TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

KLAUS TELENIUS
VAISALA ROSA-SÄÄASEMAN KÄYTTÖLIITTYMÄTOTEUTUS

Kandidaatintyö

Tarkastaja: Mikko Salmenperä
7. tammikuuta 2019

TIIVISTELMÄ

Klaus Telenius: Vaisala ROSA-sääaseman käyttöliittymätoteutus
Tampereen teknillinen yliopisto
Kandidaatintyö, 15 sivua, 1 ohjelmatiedostokansio
Marraskuu 2018
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Automaatiotekniikka
Tarkastaja: Yliopisto-opettaja Mikko Salmenperä

Avainsanat: Esineiden internet, Käyttöliittymä, Data, Tiedonsiirto, REST, API

Työssä luodaan yliopiston sääasemalle toteutus luomaan käytettävyyttä, sekä parantamaan tiedon saatavuutta. Tavoitteeseen pyritään esineiden internetin tekniikoilla. Työssä perehdytään esineiden internetiin ja sen rakenteeseen. Erityisesti työssä on tarkoituksena käyttää ohjelmistoympäristöä Node-RED.

Työkalujen avulla luodaan sääaseman tiedonsiirtolinkki väliohjelmistolle. Väliohjelmistossa luodaan käyttöliittymä, josta siirrettyjä tietoja tarkastellaan. Ohjelmisto tallentaa tiedot tietokantaan, ja niitä voi hakea käyttäen käyttöliittymään rakennettua toiminnallisuutta.

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

API	engl. Application programming interface
DLL	engl. Data link layer eli siirtoyhteyskerros
FBD	engl. Flow based programming
HTML	engl Hypertext markup language
HTTP	engl. Hypertext transfer protocol
IDS	engl. Intrusion detection system
IoT	engl. Internet of Things, esineiden internet
IPv6	engl. Internet protocol version 6
JSON	engl. JavaScript object notation
MAC	engl. Medium access control protocol
MongoDB	Tietokantasovellus
OSI-referenssimalli	engl. Open systems interconnection –malli
PUB	Publish-pistoke tietojen julkaisua varten
REST	engl. Representational state transfer
RESTful	palvelu, joka toteuttaa REST suunnitteluperiaatteet
SOA	engl. Service oriented architecture
SOAP	engl Simple object access protocol
SUB	Subscribe-pistoke johon käytäjä yhdistää vastaanottaakseen PUB-pistokkeen julkaisemat viestit
TCP	engl. Transmission control protocol
UDP	engl. User datagram protocol
URI	engl Uniform resource identifier
URL	engl Uniform resource locator, URI:n erikoistapaus
XML	engl. Extensible Markup Language
ZeroMQ	Aynkroninen viestinvälityskirjasto

1. JOHDANTO

Internetin levittyä tietokoneista laitteisiin, koneisiin ja kulkuneuvoihin prosesseista saatavan datan määrä on lisääntynyt nopeasti. Laitteet pystyvät aistimaan ympäristöään ja tuottamaan mittausdataa hyödyntäen erilaisia sensoreita, toimilaitteita ja radiotaajuuksien etätunnistussiruja. Mittausdataa tallennetaan ja hyödynnetään pilvipalveluiden avulla. Datan hyödyntäminen lisää laitteiden älykkyyttä, parantaa prosessien tehokkuutta sekä mahdollistaa laitteiden välisen työskentelyn yhteisen tavoitteen saavuttamiseksi. Laitteiden verkottaminen on saanut termin esineiden internet (engl. Internet of Things, IoT). [1]

IoT on saavuttanut 2010-luvulla suurta huomiota ja verkottuneiden laitteiden luku kasvaa jatkuvasti. Eksponentiaalisesti lisääntyvä datan määrä asettaa korkeampia vaatimuksia tiedonsiirrolle sekä verkon kapasiteetille. Suurten datamassojen käsittely luo tarpeen datan esittämiseen hyödyllisessä muodossa. Tarpeeseen vastaavat erilaiset sovellukset, jotka suodattavat dataa ja saattavat sen käyttäjälle koherenttiin muotoon.

Tämän työn tavoitteena on tutkia esineiden internetin sovellusarkkitehtuuria, sekä soveltaa sitä käytännössä. Tarkoituksena on miettiä, miten kiinteitä laitteistoja, ohjelmistoja, sekä niiden välistä tiedonsiirtoa yhdistetään esineiden internetin mukaisen arkkitehtuurin aikaansaamiseksi. Lisäksi luodaan käytännön esimerkki. Sovelluskohteena toimii ROSA -sääasema.

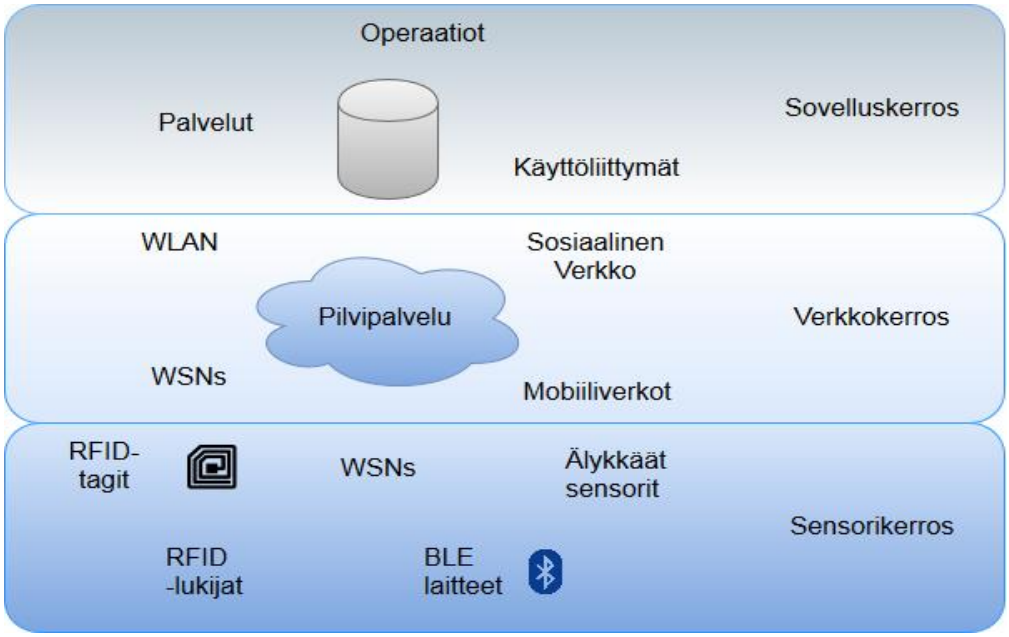
Työssä käsitellään aluksi esineiden internetiä ja sen arkkitehtuuria. Sen jälkeen keskitytään työssä käytettäviin työkaluihin ja siirrytään toteutukseen. Työn lopuksi luodaan säätietojen hakemisen mahdollistava toiminnallisuus.

2. ESINEIDEN INTERNET

Esineiden internetin ekosysteemi voidaan määritellä globaalina verkostona ja palveluinfrastruktuurina, joka koostuu virtuaalisen identiteetin lisäksi fyysisiä ominaisuuksia sisältävistä Internetiin integroiduista laitteista ja esineistä, jotka muodostavat älykkään ja itsenäisesti toimivan ekosysteemin perustuen yhteensopiviin ja standardoituihin protokolliin [2]. Käytännössä esineiden internetillä tarkoitetaan esimerkiksi laitteita, joilla on mahdollisuus langattomasti kommunikoida keräämänsä data toisille laitteille hyödyntäen prosessissa pilvipalvelua ja yhteensopivia tiedonsiirtomenetelmiä. Yhtä laitetta kutsutaan yleisesti solmuksi (engl. *Node*). Kuvattua kokonaisuutta lisäosineen ja sovelluksineen nimitetään IoT-järjestelmäksi.

2.1 IoT-järjestelmän rakenne

Esineiden internetin arkkitehtuurin perusperiaatteena voidaan pitää kuvan 1 kolmikerroksista arkkitehtuuria, joka koostuu sovelluskerroksesta, tietoverkkokerroksesta sekä sensorikerroksesta [3]. Järjestelmän arkkitehtuuria suunniteltaessa on otettava huomioon useita alueita, kuten tietoverkot, kommunikaatiomenetelmät, tietoturvaratkaisut, sekä järjestelmään liittyvät bisnesmallit ja prosessit [2, s. 4].



Kuva 1. *IoT-järjestelmän 3-kerrosrakenne, perustuu lähteisiin [2] [4]*

Sensorikerros sisältää kaikki fyysistä maailmaa mittaavat ja havainnoivat laitteet. Kerros on arkkitehtuurin pohjimmainen. Se on vastuussa solmujen yhdistämisestä internetiin sekä niiden tuottaman prosessoidun datan lähettämisestä ylemmille kerroksille.

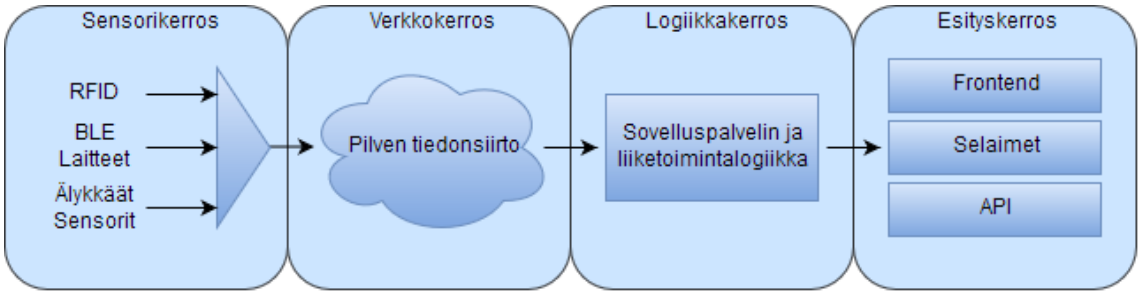
Verkkokerrokseen integroidaan järjestelmälle tärkeät laitteet ja tietoliikennetekniikat. Niiden avulla luetaan ja käsitellään sensorikerroksen tuottama data. Kerroksen vastuulla on jakaa data oikeille palveluille ja operaatioille.

Sovelluskerrokseen luodaan verkkokerrokselta saadun datan perusteella halutut palvelut ja operaatiot. Palveluita ja applikaatioita suoritetaan useita samanaikaisesti. Esimerkkinä palveluihin kuuluvat datan tallentaminen sekä esittäminen. [3]

2.2 Pilvipalvelut

Pilvipalvelut ovat internetistä hankittua tietokonekapasiteettia, sovelluksia tai muita palvelusuorituksia. Ne ovat päätelaitteesta ja käyttöpaikasta riippumattomia, ja niiden toimintaa voidaan mitata. [4] Pilvipalvelun käyttäminen mahdollistaa datan keräämisen, prosessoinnin, esittämisen sekä kontrolloimisen siten, että jokainen tehtävä voidaan eritellä ja jakaa sopiville resursseille tai laitteille [5, s. 437]. Kuvan 1 kolmikerrosrakenteessa pilvi sijoittuu verkkokerrokseen. Pilvi on yhdistävä tekijä sensorikerroksen ja sovelluskerroksen välissä, ja siksi olennainen osa mitä tahansa IoT-järjestelmää.

Pilven rakennetta, jolla luodaan valvontanäkymiä, luonnehditaan palvelukeskeisellä arkkitehtuurilla (engl. Service Oriented Architecture, SOA). Malli sisältää verkkokerroksen, logiikkakerroksen sekä esityskerroksen. Verkkokerros sisältää tietokannan tai tiedostojärjestelmän, logiikkakerros sovelluspalvelimen ja liiketoimintalogiikan ja esityskerros selaimen. [4, s. 22] SOA-malli on laajennettavissa myös IoT-järjestelmiin lisäämällä sensorikerros, kuten kuvassa 2.



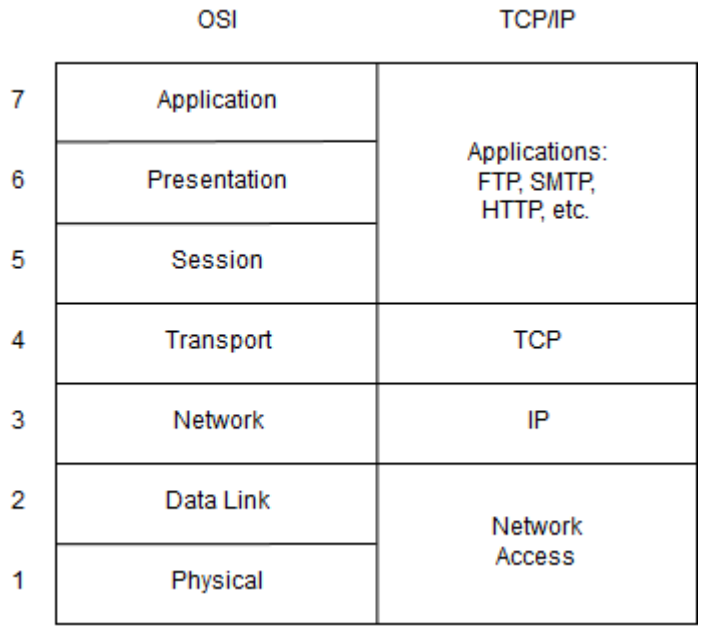
Kuva 2. *Palveluorientoitunut IoT-arkkitehtuuri [2]*

Pilvipalveluiden tärkeimpiä aihealueita on tietoturva. Palvelukoneisto tulee suojata ulkopuolisen tietoliikenteen haitalliselta sisäänpääsyltä. Pilvipalveluissa asiakkaan käyttämä tekninen ympäristön osa turvataan useilla palvelin- ja tietoliikennetekniikan menetelmillä. Suojaamiseen käytetään palomuuria, jota palveluntarjoaja hallinnoi internetin ja

koneiston välillä. Palomuuri laskee sisään vain sallitun osan tietoliikenteestä. Muita suojausmenetelmiä ovat esimerkiksi tunkeutumisenhavaitsemisjärjestelmät (engl. Intrusion Detection System, IDS), joilla oletetun hyökkääjän yhteys palveluun katkaistaan. [4, s. 93]

2.3 Tiedonsiirto

Kuvan 2 sensorikerroksen sensorit ja laitteet muodostavat tiedonsiirron fyysisen kerroksen. Fyysinen kerros on vastuussa digitaalisen datan muuntamisesta sekä vastaanottamisesta. [6, s. 85] Fyysinen kerros sisältää monesti useita solmuja. Solmujen kommunikoimaa dataa ohjataan kerroksen päällä olevalla siirtotien varausprotokollalla (engl. Medium Access Control, MAC). MAC on kuvan 3 OSI-viitemallissa (engl. Open Systems Interconnection reference model) osa Data Link kerrosta. [6, s. 11-112]



Kuva 3. *OSI-referenssimalli, sekä TCP/IP protokollarakenne*

Siirtoyhteyskerros (engl. Data link layer, DLL) varmistaa luotettavan tiedonsiirron saman verkon solmuilta toiselle. Tiedonsiirto tapahtuu paketteina, jotka puretaan ja muodostetaan sensoreilta saatavilla olevasta datasta. Network kerros käyttää kerroksen palveluita pakettien toimittamiseen. Sen muita tehtäviä ovat virheenkäsittely sekä datavirran jatkuvuuden varmistaminen. [6, 149-150]

Tiedonsiirron päätarkoitus on välittää tietoa solmulta toiselle. Siksi tärkeä osa mitä tahansa tietoliikenneverkkoa on eri solmujen tunnistaminen. Tähän käytetään ni-

meämisprotokollia, joilla solmuille annetaan nimi ja osoite. Nimeäminen noudattaa esimerkiksi Internet Protocol version 6 (IPv6) protokollaa. Tällöin solmun osoitteena toimii IP-osoite. Yhdellä solmulla voi olla useita nimiä ja osoitteita. Nimeäminen tehtävänä kuuluu niille protokollille, jotka tarvitsevat osoitteita toimiakseen. [6, 181-182]

Kuljetuskerroksessa käytetyt Transmission Control Protocol (TCP) ja User Datagram Protocol (UDP) yhteyskäytännöt tarjoavat luotettavan tiedonsiirron sovellus- ja Internetkerroksen välillä. UDP on yhteydetön tiedonsiirtotapa, mutta ei tarkista ovatko tiedot menneet perille oikein. TCP sisältää virhetarkastukset ja virheenkorjaukset. [7]

2.4 Sovellukset ja rajapinnat

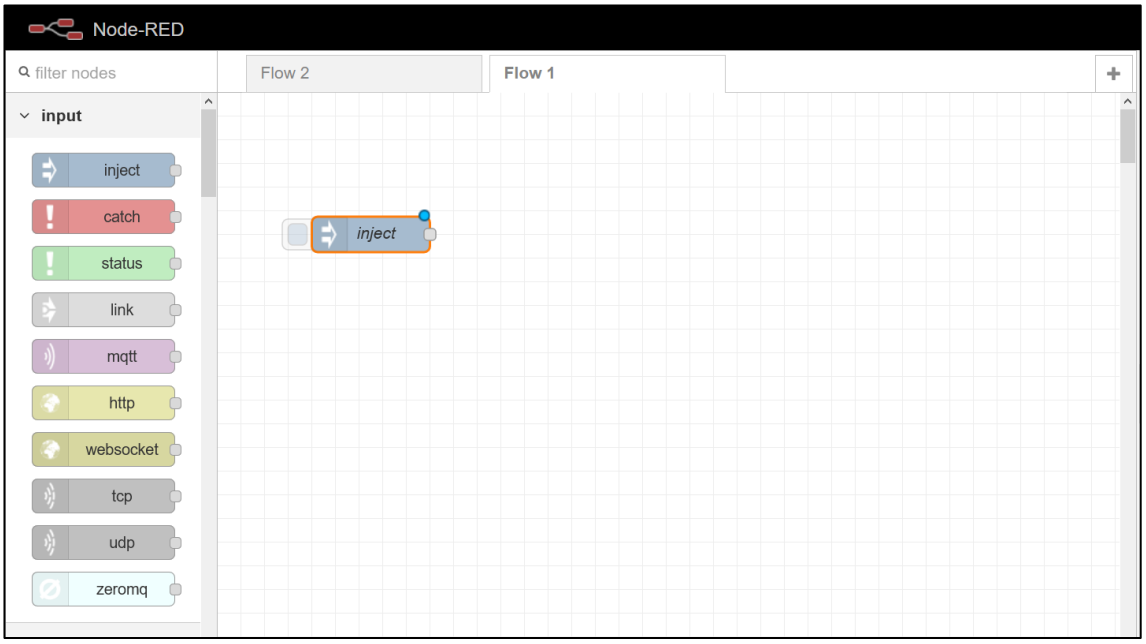
IoT-järjestelmän sovellukset sijoitetaan verkkokerroksen päälle palvelukerrokseen kuvan 2 mukaisesti. Palvelukerros jaetaan yhä palvelu- ja rajapintakerrokseen. Yksittäinen palvelu on toimenpide, kuten datan kerääminen, välittäminen tai tallentaminen. Se voi olla myös näiden yhdistelmä. Jotta sovellus toimii tehokkaasti, palvelu- ja esityskerrokseen tulee sisällyttää rajapintateknologia, palvelunhallinta, resurssien hallinta- ja jakamisteknologia sekä väliohjelmisto. [3, s. 1130-1131] Tämän työn puitteissa on tarpeen esitellä ainoastaan väliohjelmisto.

Väliohjelmisto mahdollistaa sovellusten kehittämisen eliminoiden kehittäjän tarpeen miettiä eri applikaatioiden tai infrastruktuurien välisiä yhteensopivuuksia. Väliohjelmisto piilottaa sen taustalla käytetyn teknologian ja tuo kehittäjän käyttöön standardoidun kehitysympäristön. Mahdollisuudet useiden sovellusten suorittamiseen erilaisilla alustoilla ja erilaisissa toimintaympäristöissä, sekä mahdollisuus palveluiden hajautettuun tiedonkäsittelyyn saavutetaan väliohjelmistojen avulla. [3, s. 1130-1131]

3. NODE-RED

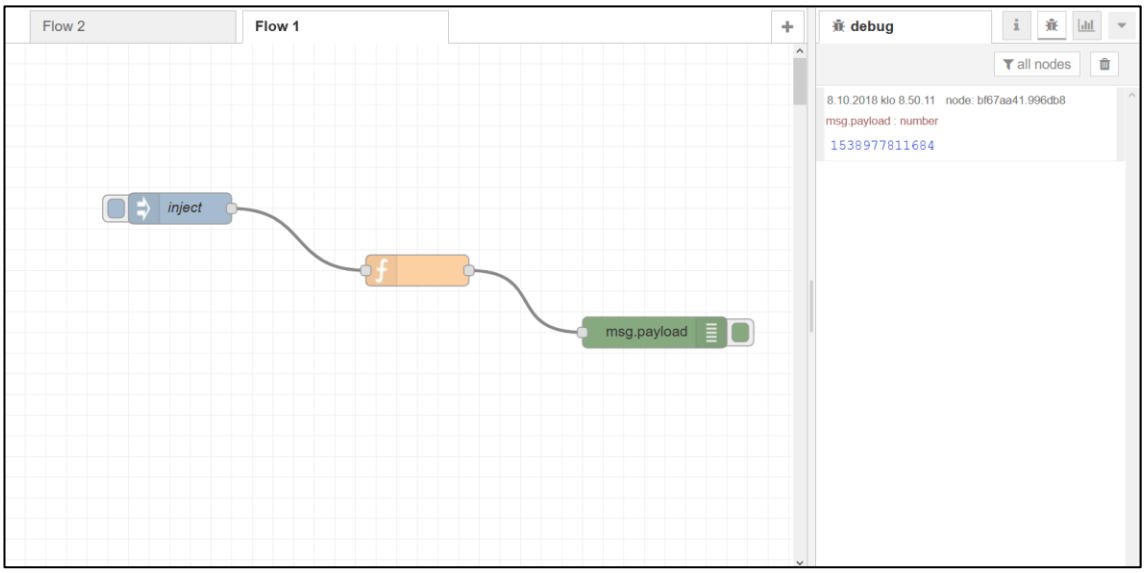
Node-RED toimii aliluvussa 2.4 esiteltynä väliohjelmistona ja sijoittuu kuvan 1 kolmikerrosrakenteessa sovelluskerrokseen. Se on vapaan lähdekoodin ohjelmointiympäristö, joka mahdollistaa Online-palveluiden, palvelurajapintafunktioiden (engl. Application protocol interface, API) ja tietokonelaitteistojen tiedonsiirron yhdistämisen. Ohjelmistoympäristö koostuu suoritusinfrastruktuuri Node.js:ään perustuvasta selainpohjaisesta editorista. Node-RED editorissa luotua ohjelmaa kutsutaan Flow-ohjelmaksi, jonka voi suorittaa selaimesta yhden napin painalluksella. Flow on mahdollista suorittaa paikallisesti Linux, sekä Windows käyttöjärjestelmillä, sekä erilaisilla tietokonelaitteistoilla, kuten Raspberry Pi ja Arduino. Suoritus on mahdollista myös pilvipalveluun integroituna. [8]

Applikaation perustana toimii flow-based programming (FBD), joka kuvaa toimintaa Node nimisten palikoiden verkkona. Yksittäinen palikka nimetään sen toiminnallisuuden mukaan. Verkon sisällä jokainen Node saa vuorollaan tietovirrasta (engl. flow) dataa. Node käsittelee, ja lähettää viestin eteenpäin. Niitä käytetään selaineditorissa vetämällä kuvan 5 mukaisesti työkalupalkista editointialustaan tai tuomalla JavaScript koodia tekstieditorin kautta. Mahdollinen Noden sisäinen toiminnallisuus rakennetaan käyttäen JavaScript funktioita. [8]



Kuva 4. *Node-RED solmuvalikko ja työpöytä*

Kuvan 5 Inject Node on esimerkki ohjelman käyttämien input-palikoiden toiminnasta. Ne mahdollistavat tiedon hakemisen nettisivuilta, mittalaitteilta, tai muusta tarkoituksellisesta osoitteesta. Input Nodet muodostavat ja lähettävät viestiobjektin (msg), johon viitataan työn edetessä termillä viesti. Viesteillä on olemassa parametrit aihe (engl. topic), sekä hyötykuorma (engl. payload). Haettu tietosähke tallentuu viestin hyötykuormaan. Hyötykuorman tyyppi tallennetaan viestin aiheeksi [8]



Kuva 5. *Esimerkki Node-RED flowsta*

Viestin sisältöä voidaan muokata funktiosolmulla kuvan 5. tapaan yhdistämällä sen sisääntulo inject-solmun ulostuloon. Funktiosolmujen sisälle kirjoitetaan Javascript -funktioita. Alusta sisältää useita valmiita funktioita tietyille toiminnoille. Esimerkiksi XML-tietosähkeiden käsittelylle ja tiedon tallentamiselle löytyvät omat Node palikat kirjastosta. Debug Node tulostaa viestin hyötykuorman debug-ikkunaan työpöydän oikealla puolella. [8]

4. REST-SUUNNITTELUPERIAATTEET

Resurssiorientoituneena arkkitehtuurityylinä Representational State transfer (REST) on erittäin käytännöllinen hajautettujen applikaatioiden rakentamiseen internetin mittakaavassa. [9] REST suunnitteluperiaatteet esitti ensimmäisen kerran Roy Fielding väitöskirjassaan [10] vuonna 2000. Suunnitteluperiaatteet asettavat kuusi vaatimusta, jotka yhdessä toteutettuna tuottavat RESTful palvelun. Richardson ja Ruby esittelevät näiden pohjalta konkreettisen arkkitehtuurin painottaen resurssien roolia mallissa. [11, s. 79]

Resurssi on heidän mukaansa: ”Mikä tahansa tarpeeksi tärkeä asia, johon viittaaminen halutaan tehdä mahdolliseksi”. Määritelmän mukaan jokaisella resurssilla tulee olla oma Uniform Resource Identifier -osoitteensa (URI), joka mahdollistaa resurssiin viittaamisen. URI-osoitteiden ja resurssien kautta määritellään ensimmäinen vaatimus *ositteellisuudelle* (*engl. Addressibility*): Osoitteellisella applikaatiolla on URI-osoite jokaiselle käytettävälle resurssille. [11, s. 84-85]

Toinen vaatimus arkkitehtuurissa on *tilattomuus* (enlg. *Statelessness*). Tilattomuudella tarkoitetaan jokaisen hypertekstin yhteyskäytännön (engl. Hypertext Transfer Protocol, HTTP) pyynnön eristymistä muista pyynnöistä. Edellisten pyyntöjen tietoja ei käytetä, vaan yksi HTTP pyyntö sisältää kaiken tarvittavan tiedon käskyn toteuttamiselle palvelimella. Käyttäjä toteuttaa haluamansa käskyn ja saa vastauksen ilman tarvetta tuntea taustaprosessia. [11]

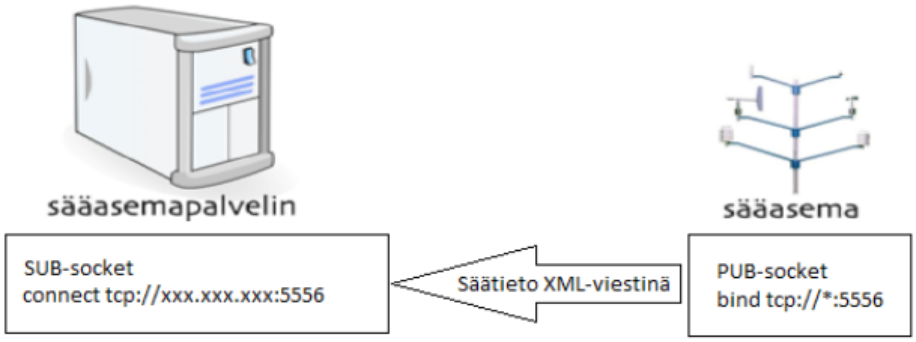
Käyttäjän toteuttamat käskyt tulee olla standardoitu, jotta vaatimus *yhtenäiselle rajapinnalle* (engl. *Uniform Interface*) täyttyy. REST:lle ominaista on käyttää HTTP:n määrittämiä käskyjä: GET, POST, PUT ja DELETE. GET hakee resurssin käyttäjälle. POST luo uuden resurssin olemassa olevaan URI-osoitteeseen. PUT luo uuden URI-osoitteen ja sille resurssin. DELETE poistaa olemassa olevan resurssin. Nämä käskyt muodostavat metadataa käsittelevien käskyjen HEAD ja OPTIONS kanssa yhtenäisen rajapinnan. [11]

5. ROSA -SÄÄASEMA

Vaisala ROSA on Tampereen teknillisen yliopiston omistama tiesääasema. Laitteeseen toteutettu sarjaporttiyhteys mahdollistaa muiden laitteiden liittämisen osaksi sääasemaa. Asemaan on liitetty anturit, jotka mittaavat lämpötilaa, kosteutta, sademäärää, tuulen suuntaa sekä tuulen nopeutta. Näiden lisäksi kokonaisuuteen kuuluu IEI WStrider-200AW-R10/4G teollisuustietokone, sekä WLAN -tukiasema. [12] Lisää tietoa Vaisala ROSA sääasemasta ja sen komponenteista löytyy lähteistä [12] [13] [14].

Sääaseman käyttämä tiedonsiirtoarkkitehtuuri on viestijonotyyppinen ja hyödyntää publish-subscribe mallia esitettynä kuvassa 6. Rosa-sääasemaan liitetty teollisuustietokone toteuttaa mallin hyödyntäen avoimen lähdekoodin viestinvälityskirjastoa ZeroMQ. [14] Tämä otetaan huomioon viestejä vastaanotettaessa.

Virtuaalinen Publish-pistoke (PUB) on lisätty TCP-porttiin. Viestejä vastaanottavien Subscribe-pistokkeiden (SUB) IP-osoitteita ei rajoiteta. Tämä tarkoittaa, että mikä tahansa samassa verkossa toimiva laite voi vastaanottaa viestejä. Sääasema kommunikoi PC:n kanssa sarjaväylän kautta 60 sekunnin välein. Asema tuottaa Extensible Markup Language viestin (XML) ja PC julkaisee sen Publish-pistokkeessa. PC sallii useiden Subscribe-pistokkeiden liittymisen ja poistumisen, ja viesti voidaan jakaa kaikkien vastaanottajien kesken vaikuttamatta lähettäjään. [14]



Kuva 6. *Sääaseman kommunikointi palvelimelle [14]*

6. SOVELLUKSEN TOTEUTUS

Työssä käytetään sovelluksen toteuttamiseen luvussa 3 esiteltyä Node-RED kehitysympäristöä. Sovelluksen tarkoituksena on toimia koulun yksityisen pilvipalvelun osana. Sovellus näyttää käyttäjälle reaaliaikaisen datan sääaseman sensoreilta, tallentaa datapaketit tietokantaan, sekä antaa mahdollisuuden hakea säätietoja tietyltä aikaväliltä. Näihin tavoitteisiin vastataan Node-RED alustalle rakennetulla käyttöliittymällä, jossa on esitettynä reaaliaikainen data, sekä RESTful API funktiolla, joka mahdollistaa tietojen hakemisen tietokannasta.

Avoin lähdekoodi ja omien flow tiedostojen helppo jakaminen antavat Node-RED yhteisölle mahdollisuuden kehittää omia kirjastoja ohjelmiston tominnallisuuksien täydentämiseksi. Kirjasto löytyy ohjelmiston nettisivuilta, sekä editorin *manage palette* –osiosta. Kehittäjät voivat luoda kustomoituja Nodeja tai kokonaisia flow-toteutuksia. [8]

Työssä käytetään kolmea kontribuutiokirjastoa. Käytetyt kirjastot ovat laajennuksia mongoDB, ZeroMQ, sekä UI-builder toimintojen käyttämiselle. MongoDB on tietokantasovellus. ZeroMQ tarjoaa protokollien yhteensopivuuden tiedon vastaanottamiselle. UI-builder mahdollistaa räätälöidyn käyttöliittymäsuunnittelun flow editorissa.

6.1 Tiedonsiirto Node-RED-sovellukseen

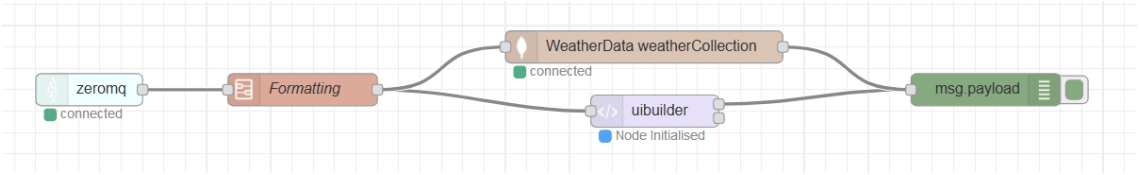
Tiedonsiirto sääaseman ja sen välisen teollisuustietokoneen välillä tapahtuu luvun 5 mukaisesti. Koska Publish-pistoke julkaisee tiedon käyttäen ZeroMQ viestinvälityskirjastoa, tiedon tilaamista varten tarvitaan kontribuutiokirjasto, joka tarjoaa käyttöön ZeroMQ Nodet.

Alkuperäisen XML-viestin siirto Node-RED virtaan tapahtuu luomalla ZeroMQ SUB-pistoke käyttöliittymään. Koska sääasema toimii samassa verkossa koulun pilven kanssa, tiedonsiirto on mahdollista TCP protokollan avulla.

6.2 Tiedon käsittely ja tallentaminen

Tieto sääasemalta tulee XML-viestinä, joka on paketoitu tarpeettomasti Simple object access protocol (SOAP) muotoon. Input Nodelta saatu viesti sisältää aiheenaan koko SOAP viestin merkkijonona. Jotta Node-RED alustan toiminnallisuus saadaan hyödynnettyä, on tarkoituksenmukaista saattaa viesti käyttökelpoiseen ja tallennettavaan muotoon. Muotoilua varten luodaan Node -ryhmä, joka muodostaa alivirran *Formatting.* Alivirralla tarkoitetaan usean Noden ryhmää, joka pääohjelmassa on paketoitu yhden Noden sisään.

Formatting sisältää funktion, joka ensin leikkaa merkkijonosta ohjelman ja datan kannalta oleellisimmat Channel osion sisältämät tiedot ja tallettaa ne viestin hyötykuormaksi. Tämän jälkeen XML-muotoinen merkkijono muutetaan XML Nodella JavaScript objektiksi. Objektista poistetaan Format-solmulla turha Name -kenttä ja lisätään aikaleimat. Toinen aikaleimoista mahdollistaa aikojen vertailun, ja toinen on helposti luettavassa muodossa. Kokonaisuudessaan prosessin flow on kuvattuna kuvassa 7.



Kuva 7. *Tietojen haku, tallennus ja syöttäminen käyttöliittymään*

Muotoilun tuloksena saadaan kuvan 8 mukainen JSON-objekti.

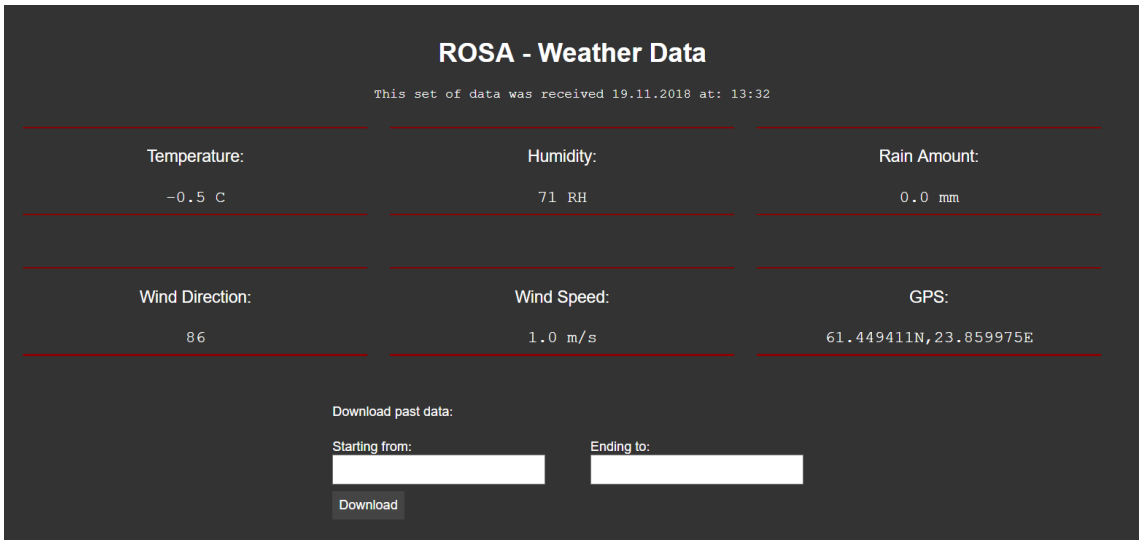
```
var jsonMsg = {
  "timeinms":md,
  "timestamp":timestamp,
  "temperature":temperature,
  "humidity":humidity,
  "windspeed":windSpeed,
  "winddirection":windDirection,
  "rainamount":rainAmount,
  "GPSinformation":GPS
};
```

Kuva 8. *JSON-objektin rakenne*

Objekti tallennetaan MongoDB tietokantaan. MongoDB on vapaan lähdekoodin NoSQL tietokanta. Tietokanta tallentaa dokumentit JSON tyylisissä paketeissa. Tietoja haetaan kyselyillä. [15]

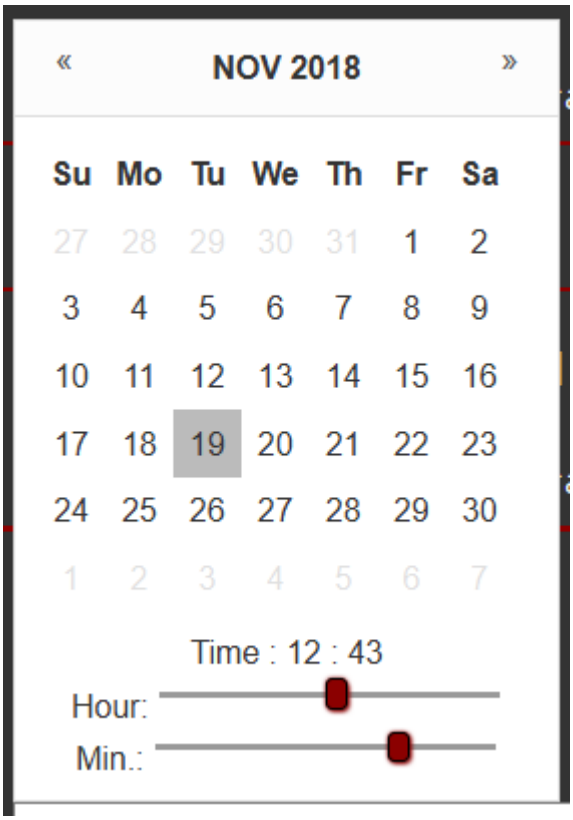
6.3 Käyttöliittymä

Sääasemalta saatu viestiobjekti muutetaan visuaaliseksi kontribuutiokirjaston tarjoamalla uibuilder Nodella. Se tarjoaa mahdollisuuden kustomoida käyttöliittymä käyttäen html, sekä javascript työkaluja. Sensoridata päivittyy käyttöliittymään heti, kun viesti kulkee uibuilder solmulle. Viestin tiedot esitetään käyttäjälle käyttöliittymässä (kuva 9.).



Kuva 9. *Käyttöliittymä*

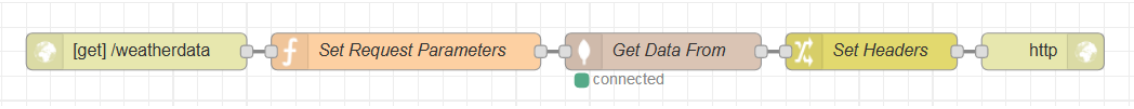
Käyttöliittymästä näkee säätietojen lisäksi päivitetyn tietopaketin aikaleiman. Kukin säätieto on omana laatikkonaan, joka sisältää otsikon, sekä päivittyvän tietokentän. Säätietojen alla käyttöliittymässä on mahdollisuus ladata tekstitiedostona tallennettua säädataa. Aikaväli valitaan klikkaamalla ensin kuvassa 9 alavasemmalla olevasta kentästä kuvan 10 mukainen valintaikkuna auki. Valintaikkunasta on mahdollista säätää aika minuutilleen. Samaan tapaan valitaan välin päättymisaika seuraavasta kentästä.



Kuva 10. *Aikavalintaikkuna*

6.4 Säätietojen hakeminen

Säätietoja haetaan käyttämällä luvussa 4 esitettyjä REST –suunnitteluperiaatteita noudattavaa palvelurajapintafunktiota. Funktiota käytetään kahdella valintaikkunalla sekä latausnappulalla. Latausnappulaa painettaessa sivu lähettää pyynnön, jonka Uniform resource locator -osoitteeseen (URL) sijoitetaan päivämäärät parametreiksi. Funktion toiminta on esitetty kuvassa 11. Käyttäjän lähettämä GET käsky muuntuu flow-ohjelmassa MongoDB tietokantaan lähteväksi FIND pyynnöksi. Pyyntöön liitetään GET käskyn parametrit. Node-RED tulkitsee parametrit ja hakee tietokannasta päivämäärien mukaiset tietopaketit. HTTP-vastauksena käyttäjä saa ladattavan tekstitiedoston, joka sisältää kyseisen aikavälin säätiedot JSON syntaksina.



Kuva 11. *API-funkntion toteutus flow editorissa*

Jokaiselle säätietopaketille annetaan aikaleiman lisäksi myös vertailtava luku. Luku kuvaa viestin ajanhetken millisekunteina vuoden 1970 alusta. Näin tietokannasta voidaan

hakea oikeat säätiedot. Päivämäärän muuntaminen on funktio selaineditorissa, eikä käyttäjä puutu sen toimintaan. Toiminto tapahtuu Kuvan 11 funktiolohkossa *Set Request Parameters*.

MongoDB antaa jokaiselle parametrille identifikaationumeron. Numero toimii resurssin URI-osoitteena. Tietokannan rakenteen ansiosta myös resurssin yksittäisiin tietolohkoihin voidaan viitata. Tämä mahdollistaa useanlaisten tietopakettimuotojen käyttämisen palautuksessa.

Käyttäjä näkee käyttöliittymässä vain kuvassa 9 esitetyt tiedot. Käyttäjälle annetaan ainoastaan mahdollisuus valita aikaväli, sekä painaa latausnappulaa. Käyttäjä ei pysty muokkaamaan tietokannan sisältöä. HTTP-pyyntö toteutetaan kokonaisuudessaan, kun käyttäjä painaa Download nappulaa. Näillä ehdoilla voidaan todeta palvelun toteuttavan luvun 4 REST suunnitteluperiaatteet.

7. YHTEENVETO

Työssä tutkittiin esineiden internetin yleistä rakennetta ja sen perusteella muodostettua sovellusarkkitehtuuria käytännön esimerkin avulla. Työn esimerkin puitteissa mietittiin kiinteiden laitteistojen, ohjelmistojen ja niiden välisen tiedonsiirron menetelmiä. Kiinteänä laitteistona toimii Vaisala ROSA tiesääasema.

Esineiden internetin rakennetta mietittiin aluksi kerrosrakenteen kautta. Sen jälkeen tarkennettiin rakenne laajennetun pilvipalvelumallin SOA pohjalta. Tärkeä osa-alue kerrosrakenteessa on kerrosten välinen tiedonsiirto ja sen turvallisuus. Olennaista on myös järjestelmän toiminnan abstrahointi lopputuotteen käyttäjältä.

Näihin vaatimuksiin vastattiin avoimen Node-RED-ohjelmistoympäristön sekä REST suunnitteluperiaatteiden avulla. Työssä on toteutettu vaatimuksen mukaisesti Node-RED-ohjelmiston avulla käyttöliittymä tietojen reaaliaikaista tarkastelua varten. Käyttöliittymään tehtiin myös toteutus säätietoja tietokannasta hakevalle API-funktiolle. Säätietoja tallennetaan MongoDB tietokantaan.

Työssä kiinnitettiin erityisesti huomiota Node-RED-ympäristön helppokäyttöisyyteen. Se on kattava työkalu esineiden internetin palveluiden luomiseen. Laitteistojen tiedonsiirron yhdistäminen internetiin on helppoa ja intuitiivista. Node-RED ja sen kontribuutiokirjastot tarjoavat helpon ratkaisun työn tietokannan toteuttamiseen. MongoDB tietokantaa käsittelevät Nodet ovat integroitavissa kontribuutiokirjaston avulla muutamalla klikkauksella.

Node-RED mahdollistaa käyttöliittymien räätälöimisen käyttäjän tarpeisiin. Kehittäjä voi luoda käyttöliittymän valmiista osista dashboard kontribuutiokirjastolla tai luoda sen alkutekijöistä asti uibuilder Nodella.

Kontribuutiokirjastot ovat ratkaiseva osa Node-RED-ohjelmiston kykyä antaa kehittäjälle standardoitu kehitysympäristö. Avoin lähdekoodi, ja yhteisön mahdollisuus laajentaa ohjelmiston kirjastoja ovat tehokas keino uusien ja kattavien ratkaisujen mahdollistamiseksi kehitysympäristössä.

LÄHTEET

[1] L. Atzori, A. Iera, & G. Morabito, The Internet of Things: A survey, Computer Networks, Elsevier B.V, in 2010, Vol. 54(15), pp. 2787-2805. Available: https://www.sciencedirect.com/science/article/pii/S1389128610001568

[2] O. Mazhelis, E. Luoma and H. Warma, "Defining an internet-of-things ecosystem," 2012, DOI: 10.1007/978-3-642-32686-8_1.

[3] Jie Lin *et al*, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," IEEE Internet of things journal (*JIoT),* vol. 4, *(5),* pp. 1125-1142, 2017. Available: https://ieeexplore.ieee.org/document/7879243. DOI: 10.1109/JIOT.2017.2683200.

[4] P. Heino: "Pilvipalvelut", Talentum Media Oy ja Petteri Heino, 2010.

[5] S. Andreev et al. (Eds.): "Internet Of Things, Smart Spaces, and Next Generation Networking", NEW2AN/ruSMART, in 2012, LNCS 7469, pp. 429–439

[6] K. Holger, A. Willig, "Protocols and Architectures for Wireless Sensor Networks", John wiley & Sons LTD, 2005

[7] G. Hunt: "TCP/IP: verkonhallinta", O'Reilly, 1998

[8] Node-RED, [WWW] , https://nodered.org/, [Viitattu 27.9.2018]

[9] B. Cheng *et al*, "Lightweight Service Mashup Middleware With REST Style Architecture for IoT Applications," *T-Nsm,* vol. 15, *(3),* pp. 1063-1075, 2018. Available: https://ieeexplore.ieee.org/document/8340176. DOI: 10.1109/TNSM.2018.2827933.

[10] R. Fielding and R. Taylor, "Principled design of the modern web architecture," Information and computer science, California university, in Jun 1, 2000, Available: http://dl.acm.org/citation.cfm?id=337228. DOI: 10.1145/337180.337228.

[11] L. Richardson and S. Ruby, "RESTful Web Services", O'Reilly, 2007

[12] H. Mäenpää: "IP-pohjainen langaton liityntä Vaisala ROSA-tiesääasemaan", kandidaatintyö, Tampereen teknillinen yliopisto, 2011.

[13] M. Salonen: "Vaisala ROSA-sääaseman etäkäyttö Telit GE863 G moduulin avulla", erikoistyö, TTY Automaatio- ja säätötekniikan laitos, 2009.

[14] O. Teikari: “Systeemitekniikan laitoksen sääaseman tietoliikenteen muuttaminen ZeroMQ pohjaiseksi”, kandidaatintyö, Tampereen teknillinen yliopisto, 2013

[15] MongoDB, [WWW], (Viitattu 21.11.2018), https://docs.mongodb.com/