityökalu.
Logging	Lokien tarkastelutyökalu.
Error Reporting	Virheiden raportointityökalu.
Development	Kehitystyökalut ja App Engine lähdekoodit.
Deployment Manager	Palveluiden käyttöönotto, hallinta ja poisto.
Cloud Test Lab	Android sovellusten testaustyökalu.
Endpoints	Omien rajapintojen hallintatyökalu.
BigQuery	Big data tiedon analysointityökalu SQL-hakujen avulla. (Samantyyppinen kuin Apache Drill.)
Pub/Sub	Reaaliaikainen viestinvälitysjärjestelmä sovellusten välille.
Dataproc	Apache Hadoop, Sparc, Pig ja Hive pilvipalvelu.
Dataflow	Big data sovellusten hallinnointityökalu.
Machine Learning	Työkaluja koneoppimiseen (beta).
Genomics	Geneettisen datan analysointityökalu.

LIITE 6 Lähdekoodit

Raspberry Pi – ThingSpeak luku ja tiedonsiirto

ThingSpeak2.py

```
#!/usr/bin/python

import Queue
import threading
import time
import datetime
import httplib
import urllib
import signal
import sys
from datetime import datetime
from time import localtime, strftime

resenddelay = 10    #
low_active = 1        # GPIO pin active on low
state_pin = 7
led_pin = 11
#apiKey = 'LI5UJWAWGQ5ZISOK'
#apiKey = 'WWUMSRKFIZ9KSQQV'
apiKey = 'EEHA8KF2Z50KGCA9'
#server    = "api.thingspeak.com:80"
#server    = "192.168.1.99:3000"
server = "192.168.0.1:3000"

send_interval = 60     # seconds
gpiodelay = 0.1
queuesize = 10    # In production 10000

exitflag = 0

class GpioThread(threading.Thread):
    '''.'''
    def __init__(self, threadID, name, q):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.q = q
    def run(self):
        print("Starting " + self.name)
        read_gpio(self.name, self.q)
        print("Exiting " + self.name)

class HttpThread (threading.Thread):
    '''.'''
    def __init__(self, threadID, name, q):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.q = q
    def run(self):
        print("Starting " + self.name)
        send_http(self.name, self.q)
        print("Exiting " + self.name)

def signal_term_handler(signal, frame):
    '''.'''
    exitflag = 1

signal.signal(signal.SIGTERM, signal_term_handler)

def read_gpio(threadname, q):
    '''.'''
    try:
```

```
64          import RPi.GPIO as GPIO
65      except RuntimeError:
66          print("RPi.GPIO␣error!")
67
68      GPIO.setmode(GPIO.BOARD)
69
70      output_list = [led_pin]
71      GPIO.setup(output_list, GPIO.OUT)
72      input_list = [state_pin]
73      GPIO.setup(input_list, GPIO.IN, pull_up_down=GPIO.PUD_UP)
74      print("Initialized")
75
76      old_state = -1
77      old_time = 0
78
79      while not exitflag:
80          new_time = int(time.time())
81          state = GPIO.input(state_pin)^low_active
82          GPIO.output(led_pin, state)
83          if old_state != state or (old_time + send_interval) < new_time:
84              old_time = new_time
85              old_state = state
86              t = datetime.fromtimestamp(new_time).isoformat()
87              # If queue is full, discard latest items
88              if q.full():
89                  q.get()
90              q.put((t, state))
91              print("%s␣processing␣%s" % (t, state))
92          time.sleep(gpiodelay)
93      # Clean up
94      GPIO.cleanup()
95
96  def send_http(threadname, q):
97      '''.'''
98      while not exitflag:
99          if not workqueue.empty():
100             data = q.get()
101         else:
102             time.sleep(1)
103             continue
104
105         print(data[0])
106         print(data[1])
107
108         resend = True
109         while resend and not exitflag:
110             try:
111                 params = urllib.urlencode({'created_at':data[0], 'field1': data[1], '
                        key':apiKey})
112                 headers = {"Content-type": "application/x-www-form-urlencoded", "
                        Accept": "text/plain"}
113                 conn = httplib.HTTPConnection(server)
114                 print("try␣connecting")
115                 conn.request("POST", "/update", params, headers)
116                 print("connected")
117                 response = conn.getresponse()
118                 print(response.status, response.reason, response.msg)
119                 print(response.read())
120                 conn.close()
121                 resend = False
122             except:
123                 print("connection␣failed")
124                 time.sleep(resenddelay)
125
126 workqueue = Queue.Queue(queuesize)
127 threads = []
128 threadid = 1
129
130 # Create new threads
131 threadGPIO = GpioThread(1, "GPIO", workqueue)
132 threadGPIO.start()
133 threads.append(threadGPIO)
134 threadHTTP = HttpThread(2, "HTTP", workqueue)
135 threadHTTP.start()
136 threads.append(threadGPIO)
137
138 # Main thread waits
```

```
try:
    while not exitflag:
        time.sleep(1)
except KeyboardInterrupt:
    exitflag = 1

# Wait for all threads to complete
for t in threads:
    t.join()
print("Exiting Main Thread")

#### End of program ####
```

LIITE 7 Tietokanta

Tietokannan luonti:

InitDatabase.sql

```
create user 'iot'@'localhost' identified by 'iot';
create database iot;
grant all on iot.* to 'iot';
use iot;

create table settings (
        SetId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        KeyName CHAR(20) NOT NULL,
        Value CHAR(64),
        GroupId INT
    );

insert into settings (KeyName, Value)
    VALUES ('AdminKey', 'admin');

create table groups (
        GroupId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        GroupName CHAR(20) CHARACTER SET utf8 COLLATE utf8_bin
    );

create table sensors (
        SensorId INT NOT NULL AUTO_INCREMENT,
        GroupId INT NOT NULL,
        SensorName CHAR(20) CHARACTER SET utf8 COLLATE utf8_bin,
        SensorType INT NOT NULL,
        Timeout INT NOT NULL DEFAULT 60,
        PRIMARY KEY(SensorId, GroupId)
    );

/*  Following tables are created dynamically as needed.
create table msgStatus_nn (
    MsgId INT NOT NULL AUTO_INCREMENT,
    MsgTime DOUBLE NOT NULL,
    MsgActiveTime DOUBLE NOT NULL,
    MsgStatus INT NOT NULL
    PRIMARY KEY(MsgId, MsgTime)
);
*/
```