TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MAX ANGELMA

OHJELMOITAVILTA LOGIIKOILTA KERÄTYN TIEDON VISUALISOINTI

Kandidaatintyö

Tarkastaja: Jani Jokinen

TIIVISTELMÄ

MAX ANGELMA: Ohjelmoitavilta logiikoilta kerätyn tiedon visualisointi
Visualization of Data Collected from Programmable Logic Controllers
Tampereen teknillinen yliopisto
Kandidaatintyö, 20 sivua
Maaliskuu 2017
Automaatiotekniikan koulutusohjelma
Pääaine: Factory Automation
Tarkastaja: Jani Jokinen

Avainsanat: ohjelmoitava logiikka, teollinen internet, visualisointi, Snap7, Highcharts

Ensimmäinen askel kohti teollista internetin tuomia mahdollisuuksia on tiedon kerääminen tehtaan lattiatason laitteilta. Kun tietty laite yhdistetään internettiin, sen kuntoa ja käyttöä voidaan seurata reaaliajassa. Keräämällä tietoa pidemmältä aikaväliltä, voidaan kasvattaa ymmärrystä, miten laitetta tosiasiassa käytetään. Tätä lisääntynyttä tietämystä voidaan hyödyntää myös esimerkiksi optimoinnissa, tuotekehityksessä, kunnossapidossa, markkinoinnissa ja myynnissä.

Kerätylle tiedolle voidaan toki tehdä erilaisia tilastollisia analyyseja, mutta visualisointi on havainnoinnin työkalu ja sitä käyttämällä datasta pystytään usein oikeasti näkemään trendejä, korrelaatioita ja epätavallisuuksia. Onnistuneen visualisoinnin rakentaminen ei kuitenkaan ole välttämättä yksinkertaista, mutta työssä esitellyllä Andy Kirkin metodologialla voi päästä hyviin lopputuloksiin.

Tämän kandidaatintyön käytännön osuudessa tutustuttiin teollisuuslogiikalta kerätyn datan visualisointiin. Tieto tallennettiin taulukkona tekstitiedostoon, josta luotiin graafisia kuvaajia selainkäyttöliittymään. Sovellukselle ei määritelty erityisiä tavoitteita, mutta sitä voidaan pitää silti onnistuneena. Kehittämisessä käytettiin avoimen lähdekoodin kirjastoja, ja ohjelmakoodit julkaistiin jatkokehitystä varten MIT-lisenssillä.

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

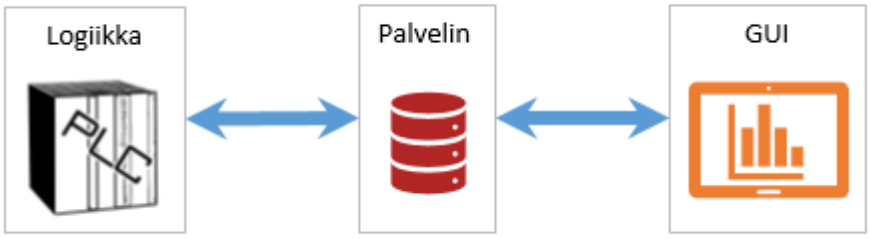
CSV	engl. Comma-Separated Values, taulukkomuotoinen tekstitiedosto-muoto
DHCP	engl. Dynamic Host Configuration Protocol, IP-osoitteita jakava verkkoprotokolla
GUI	engl. Graphical User Interface, graafinen käyttöliittymä
FAST-Lab	engl. Factory Automation Systems and Technologies Laboratory, TTY:n systeemitekniikan laitoksen tutkimuslaboratorio
MIT-lisenssi	engl. Massachusetts Institute of Technology, avoimen lähdekoodin lisenssi
PLC	engl. Programmable Logic Controller, ohjelmoitava logiikka
S7-300	Siemensin S7-tuoteperheeseen kuuluva ohjelmoitava logiikka
TIA Portal	engl. Totally Integrated Automation, Siemensin ohjelmointityökalu
TTY	Tampereen teknillinen yliopisto

1. JOHDANTO

Ohjelmoitavia logiikoita käytetään laajasti teollisuudessa erilaisten laitteiden ohjauksessa. Tiedon kerääminen teollisuuslaitteilta etäyhteyksien kautta eivät ole mikään uusi asia, mutta viimeistään nyt teollinen internet -ilmiön myötä se alkaa kasvattaa merkitystään. Tässä kandidaatintyössä tutkitaan tiedon keräämistä vanhemmalta Siemensin teollisuuslogiikalta ja tallennetun tiedon visualisointia.

Työn tarkoituksena on selvittää, miksi tietoa kannattaa kerätä ja miten datajoukosta rakennetaan onnistunut visualisointi. Lisäksi visualisoinnin suhteen pohditaan, mitä lisäarvoa käyttäjälle voidaan tuoda: pelkkä tallennettu tieto itsessään kun harvoin on hyödyllistä. Painotus työssä on kuitenkin käytännön sovelluksessa, joka esitellään luvussa neljä.

Käytännön sovelluksessa kerätään tietoa TTY:n FAST-Laboratorion Festolinjan teollisuuslogiikalta. Kuvassa 1 on esitetty karkeasti ja korkealla tasolla tiedon kulkua eri komponenttien välillä. Palvelin tallentaa ohjelmoitavalta logiikalta (Programmable Logic Controller) halutut tiedot CSV-taulukkoon (engl. Comma Separated Values), josta ne sitten visualisoidaan graafisessa käyttöliittymässä (engl. Graphical User Interface).



Kuva 1. *Periaatekuva käytännön sovelluksesta*

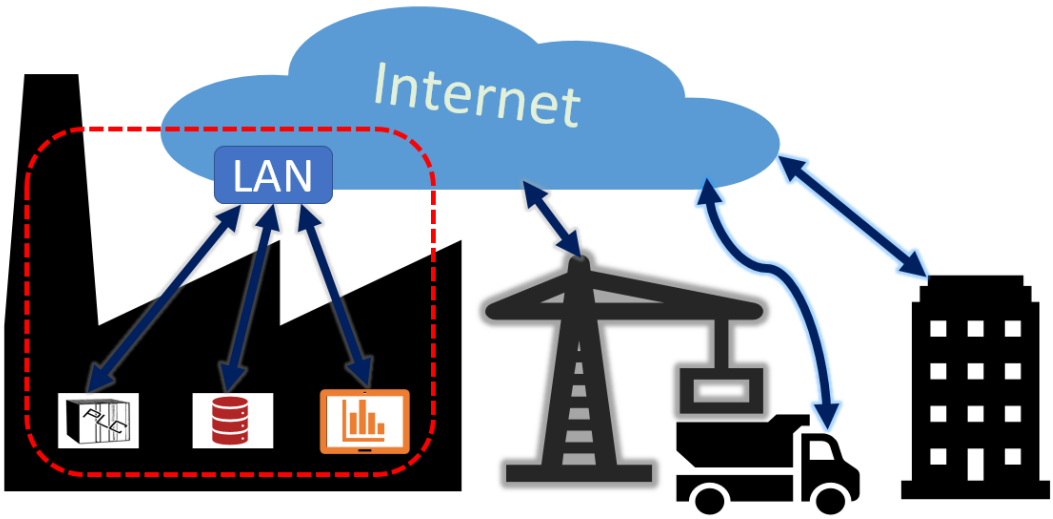
2. TYÖN TARKOITUS

Tässä luvussa pohditaan, miksi työtä lähdettiin toteuttamaan, motivaatiota tiedon keräämiseen ja avoimen lähdekoodin tarjoamia mahdollisuuksia. Teollinen internet on kasvava trendi, mutta pystytäänkö tietoa keräämään myös vanhemmilta teollisuuslaitteilta?

2.1 Teollinen internet

Industrial Internet Consortiumin mukaan nykypäivän teollinen vallankumous pohjautuu esineiden internetiin (engl. Internet of Things). Nämä esineet ovat sensoreita, ohjaimia, laitteita ja sulautettuja järjestelmiä, jotka kommunikoivat toistensa kanssa reaaliajassa. *Smart Factories: a Symphony of the Industrial Internet in Action* -esitteen mukaan ensimmäinen askel älykkääseen valmistukseen on tiedon kerääminen tehtaan lattiatason laitteilta. Esite myös mainitsee jälkikäteisasennukset (engl. retrofit) vanhalle konekannalle. (www.iiconsortium.org 2016)

Keskeisenä ajatuksena on siis kytkeä työkalut, laitteet, koneet, rakennukset, ajoneuvot ynnä muut internettiin. Tämän työn kannalta olennaisin osa teollista internettia on kuvassa 2 rajattuna punaisella katkoviivalla, jonka sisällä on ohjelmoitava logiikka, tallennusmedia (palvelin) ja käyttöliittymä visualisointineen. Jokainen näistä komponenteista voisi toki olla ”suoraan” yhdistettynä internettiin, mutta samaan lähiverkkoon (engl. Local Area Network) yhdistettynä vältytään porttiohjausten ja palomuurien konfiguroinnilta.



Kuva 2: *Tämän työn rooli teollisen internetin kontekstissa*

(Porter & Heppelmann 2014) ryhmittelee internettiin yhdistettyjen esineiden ominaisuudet neljään ryhmään: monitorointi, ohjaus, optimointi ja autonomia. Jokainen näistä rakentuu edellisen päälle, joten tiedon keräämisen rooli on siten tärkeä. Monitoroinnilla voidaan valvoa tuotteen kuntoa ja käyttöä, sekä mahdollistaa hälytykset ja ilmoitukset. Tietoa voidaan käyttää muun muassa käyttötapojen seuraamiseen, jotta voidaan ymmärtää paremmin, miten tuotetta oikeasti käytetään. Näin voidaan esimerkiksi poistaa tuotteista ominaisuuksia, joita ei käytetä (engl. reduce over-engineering). Lisäksi tiedosta on hyötyä markkinasegmentoinnissa, palvelujen myynnissä jälkikäteen (engl. after-sales service), takuuasioissa ja uusissa myyntimahdollisuuksissa. (Porter & Heppelmann 2014)

2.2 Siemens S7

FAST-Laboratorion Festolinjalla käytetty Siemensin ohjelmoitava teollisuuslogiikka S7-300 on julkaistu vuonna 1995. Siemens lupaa sen kuitenkin olevan saatavilla ainakin vuoteen 2020 asti ja tarjota varaosia vielä kymmenen vuotta sen jälkeenkin (w3.siemens.com). Lisäksi Siemensilla oli vuonna 2013 yli 30% markkinaosuus PLC-markkinoilla. (Electrical Engineering Blog). Voidaan siis olettaa, että kyseistä logiikkaa käytetään nykyisin ja tulevaisuudessakin varsin laajasti teollisuudessa.

Siemens S7-300 -logiikan pystyy liittämään Ethernetiin CP 343-1 -moduulin avulla. Ethernettiin liitettynä se tarjoaa integroidun palvelimen, jolla pystyy tarkastelemaan rajatusti logiikan tilatietoja. Sille ei kuitenkaan ole tarjolla varsinaista rajapintaa tiedon keräämistä varten. Käytännön osuus -luvussa esittelen työkaluja, joilla S7-300 -logiikan muistiosotteita pääsee monitoroimaan. Samat työkalut soveltuvat myös muiden S7-tuoteperheen logiikoiden monitorointiin.

2.3 Avoin lähdekoodi

Avoimella lähdekoodilla tarkoitetaan ohjelmistojen kehitysmenetelmää, jossa käyttäjä voi vapaasti tutustua lähdekoodiin. Täten ohjelmistoja tai kirjastoja voi muokata tarpeidensa mukaisiksi. Kehittäjäyhteisöt voivat olla yksittäisiä ihmisiä, kuten Snap7:n kehittäjä (Nardella 2016) tai yrityksiä, kuten Highchartsin kehittänyt norjalainen Highsoft. Periaatteisiin kuuluu yleensä myös vapaus käyttää, kopioida ja levittää niin alkuperäistä kuin itse muokattua versiota ohjelmasta. Lähdekoodin julkaisu ei kuitenkaan tee ohjelmistosta avoimen lähdekoodin edustajaa, vaan käyttöä säädellään lisenssein. Myös fyysiset laitteet voivat hyödyntää avointa lähdekoodia (engl. Open Source Harware), esimerkkinä Arduino-mikrokontrollerialusta.

Avoimen lähdekoodin kirjastot osoittautuivat hyödyllisiksi tässä työssä. Lisäksi tässä työssä kehitetty sovellus on julkaistu MIT-lisenssillä (engl. Massachusetts Institute of Technology), joten kenen tahansa mahdollista jatkokehittää sitä. Käyttämistäni avoimen lähdekoodin kirjastoista ja niiden lisensseistä on kerrottu lyhyesti luvussa neljä.

3. VISUALISOINNIN RAKENTAMINEN

Nykypäivänä dataa kerätään yhä enenevissä määrin. Tämän tiedon esittämisessä ymmärrettävästi on omat haasteensa, mutta tätä haastetta voidaan helpottaa suunnittelemalla visualisointi huolella. Tässä luvussa pohditaan visualisoinnin tärkeyttä ja käydään läpi vaatimuksia ja tekniikoita visualisointien tekoa varten. Lähteenä on käytetty muun muassa Andy Kirkin kirjaa ”Data Visualization: a successful design process” (2012), josta esitellään metodologia onnistuneen visualisoinnin rakentamiseen alaluvussa 3.1.

Tiedon visualisointi ei ole eksaktia tiedettä, jossa ongelmaan olisi yksi oikea vastaus tai ratkaisu. Visualisointia pidetäänkin eräänlaisena käsityötaitona, joka vaatii harjoitusta, aikaa ja kärsivällisyyttä. Visualisoinnin takana onkin kognitiotiedettä, matematiikkaa, statistiikkaa, graafista suunnittelua, kartografiaa ja tietojenkäsittelytiedettä. (Kirk 2012)

3.1 Andy Kirkin metodologia ja työvaiheet

Kirk esittelee kirjassaan yleiskäyttöisen ja teknologianeutraalin tavan visualisointien luomiseen painottaen konseptointia, perustelua ja päätöksentekoa. Vaikka hän ei pidäkään metodologiaansa mullistavana, hän uskoo perustavanlaatuiseen lähestymistapaansa, joka on auttanut häntä kehittymään omassa työssään. (Kirk 2012, s. 20)

Alussa selvitetään, mikä projektin tarkoitus on, miksi projektia tehdään, mitkä ovat laajuus ja konteksti, kuinka paljon luovuutta on mahdollista käyttää ja mitä ideoita on ennestään. Projektit syntyvät yleensä kahdella tavalla: sinua on joko pyydetty tekemään tai olet päättänyt itse toteuttaa jotain. Ero voi tuntua selvältä, mutta luovuuden kannalta tapaukset ovat hyvin erilaisia. (Kirk 2012, s. 30)

Kuvassa 3 Kirk on listannut erilaisia toivottuja vaikutuksia (engl. intended effect) visualisoinneille. Tunnistamalla toivotun vaikutuksen tulee selvittäneeksi, mitä on tarkoitus saavuttaa ja miten. Tärkeää on myös ottaa kohdeyleisö huomioon. (Kirk 2012, s. 32) Loppukäyttäjä tulee asettaa etusijalle ja miettiä mitä hän oikeasti tarvitsee. Tässä voidaan käyttää apuna erilaisia skenaarioita ja käyttötapauksia. (Degeler 2015)

Kuva 3. *Visualisoinnin mahdolliset vaikutukset (Kirk 2012, s. 32)*

Seuraavaksi tulee tunnistaa tärkeimmät ja oleellisimmat tarinat datasta. Suunnittelijan tulee ottaa vastuuta lopputuloksesta, jotta kohdeyleisö osaa varmasti tulkita visualisointia. Lisäksi tässä vaiheessa kerätään, valmistellaan ja tutustutaan datajoukkoon. (Kirk 2012, s.56).

Kirk esittelee myös merkittävimmät ratkaisut, jotka visualisoinnin eri tasoilla täytyy tehdä. Hän kertoo myös taustoja päätöksille ja miten niitä voi järkeistää perustuen projektin kontekstiin. Tarkoituksena on auttaa suunnittelijaa ymmärtämään haasteita, jotka liittyvät datan esittämiseen. (Kirk 2012)

Viimeisessä vaiheessa Kirk keskittyy itse visualisoinnin toteutukseen (engl. execution) ja antaa vinkkejä viimeistelyyn ja julkaisuun. Hän myös kannustaa jatkamaan harjoittelua ja listaa verkkosivuja, joilla voi tutustua aiheeseen syvemmin. (Kirk 2012, s. 176) Lopputulosta ei välttämättä kannata koristella esimerkiksi 3D-efekteillä, sillä ne todennäköisesti tekevät visualisoinnista vain vaikeammin ymmärrettävän. Tiedon suuntauksen kanssa tulee olla tarkkana ja erityisesti tulee välttää esittämästä epäolennaisia korrelaatioita tai vertailuja. (Degeler 2015)

3.2 Esimerkkejä

Tässä alaluvussa esitellään graafisien esityksien tärkeyttä esimerkkien avulla. Toisaalta visualisoinnilla voidaan myös tehdä eräänlaista taidetta, tai piilottaa tietoa.

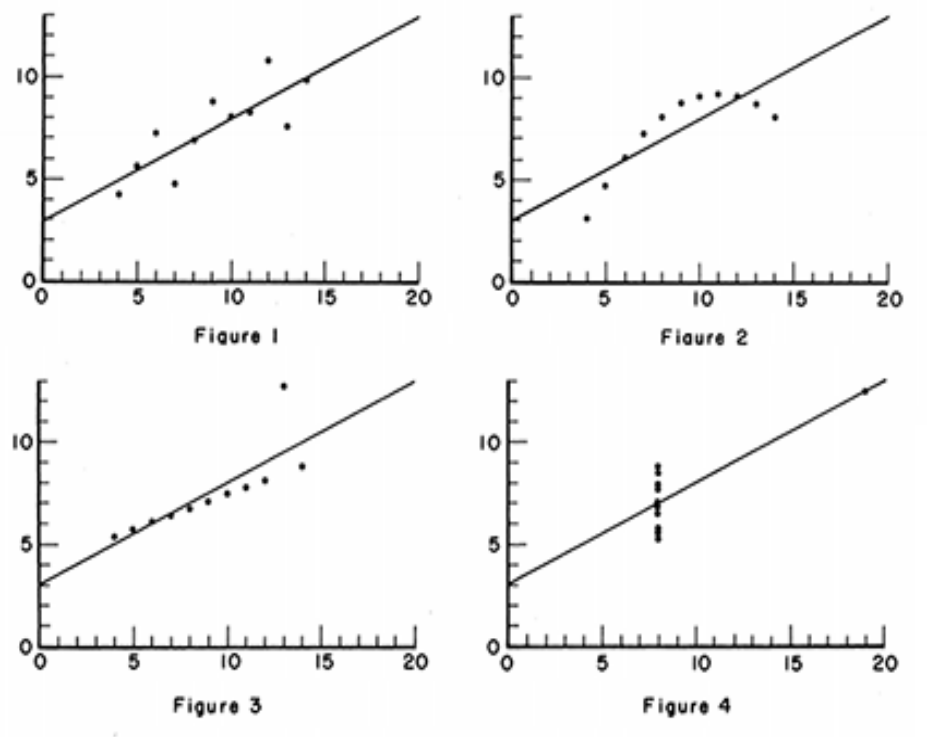
3.2.1 Francis Anscomben kvartetti

Francis Anscomben kvartetti (1973) on tunnettu esimerkki visualisoinnin hyödyistä. Siinä on neljä arvojoukkoa, jotka kaikki ovat tilastollisesti hyvin samankaltaisia. Nämä neljä arvojoukkoa on esitetty taulukossa 1.

Taulukko 1. *Anscomben kvartetin arvojoukot*

I		II		III		IV	
X	Y	X	Y	X	Y	X	Y
10.00	8.04	10.00	9.14	10.00	7.46	8.00	6.58
8.00	6.95	8.00	8.14	8.00	6.77	8.00	5.76
13.00	7.58	13.00	8.74	13.00	12.74	8.00	7.71
9.00	8.81	9.00	8.77	9.00	7.11	8.00	8.84
11.00	8.33	11.00	9.26	11.00	7.81	8.00	8.47
14.00	9.96	14.00	8.10	14.00	8.84	8.00	7.04
6.00	7.24	6.00	6.13	6.00	6.08	8.00	5.25
4.00	4.26	4.00	3.10	4.00	5.39	19.00	12.50
12.00	10.84	12.00	9.13	12.00	8.15	8.00	5.56
7.00	4.82	7.00	7.26	7.00	6.42	8.00	7.91
5.00	5.68	5.00	4.74	5.00	5.73	8.00	6.89

Kuten kuvasta 4 voidaan havaita, lineaarinen regressiosuora on kaikissa kuvaajissa lähes sama. Kvartetti osoittaa hyvin sen, miten numeroista ei juuri pystytä päättelemään datan muodostamia kuvioita, mutta kuvaajista ne ovat nähtävissä heti.

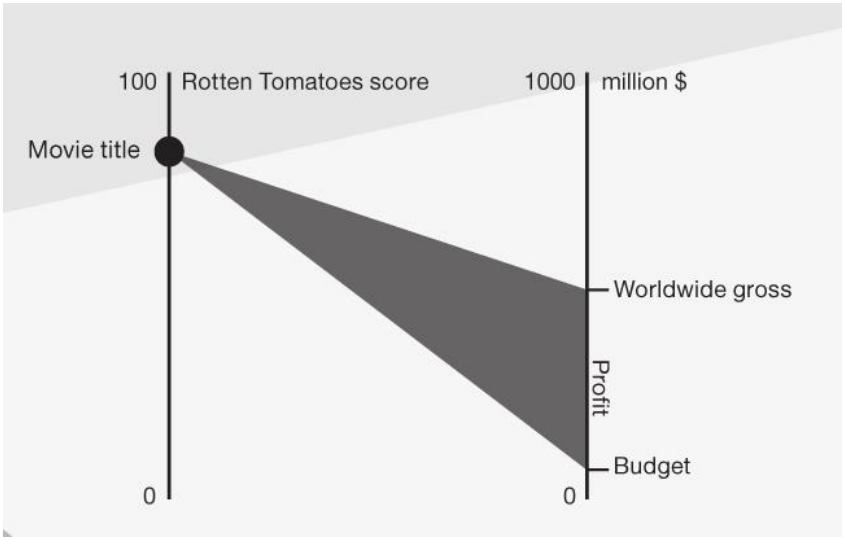


Kuva 4. *Anscomben kvartetti visualisoituna (1973)*

Anscomben (1973) mukaan tilastotieteessä ei kiinnitetä tarpeeksi huomiota kuvaajiin. Vaikka numeeriset laskelmat olisivatkin tarkkoja, kuvaajat auttavat käyttäjää kokonaisymmärryksessä.

3.2.2 Hollywood-elokuvien suosio

Krisztina Szűcsin visualisoinnissa (kuva 5) on vasemmalla pystyakselilla Rotten Tomatoes -verkkosivuston antama pistemäärä ja oikealla elokuvan bruttotulot sekä budjetti. Esitystapa on epätavallinen, joten käyttäjä joutuu miettimään hetken, miten kuvaajaa luetaan. Andy Kirkin (2012, s. 25) mukaan lähestymistapa on kuitenkin oikeutettu, kunhan käyttäjän on mahdollista ymmärtää lopulta, mitä visualisoinnilla haetaan.



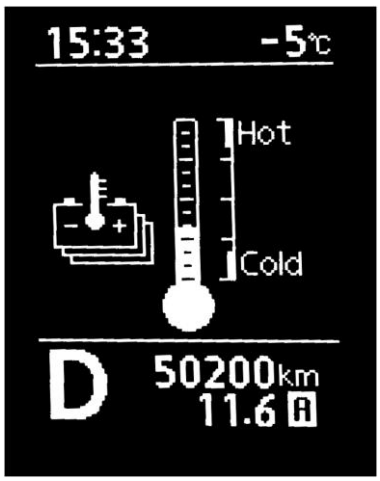
Kuva 5: *Spotlight on Profitability (Szűcs)*

Visualisointi on innovatiivinen ja mieleenpainuva, sillä suurempi nettotuotto kuvautuu leveämmäksi valokeilaksi. Vaikka tämänkaltainen taiteellinen lähestymistapa ei välttämättä tuntuisi oleelliselta tekniikan tai teollisuuden kontekstissa, johtopäätöksenä kuitenkin se, että tavanomaisten kuvaajien sijaan voidaan keksiä aina uusia tapoja esittää tietoa.

3.2.3 Sähköauton akuston lämpötila

Esimerkkinä tiedon piilottamisesta visualisoinnin avulla on osa Nissanin e-NV200 -sähköauton mittaristosta kuvassa 6. Akuston lämpötilaa kuvataan lämpömittarin muotoisella pylväsdiagrammilla.

Kuvaajalle ei kuitenkaan ole määritetty numeerista asteikkoa, joten käyttäjälle ei ilmeisesti jostain syystä haluta kertoa tarkkaa lämpötilaa. Mittaristo on muutenkin pieni, joten lämpötilan seuraamisesta on tehty erityisen hankalaa. Puhdas numeroarvo olisikin tässä tapauksessa huomattavasti havainnollisempi.



Kuva 6: *Akuston lämpötila Nissan e-NV200:n mittaristossa*

4. KÄYTÄNNÖN OSUUS

4.1 Suunnitelma

Aluksi hahmoteltiin seuraavanlaisen lista työvaiheista:

- Yksinkertaisen ohjelman toteuttaminen jollekin Festolinjan työasemista
- Yhteyden muodostaminen ohjelmoitavaan logiikkaan
- Tiedon tallennus esimerkiksi CSV-tiedostoon
- Tiedoston avaaminen ja prosessointi esimerkiksi JavaScriptilla
- Visualisointi verkkoselainkäyttöliittymässä

Koska LCFA-kurssin suorituksesta oli kulunut aikaa, tallessa ei ollut toimivia PLC-ohjelmia. Lisäksi ohjelmointiin käytettävä ympäristö oli vaihtunut TIA Portaliin, jonka käytöstä ei ollut aiempaa kokemusta. Jouduinkin kysymään neuvoa ja sain käyttööni Distribution-työasemalle Demo Factory -kurssilla toteutetun ohjelman. Näin päästiin toteutuksen kanssa sujuvasti alkuun eikä ohjelmaan tarvinnut tehdä kuin pieniä muutoksia.

4.2 Laitteisto

Festolinja koostuu modulaarisista työasemista, joita ohjaavat Siemens S7-300 -logiikat. CPU-korttien lisäksi konfiguraatioissa on 343 Ethernet-kortit, joiden kautta ohjelmia pystytään muokkaamaan. Tässä työssä Ethernetia käytettiin myös tiedon keräämiseen.

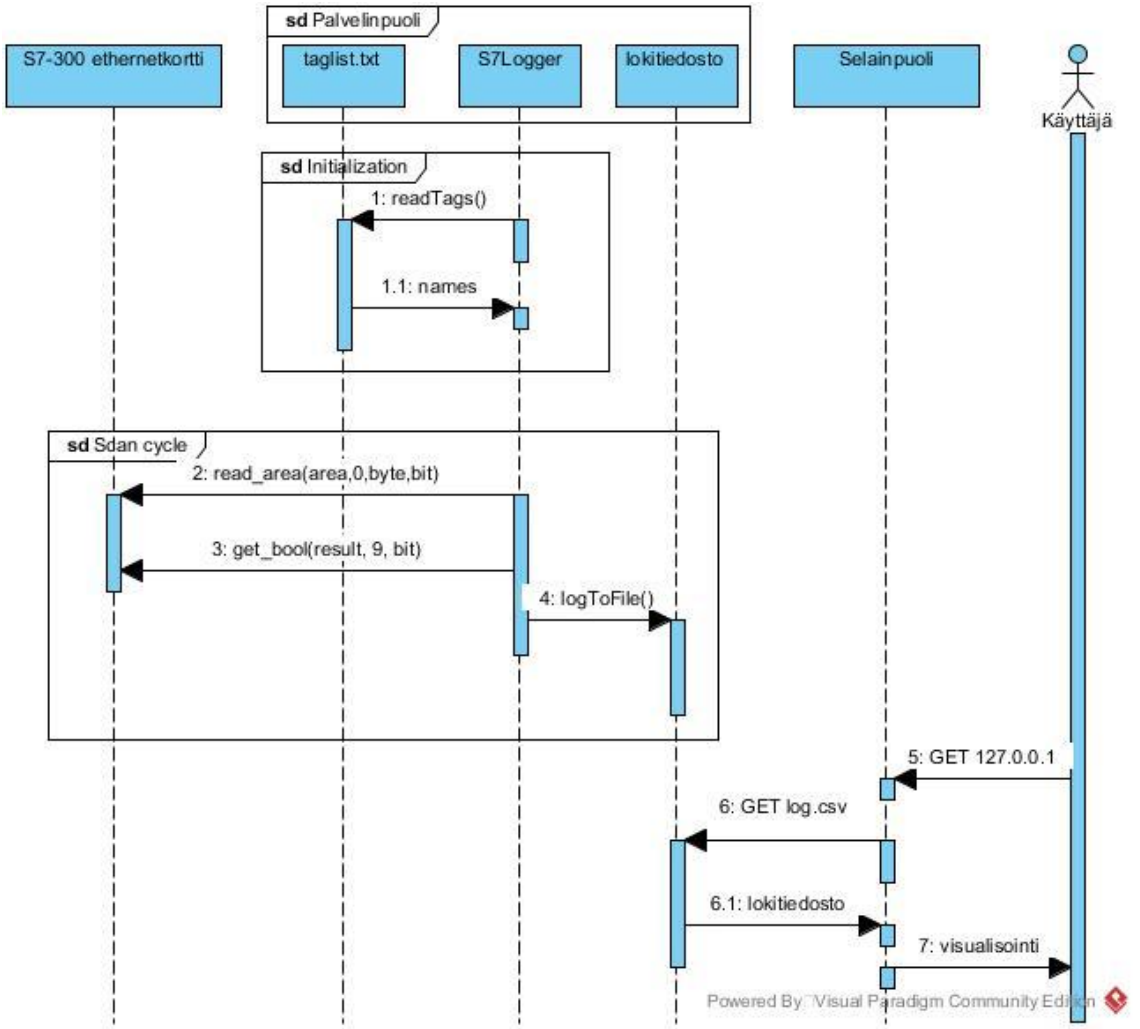
Työasemat käyttävät lähinnä digitaalisia tuloja ja lähtöjä (engl. digital input/output), joten muistiosoitteiden arvot olivat totuusarvomuuttujia (engl. Boolean). Tämä asetti haasteen tiedon keräämisen suhteen, sillä toteutettu ohjelma skannaa muistiosotteiden arvoja tietyin väliajoin, eikä saman True-arvon tallentaminen moneen kertaan tunnu järkevältä. Sovellukseen toteutettiin nousevan reunan tunnistus, jotta CSV-tiedoston rivimäärä pysyi järkevänä. Skannauksen intervalli on tallennettu scantime-muuttujaan, jonka arvoksi asetettiin 0,1 sekuntia.

Toteutettu sovellus pyöri kokonaan kannettavalla tietokoneella. Mikäli lokitiedoston loisi johonkin pilvipalveluun tai tekisi verkkoon tarvittavat porttiohjaukset, voisi selainkäyttöliittymää tarkastella mistä vain.

4.3 Sovelluksen toiminta

Palvelinpuoli pyöri Pythonin päällä ja se käyttää apunaan kahta tekstitiedostoa. Ensimmäiseen on tallennettu halutut muistiosoitteet ja niiden nimet. Nämä tiedot luetaan ohjelman käynnistyessä välimuistiin. Lokitiedostoon taas tallennetaan tiedot muistiosoitteiden

tiloissa tapahtuneita muutoksia. Palvelinpuoli myös tarjoaa lokitiedoston selainkäyttöliittymälle visualisoitavaksi.



Kuva 7: *Sekvenssikaavio sovelluksen toiminnasta*

Kuvassa 7 on esitetty toiminta suurpiirteisesti sekvenssikaavion avulla. Alaluvuissa on esitelty lyhyesti keskeisimmät kirjastot, moduulit ja työkalut sekä kerrottu tarkemmin sovelluksen toiminnasta. Githubista (Angelma, 2017) löytyvään ohjelmakoodiin on myös lisätty runsaasti kommentteja, joissa esimerkiksi muuttujien käyttöä selitetään.

4.3.1 Snap7-kirjasto

Snap7 on avoimen lähdekoodin kirjasto, joka on kehitetty Siemensin S7 -logiikoiden kanssa kommunikointiin Ethernetin yli. Se on saatavilla eri alustoille ja sitä kehitetään edelleen aktiivisesti (Nardella 2016). Kirjaston kehitys on tapahtunut takaisinmallintamalla (engl. reverse engineering). Sitä on myös käytetty tutkimuksissa, joissa on yritetty havaita tunkeutumista (engl. intrusion detection) (Kleinmann & Wool 2014).

Kirjaston avulla pystyy lukemisen lisäksi kirjoittamaan muistiosoitteisiin arvoja. Logiikalta saadaan myös haettua erilaisia tilatietoja. Python-version dokumentaatio listaa esimerkiksi seuraavat funktiot Client-luokalle:

- `ab_write(start, data)`
- `get_cpu_info()`
- `list_blocks()`
- `set_connection_params(address, local_tsap, remote_tsap)`

Tässä työssä painotus oli kuitenkin tiedon tallentamisella, joten sen kannalta tärkein funktio oli *read_area(area, dbnumber, start, size).* Samalla funktiolla pystyy lukemaan niin digitaalisia kuin analogisiakin sisään- ja ulostuloja sekä muistiarvoja (engl. merker).

4.3.2 Palvelinpuolen toteutus

Esimerkkitoteutuksia selatessani päädyttiin käyttämään kirjaston Python-versiota. Kehitys tapahtui aluksi tekstieditorin (Sublime Text) ja komentorivin avulla, mutta sittemmin otettiin käyttöön PyCharm-kehitysympäristö. Sublime Text tarjoaa tekstieditoriksi hyvin toiminnollisuuksia, mutta ei osaa huomauttaa esimerkiksi syntaksivirheistä. Komentoriviltä ajaessa Python-kaatuu, mikäli ulkopuolinen kirjasto heittää (engl. throw) virheen. Näin tapahtuu esimerkiksi yrittäessä ajaa sovellustani online-moodissa ilman että sovellus on yhdistyneenä FAST-Labin lähiverkkoon. PyCharm sen sijaan osasi ottaa *snap7.snap7exceptions.Snap7Exception: TCP: Unreachable peer* -virheilmoituksen kiinni.

Snap7:n dokumentaatio ei ollut kaikilta osin täysin selkeä, mutta yrityksen ja erehdyksen kautta löysin tarvitsemani toiminnallisuudet. Toteutus haluttiin pitää mahdollisimman joustavana, joten muistiosotteita käsitellään olioina. Ohjelman käynnistyessä halutut muistiosoitteet luetaan siis taglist.txt-tekstitiedostosta ja nimistä luodaan oliot, joiden attribuuteiksi tyyppi ja itse muistiosoite tallennetaan. Toteutuksessa näille olioille on määritetty funktio *readIO(self)*, jonka sisältä Snap7:n toiminnallisuutta *read_area(…)* kutsutaan. Alla esitetty ohjelmakoodi toimii seuraavasti: mikäli olion *ioType* on Q, muuttujaan *result* luetaan *process output* -alue kyseiselle oliolle määritetystä muistiosoitteesta (*self.byte*). Tämän alueen avulla taas haetaan kyseinen totuusarvomuuttuja oliolle määritetyn muistiosoitteen sisältä (*self.bit*).

```
if self.ioType == "Q": # [’PA’] = Process output
    result = plc.read_area(areas['PA'], 0, int(self.byte), S7WLBit)
return get_bool(result, 0, int(self.bit))
```

Linjalta tunnistetut nousevat reunat päädyttiin tallentamaan CSV-tiedostoon, jonka kentät koostuvat muistiosoitteen nimestä (Memory), tyypistä (Type), päivämäärästä (Date) ja aikaleimasta (Time). Päivämäärä ja kellonaika luetaan Pythonin time-kirjastosta skannaussyklin aikana jokaiselle tunnistetulle reunalle. Muistiosoiteoliot ovat tallennettuna sanakirjaan (engl. dictionary), jota käydään skannauksen aikana läpi.

4.3.3 Javascript-kirjastot ja selainpuoli

Käytetty Highcharts-visualisointikirjasto (http://www.highcharts.com/) on helppokäyttöinen, mikäli tieto on sopivasti taulukoituna. Se myös tarjoaa erillisen Data-moduulin, jolla esimerkiksi CSV-tiedostoja voi jäsentää. Työssä päädyttiin kuitenkin käyttämään erillistä PivotTable.js-kirjastoa, sillä tämä Excelistä tuttu ominaisuus on osoittautunut käteväksi. Lisäksi sille oli valmiiksi tarjolla C3, D3 ja Google Charts -visualisointityökalut. Dokumentaatiota selatessa löydettiin kuitenkin NovixPivotJS-liitännäinen, jolla PivotTablen tiedot sai visualisoitua myös alun perin valittua Highchartsia käyttäen. Viitteet näihin kirjastoihin löytyvät Github-sivulta (Angelma 2017).

Selainpuoli siis hakee käyttöönsä jQuery, Papa Parse, HandsOnTable, PivotTable, Highcharts ja NovixPivotJS -kirjastot. Palvelinpuoli tarjoaa CSV-tiedoston Papalle jäsennettäväksi, joka taas syöttää jäsennetyn datan PivotTablelle. Käyttäjä voi tämän jälkeen valita selainkäyttöliittymässä haluamansa visualisoinnin eri vaihtoehdoista. Eri kuvaajien lisäksi tietoa voi suodattaa (engl. filter). Alaluvussa 4.5 on esitetty muutama esimerkki.

4.3.4 Vaihtoehtoiset työkalut

Palvelinpuolelle tarjolla olisi ollut Libnodave (Hergenhahn 2011), joka tarjoaisi suurin piirtein Snap7 -kirjastoa vastaavat toiminnallisuudet. Kirjoitushetkellä uusin päivitys Libnodaveen oli toukokuulta 2014, joten sen kehittämisestä on ilmeisesti luovuttu. Snap7:n uusin versio oli joulukuulta 2016, mutta käyttettyyn Python-version Githubiin viimeisin päivitys oli maaliskuulta 2017. Maksullisesta Kepwaren kehittämästä Siemens TCP/IP Ethernet -ajurista olisi ollut saatavilla maksuton kokeiluversio, jota pystyy käyttämään kaksi tuntia kerrallaan (www.kepware.com).

Selainpuolella Papa Parsen, PivotTablen ja NovixPivotJS:n sijaan olisi todennäköisesti voinut käyttää Highchartsin Data-moduulia. Se tarjoaa CSV-tiedostojen latauksen lisäksi muun muassa työkaluja päivämäärien käyttöön, mikä olisi tarpeen, jos rakennettuja visualisointeja haluaisi skaalata ajan mukaan. Toteutuksessa päädyttiin jättämään päivämäärät ovat pelkiksi merkkijonoiksi, sillä visualisointia ei tässä vaiheessa oikeasti tarvinnut käyttää muuhun kuin demotarkoituksiin.

Mikäli visualisoinnista olisi halunnut kustomoidumman, olisi kannattanut ehkä valita Highchartsin sijaan D3.js (Bostock 2015). Sen kompleksisuus aiheuttaa kuitenkin jyrkemmän oppimiskäyrän, mutta sitä pidetään joustavimpana JavaScriptin visualisointikirjastona.

4.4 Haasteet ja ongelmat

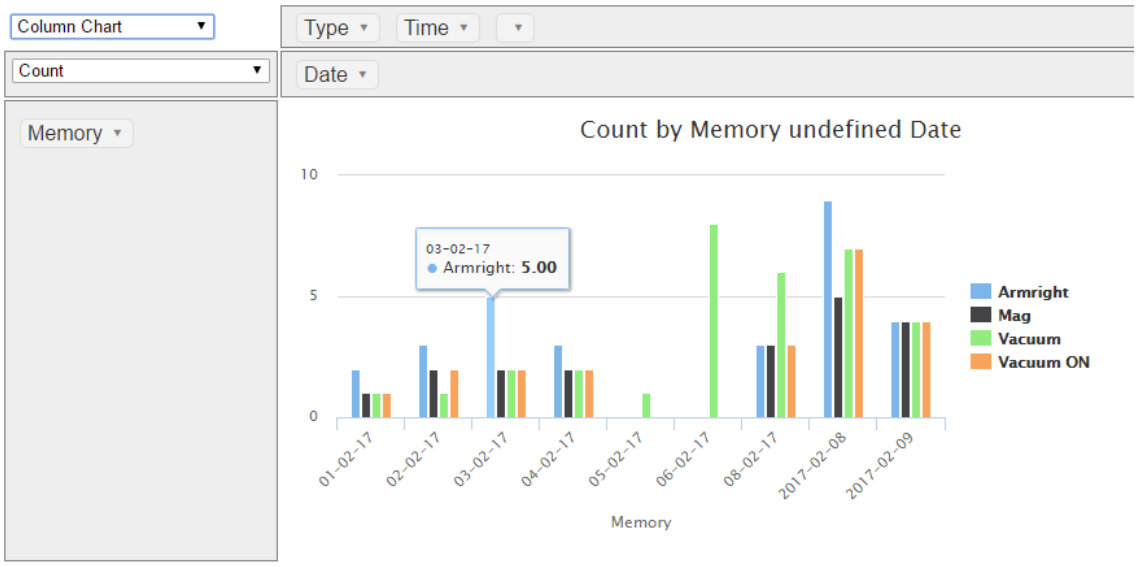
Vaikka Festolinja olikin tuttu *Laboratory Course in Factory Automation* -kurssilta, oli kahdessa vuodessa muutama asia muuttunut. Pelkästään linjaan yhteyden saaminen tuotti

hankaluuksia, niin Ri208:n tietokoneilta kuin kannettavalta tietokoneeltakin. Verkkokortille oli asetettu staattinen IP-osoitteen *FASTory*-linjaa varten, mutta se piti muuttaa DHCP:ksi (engl. Dynamic Host Configuration Protocol) Festolinjan takia.

Eri kirjastojen dokumentaatiot olivat hajallaan pitkin verkkoa. Esimerkkitoteutuksia ei muutenkaan ollut liikaa, mutta koska käytetyt kirjastot perustuvat avoimeen lähdekoodiin, niiden sisäistä toimintaa pääsi tarkastelemaan. Aina ei kuitenkaan ollut selvää, varsinkaan selainpuolella, mikä kirjasto vastaa minkäkin osan toiminnallisuudesta. Täten esimerkiksi visualisoinnin reaaliaikainen päivitys jäi toteuttamatta. Palvelinpuolella eniten apua oli *Simply Automationized* -blogin videoista ja ohjelmakoodeista (simplyautomationized.blogspot.fi 2016).

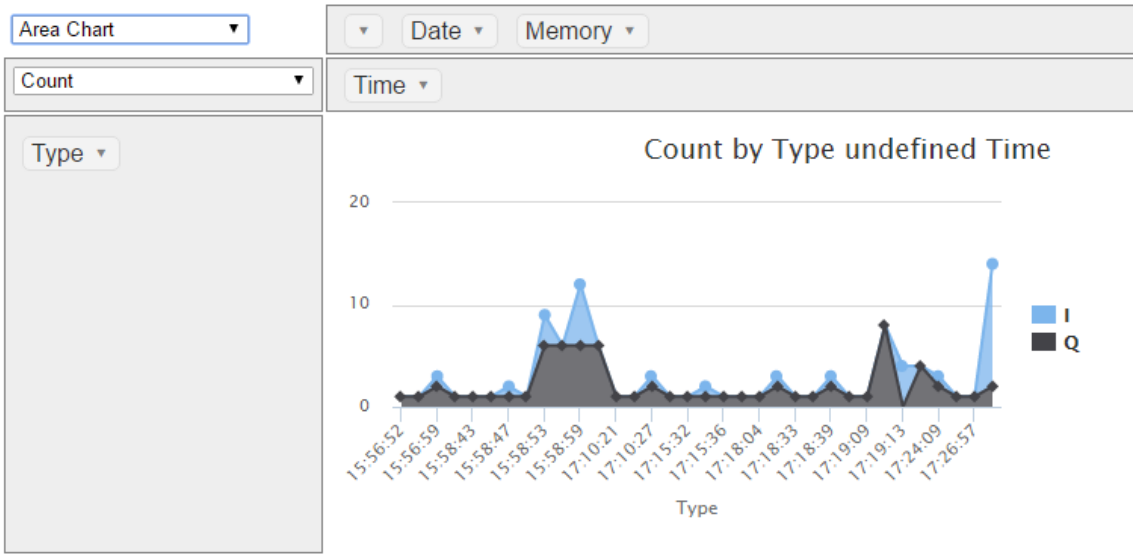
4.5 Lopputulos

Sovellus käynnistetään ajamalla S7Logger.py. Python-ohjelma skannaa halutun PLC:n muistiosotteita ja tallentaa havaitsemansa kytkentäreunat CSV-tiedostoon. Samaan aikaan Python toimii myös palvelimena, jolloin selainkäyttöliittymän Javascript pääsee noutamaan kyseisen lokitiedoston. Edellisissä luvuissa mainittujen kirjastojen avulla tieto saadaan visualisoitua ja lopputulos näyttää seuraavalta. Kuvaan 8 on valittu eri toimilaitteiden käyttömäärät päivän mukaan ja kuvaajaksi pylväsdiagrammi.



Kuva 8: *Toimilaitteiden käyttö eri päivämäärinä*

PivotTable.js mahdollistaa eri kuvaajatyyppien valinnan pudotusvalikosta. Myös visualisoitavia asioita pystyy vaihtamaan ja suodattamaan. Kuvassa 9 on eritelty sisään- ja ulostulot kellonajan mukaan. Kuten päivämäärät, myös kellonajat käsitellään tässä toteutuksessa merkkijonoina.



Kuva 9: *Sisään- ja ulostulot kellonajan mukaan*

4.6 Avoimen lähdekoodin lisenssit

Highchartsia saa käyttää voittoa tavoittelemattomaan (engl. non-commercial) käyttöön *Creative Commons Attribution-NonCommercial* -lisenssillä. Kyseinen lisenssi sallii jakamisen ja muokkaamisen, kunhan muutoksista ja alkuperäisestä tekijästä. Muut hyödynnetyt kirjastot käyttävät MIT-lisenssiä, joka antaa oikeudet muokata, kopioida ja käyttää teosta vapaasti, ehtona lisenssin tekstin säilyttäminen lähdekoodissa. MIT-lisenssi ei kuitenkaan vaadi lähdekoodin julkistamista.

Koska kandidaatintyöt ovat muutenkin julkisia ja kaikki käytetyt kirjastot perustuivat avoimeen lähdekoodiin, sovelluksen tiedostot julkaistiin Githubissa (Angelma 2017) MIT-lisenssillä. Sovellus toimii toisaalta hyvänä referenssinä, ja toisaalta muiden on helppo lähteä sitä jatkokehittämään koska lisenssi on ilmaistu selkeästi.

4.7 Jatkokehitys

Kuten aiemmissa luvuissa mainittiin, Highchartsin data-moduulia kannattaisi hyödyntää eli käsitellä päivämääriä olioina eikä merkkijonoina. Pidemmän aikavälin visualisoinneista saataisiin näin paljon käyttökelpoisemmat. Jatkuvan monitoroinnin tapauksessa voisi olla myös järkevää käyttää jotain oikeaa tietokantaa tallennukseen CSV-tiedoston sijasta.

Palvelinpuolen ohjelmaa voi toki ajaa millä tahansa verkkoon liitetyllä tietokoneella, mutta luontevaa olisi käyttää jatkuvaan monitorointiin DIN-kiskoon asennettua Raspberry Pi:tä. Raspberry Pi on erityisen suosittu kotiautomaatiojärjestelmissä, mutta sitä käytetään ilmeisesti jonkin verran myös teollisuudessa (Li et al. 2015).

Kaikkien Festolinjan laitteiden monitorointi onnistuisi samalla ohjelmalla, luomalla jokaisesta PLC:sta oma olionsa. Testasin tätä loppuvaiheessa tallentamalla sekä Distribution- että Handling -asemien Start-painikkeiden tiloja, joiden molempien muistiosoite oli %1.0. Tietorakenteiden monimutkaisuuden takia tätä toiminnallisuutta ei viety tässä työssä pidemmälle.

5. YHTEENVETO

Esineiden internetin hyödyntäminen teollisuudessa avaa uusia mahdollisuuksia. Kun työkone, teollisuuslaite tai vaikka ajoneuvo on yhdistetty internettiin, sen kuntoa ja käyttöä voidaan seurata keräämällä dataa. Monitoroinnilla mahdollistetaan myös esimerkiksi reaaliaikaiset hälytykset ja ilmoitukset. Kerättyä tietoa voidaan hyödyntää myös muun muassa tuotekehityksessä, kunnossapidossa, markkinoinnissa ja myynnissä.

Pelkkä tallennettu data ei kuitenkaan yleensä riitä luomaan kokonaisvaltaista ymmärrystä tietyn tuotteen, laitteen tai prosessin käyttäytymisestä. Datalle voidaan toki soveltaa erilaisia tilastollisia analyysejä, mutta usein vasta graafisella esityksellä voidaan havaita esimerkiksi trendejä, korrelaatioita ja epätavallisuuksia. Visualisointia voidaankin pitää havainnoinnin työkaluna. Onnistuneen visualisoinnin rakentaminen ei kuitenkaan ole välttämättä yksinkertaista, mutta esimerkiksi Andy Kirkin esittämällä metodologialla voi päästä hyviin lopputuloksiin.

Tämän kandidaatintyön käytännön osuudessa tutustuttiin teollisuuslogiikalta kerätyn datan visualisointiin. Tietoa tarjoavana laitteena käytettiin Siemens S7-300 -logiikkaa TTY:n tutkimuslaboratoriossa. Tieto tallennettiin Python-ohjelman avulla taulukkomuodossa tekstitiedostoon ja visualisoitiin selainkäyttöliittymässä. Vaikka sovellukselle ei asetettu erityisen selkeitä tavoitteita, voidaan sitä pitää onnistuneena, sillä missään vaiheessa ei kohdattu ylitsepääsemättömiä haasteita. Yhteenveto sovelluksesta on kasattu taulukkoon 2. Taulukossa on esitetty myös lyhyesti sovelluksen jatkokehitysmahdollisuuksia.

Taulukko 2: *Yhteenveto käytännön sovelluksesta*

Aihe	Ohjelmoitavalta logiikalta tallennetun tiedon visualisointi
Tallennettava tieto	Sisään- ja ulostulojen nousevat reunat, tyyppi, aika ja päivämäärä
Palvelinpuolen ohjelma	Python, Snap7-kirjasto, tallennus CSV-taulukkoon, n. 280 riviä
Vahvuudet	Muistiosoitteiden käsittely olioina, offline-testaustila, julkaistu MIT-lisenssillä
Heikkoudet	Ei sovellu suoraan usean logiikan eikä muiden kuin digitaalisten sisään- ja ulostulojen monitorointiin
Selainpuolen ohjelma	HTML/Javascript, visualisointina Higcharts (taustalla useita muita kirjastoja), n. 70 riviä
Vahvuudet	Mahdollistaa näkymien muokkauksen PivotTable-ominaisuudella, skaalautuva
Heikkoudet	Päivämäärien käsittely merkkijonoina
Muu jatkokehitys	Raspberry Pi:n käyttö, Highchartsin data-moduuli, tietokannan käyttö, useamman laitteen monitorointi

Keskeisimpinä havaintoina käytännön sovelluksen kannalta voidaan pitää suurta hyötyä avoimen lähdekoodin kirjastoista. Näiden avulla vanhaltakin teollisuuslogiikalta pystyttiin keräämään dataa helposti. Lisäksi tätä tallennettua tietoa voitiin edelleen jäsentää ja esittää visualisoituna selainkäyttöliittymässä. Toteutetun sovelluksen jatkokehitys mahdollistettiin julkaisemalla se MIT-lisenssillä.

LÄHTEET

Angelma, M. (2017), *S7Logger*, Github-sivusto. Saatavissa (viitattu 20.03.2017): https://github.com/mangelma/S7Logger.

Anscombe, F. (1973), *Graphs in Statistical Analysis*, The American Statistician, Vol. 27, s. 17-21.

Mike Bostock 2015, *D3.js - Data-Driven Documents*, verkkosivu. Saatavissa (viitattu 20.03.2017): https://d3js.org.

Degeler, A. (2015), *Data Visualization Tools For IoT*, blogikirjoitus. Saatavissa (viitattu 20.03.2017): https://stanfy.com/blog/data-visualization-tools-for-iot.

Electrical Engineering Blog (2013), *The top most used PLC Systems around the world*, blogikirjoitus. Saatavissa (viitattu 20.03.2017): http://engineering.electrical-equipment.org/electrical-distribution/the-top-most-used-plc-systems-around-the-world.html.

Highsoft, *Interactive JavaScript charts for your webpage | Highcharts*, verkkosivu. Saatavissa (viitattu 20.03.2017): http://www.highcharts.com.

Kirk, A. (2012). *Data Visualization: a successful design process*, Packt Publishing.

Kepware, *Siemens TCP/IP Ethernet Driver*, verkkosivu. Saatavissa (viitattu 20.03.2017): https://www.kepware.com/en-us/products/kepserverex/drivers/siemens-tcp-ip-ethernet.

Li, W. J., Tung, S. C., & Huang, S. M. (2015), *Web-Based Supervisory Control System Based on Raspberry Pi*, Applied Mechanics and Materials, Vol 764-765, s. 640-643.

Nardella, D. (2016), *Snap7 Homepage*, verkkosivu. Saatavissa (viitattu 20.03.2017): http://snap7.sourceforge.net.

Porter, M. E. & Heppelman, J. E. (2014), *How Smart, Connected Products Are Transforming Competition*, Harvard Business Review, 92 (11), 64-88.

Siemens, *Universal Controller SIMATIC S7-300 - PLCs – Siemens*, verkkosivu. Saatavilla (viitattu 20.03.2017): http://w3.siemens.com/mcms/programmable-logic-controller/en/advanced-controller/s7-300/pages/default.aspx.

Simply Automationized (2016), *Raspberry Pi - Python Snap7 - Mapping and Reading Datablocks*, blogikirjoitus. Saatavissa (viitattu 20.03.2017): http://simplyautomationized.blogspot.fi/2016/02/raspberry-pi-python-snap7-data-blocks.html

Szűcs, K. *Spotlight on Profitability*, verkkosivu. Saatavissa (viitattu 20.03.2017): http://krisztinaszucs.com/?my-product=hollywood.