5G TESTIVERKON IOT-SENSORIEN TUOTTAMAN DATAN HALLINOINTI JA KÄYTTÖ

Atso Simojoki

Ohjaaja: Olli Liinamaa

ELEKTRONIIKAN JA TIETOLIIKENNETEKNIIKAN TUTKINTO-OHJELMA
2019

Simojoki A. (2019) 5G testiverkon IoT-sensorien tuottaman datan hallinnointi ja käyttö. Oulun yliopisto, Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Kandidaatintyö, 28 s.

TIIVISTELMÄ

Työn tarkoituksena oli käydä läpi ja kartoittaa Oulun yliopiston 5G testiverkkoon integroitujen Internet of Things (IoT) -sensorien toiminta sekä niiden tuottaman datan käyttö ja mahdollisuudet.

Työssä käydään läpi järjestelmän eri osa-alueet mistä järjestelmä koostuu. Sensorit keräävät dataa ympäristöstä, joka tallennetaan MySQL-tietokantaan, jonka jälkeen se tulee käsitellä ja analysoida jotta data muuttuu tiedoksi ja sitä pystytään hyödyntämään.

Avainsanat: 5G, IoT, Tietokanta.

Simojoki A. (2019) Management and Use of Data Produced by IoT-sensors in 5G Test Network. University of Oulu, Degree Programme in Electronics and Communications Engineering. Bachelor's Thesis, 28 p.

ABSTRACT

The goal in this thesis was to introduce and examine the operation of Internet of Things (IoT) sensors integrated in the 5G test network at University of Oulu.

In this thesis is covered systems different parts and what the system contains. Sensors collect data from environment which is saved to MySQL database. After this the data needs to be dealt with and analyzed so that the data is transformed into knowledge and can be utilized.

Key words: 5G, IoT, Database.

SISÄLLYS

ALKULAUSE

Haluan kiittää Olli Liinamaata mahdollisuudesta tutustua ja tutkia Oulun Yliopiston kehittämää 5G testiverkkoa ja sen tarjoamia tulevaisuuden mahdollisuuksia. Lisäksi haluan kiittää Arto Matilaista hänen avustaan työn suhteen.

Oulussa 24.11.2019

Atso Simojoki

LYHENTEIDEN JA MERKKIEN SELITYKSET

5G	5th Generation Mobile Network
5GTN	5G Test Network
BT LE	Bluetooth Low Energy
CSV	Comma-separated Values
CWC	Centre for Wireless Communications
HTML	Hypertext Markup Language
IoT	Internet of Things
JSON	JavaScript Object Notation
NB-IoT	Narrow Band
SSH	Secure Shell
UFW	Uncomplicated Firewall

1. JOHDANTO

Oulun Yliopistolla sijaitsee viidennen sukupolven matkapuhelinverkon (5G) testiverkko (5GTN). Testiverkko tarjoaa mahdollisuuden 5G teknologian, komponenttien ja palveluiden testaamiseen. [1] Yksi 5G teknologian mukana tulevista teknologioista on Internet of Things (IoT) eli esineiden internet. [2] Yliopiston testiverkossa on mahdollisuus omien IoT-sensorien testaamiseen ja kehittämiseen.

Työssä käydään läpi Yliopiston testiverkon toimintaa. Sen lisäksi esitellään järjestelmän rakenne ja kiinnitetään erityisesti huomiota IoT-sensorien käyttömahdollisuuksiin.

Työssä esitellään erilaisia työkaluja sensorien keräämän data esittämiseen ja analysointiin. Datan käyttömahdollisuudet tulevat esiin vasta kun se saadaan muokattua hyödylliseksi tiedoksi.

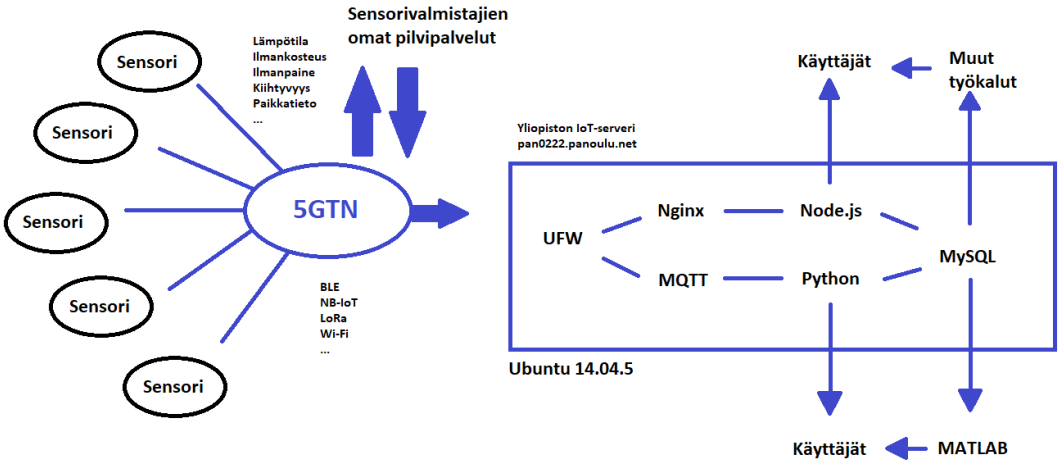
Lopuksi käydään läpi käyttö- ja kehittämismahdollisuuksia. IoT-teknologia mahdollistaa tulevaisuudessa älykään ympäristön rakentamista.

Suurin osa sensorivalmistajista tarjoaa oman pilvipalvelun sensoriensa datan tallennusta ja analysointia varten. Kuluttajalle tämä on suotuinen vaihtoehto koska se vähentää huomattavasti oman vaivan määrää datan varastoinnin ja käytön suhteen. Yrityksillä tilanne ei kuitenkaan ole sama. Sensorit usein keräävät data yksityisistä tiloista ja niiden keräämä data saattaa olla arkaluonteista eikä sen haluta päätyvän muiden käsiin. tällöin Yliopiston kehittämä ratkaisu on parempi vaihtoehto. Siinä kaikkien sensorien data tallennetaan samaan tietokantaan. Tällöin kaikki data löytyy samasta paikkaa, jonka lisäksi data ei päädy muiden käsiin. Laitevalmistajasta riippumatta kaikki data olisi samassa tietokannassa toisin kuin silloin jos laitevalmistajien pilvipalveluita käytettäisiin, jolloin jokaisen laitevalmistajan oman sensorin data menisi kyseisen valmistajan pilvipalveluun.

2. JÄRJESTELMÄN RAKENNE JA YMMÄRTÄMINEN

2.1. Järjestelmän osat

IoT-sensorit ovat osa 5G testiverkkoa. Sensorien keräämä data siirtyy erilaisilla tiedonsiirtotekniikoilla testiverkkoon, josta se tallentuu Yliopistolla sijaitsevaan MySQL-tietokantaan. Kuvassa 1 on esitelty järjestelmän osat pintapuolisesti.



Kuva 1. Järjestelmän rakenne

2.1.1. Sensorit

Järjestelmästä löytyy usean laitevalmistajan kehittämiä IoT-sensoreita. Sensorit ovat ketjussa niin sanottu ensimmäinen lenkki. Jokainen sensori on suunniteltu keräämään tietoa tietyistä ympäristön arvoista kuten lämpötilasta tai ilmanpaineesta. Jokainen IoT-sensori sisältää yhden tai useamman sensorin arvojen mittausta varten. Yhdessä sensorit luovat kokonaisuuden ympäristöstä.

Järjestelmän sensorit voidaan jakaa kahteen yläluokkaan. Ensimmäisen luokan sensorit ovat puhtaasti integroitu yliopiston järjestelmään ja niiden data tallentuu Yliopiston omaan MySQL-tietokantaan. Toinen yläluokka on sensorit, joiden data kulkee samaa reittiä muiden sensorien kanssa 5G testiverkkoon asti, josta data siirtyy laitevalmistajan omaan pilvipalveluun, johon sensorin data tallentuu.

2.1.2 5GTN

Yliopiston 5G testiverkko (5GTN) on innovaatioalusta seuraavan sukupolven palveluille kuten puhelimille, tableteille ja sensoreille. Testiverkko mahdollistaa erilaisilla tiedonsiirtomenetelmillä toimivien sensorien integroinnin yhdeksi kokonaisuudeksi. Tieto siirtyy sensoreista erilaisten lisensoimattomien vähäenergisten tiedonsiirtomenetelmien avulla testiverkon kautta tietokantaan.

Bluetooth Low Energy sekä Narrow Band teknologiat ovat hyviä vaihtoehtoja sensorien tiedonsiirtoon niiden pienen virrankulutuksen ansiosta.

2.1.3 MySQL tietokanta

Sensorien tuottaman datan tallentamista varten on luotu MySQL-tietokanta, johon data tallentuu. Tietokanta toimii rajapintana sensorien ja käyttäjän välillä. Tietokantaa ylläpidetään Linux-palvelimella, jonka toimintaan käydään läpi kolmannessa kappaleessa.

Palvelimelta löytyy Node.js JavaScript run-time ympäristö, joka mahdollistaa JavaScriptillä luotujen ohjelmien suorittamisen palvelimella. Palvelimelta löytyy lisäksi Python 2.7 tulkki python-ohjelmien suorittamisen.

3. KÄYTTÖ

Sensorien kaikki data löytyy tallentuneena aikaisemmin mainittuun tietokantaan. Nyt data haetaan tietokannasta ja käsitellään eri työkalujen avulla koska data muuttuu tiedoksi vasta kun siitä tehdään älykkäitä päätelmiä.

Alkuun käydään läpi, miten tietokantaan päästään käsiksi, jonka jälkeen dataa analysoidaan ja visualisoidaan. Tässä kappaleessa käydään läpi kaksi esimerkkiä datan käyttöön. Ensimmäisessä esimerkissä tietokannan sisältämä data visualisoidaan käyttäen apuna JavaScriptiä ja HTML-koodia. Tämä esimerkki visualisoi datan mutta ei kerro vielä kaikkea sen luonteesta.

Toisessa esimerkissä data haetaan tietokannasta manuaalisesti tiedostona, jota analysoidaan MATLAB-ympäristössä.

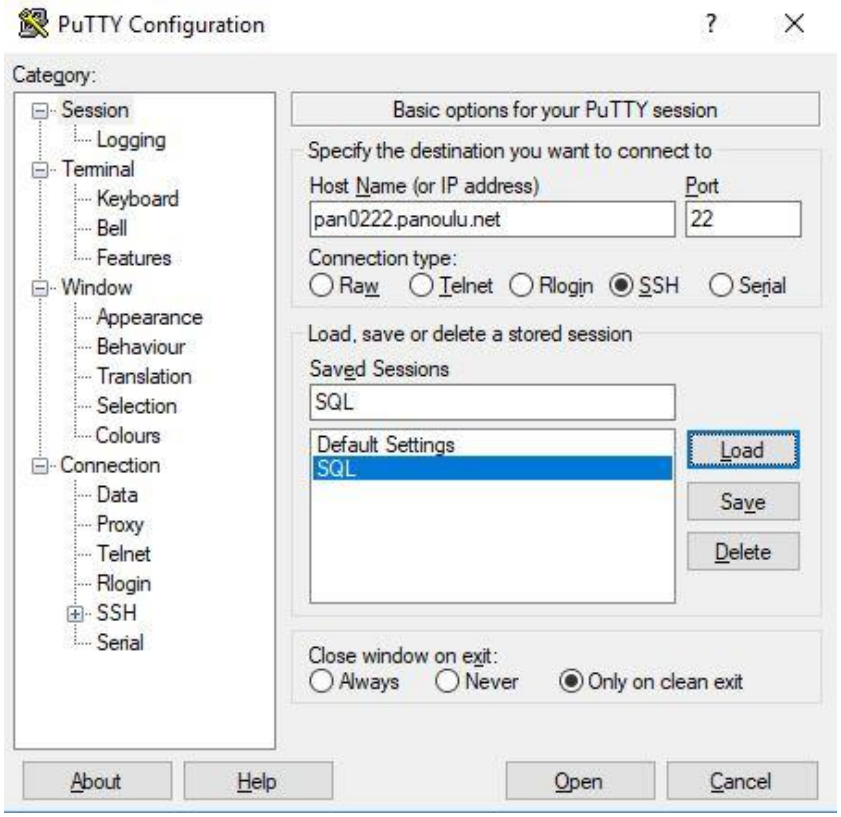
3.1. Yhdistäminen MySQL-tietokantaan

Tässä esimerkissä käytetään hyödyksi kahta ohjelmaa, joiden avulla päästään tietokantaan käsiksi. PuTTY:a jota käytetään yhteyden luomiseen palvelimelle jonka jälkeen HeidiSQL-ohjelman avulla avataan itse tietokanta.

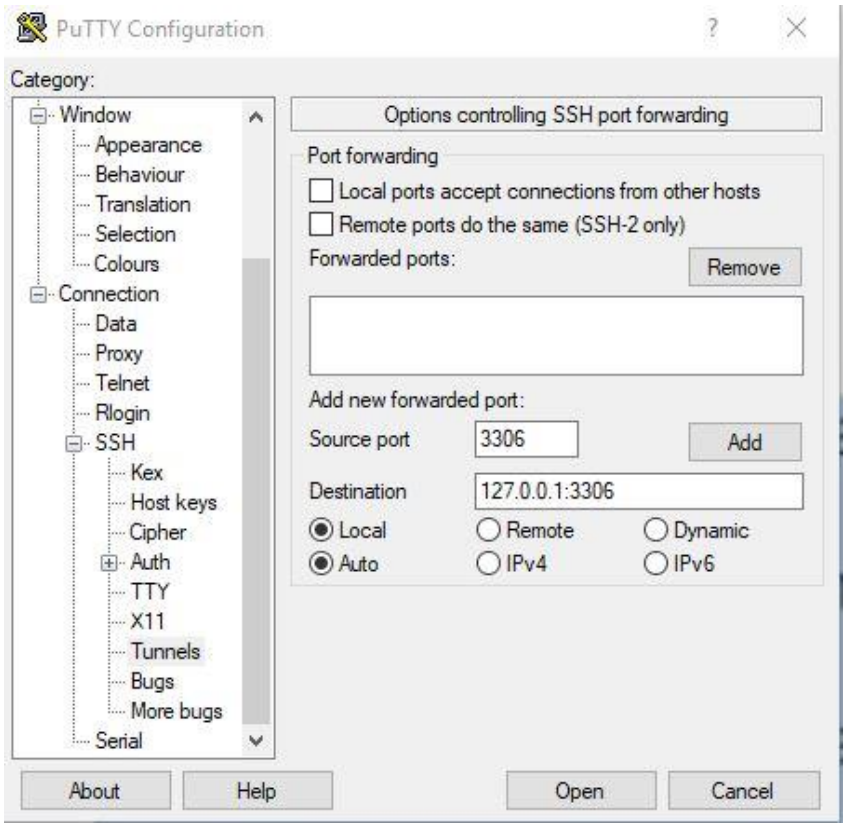
PuTTY:llä luodaan yhteys Oulun yliopistolla sijaitsevaa palvelimeen, josta tietokanta löytyy. Isäntäosoite on pan0222.panoulu.net ja portin numero 22 kuten kuvasta 2 näkyy. Kun isäntäosoite ja portti on asetettu, tulee määritellä vielä tunnelointi. Tunnelointiin pääsee PuTTY:n vasemmassa laidassa olevasta listasta.

Listasta valitaan SSH, josta valitaan Tunnels. Nyt ikkunaan tulee syöttää määränpää sekä lähdeportti. Määränpää on tässä tapauksessa 127.0.0.1:3306 ja lähdeportti 3306 kuten kuvasta 3 näkyy. Lopuksi painetaan *ADD*-painiketta, joka syöttää ylläolevaan kenttään syötetyt tiedot ja avataan yhteys painamalla *Open*. Tämä aukaisee PuTTY:n terminaalin, johon tulee syöttää oma käyttäjätunnus sekä salasana palvelimelle.

Kun yhteys on luotu, Käynnistetään tietokantaa hallinnoiva ohjelma, joka tässä esimerkissä on HeidiSQL.

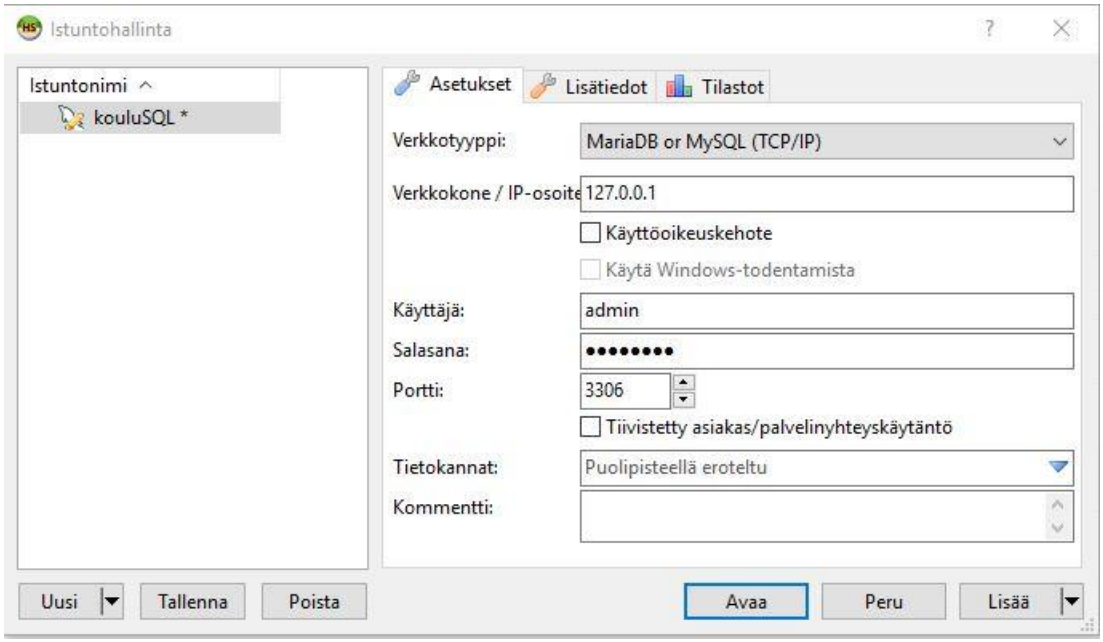


Kuva 2. PuTTY:n käyttöliittymä



Kuva 3. SSH tunnelointi

Kuvassa 4 näkyy HeidiSQL:n käyttöliittymä, johon tulee syöttää palvelimen tiedot. Tässä tapauksessa IP-osoite on 127.0.0.1 ja portin numero 3306. Käyttäjä ja salasana kohtiin tulee syöttää omat tunnuksensa palvelimelle ja verkon tyypiksi tulee valita MySQL (TCP/IP). Lopuksi painetaan *Avaa*-painiketta, jonka jälkeen tietokanta aukeaa.



Kuva 4. HeidiSQL käyttöliittymä

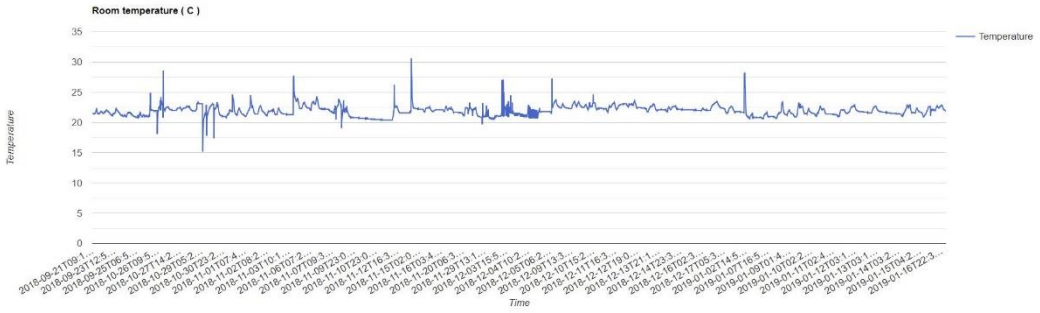
3.2. Datan visualisointi

Järjestelmään on luotu valmiita työkaluja, joiden avulla dataa pystytään käsittelemään. Tässä esimerkissä käytetään JavaScriptiä web_queries.js joka löytyy liitteestä 1. Koodin tulee olla palvelimella käyttäjän omassa profiilissa, jotta se pystytään suorittamaan palvelimella.

Koodissa määritellään mistä tietokannasta sekä pöydästä (table) dataa halutaan hakea. Nämä tiedot voidaan muuttaa koodiin, jos tietoa halutaan hakea esimerkiksi toisen sensorin tietokannasta. Liitteen 1. esimerkkikoodissa tietokannaksi on määritelty 5gtn_iot_demo_db, jonka pöydästä 5gtn_gps_test_db data haetaan.

Kun palvelin halutaan suorittamaan kyseistä koodia, kirjoitetaan PuTTY:n terminaaliin komento *node web_queries.js* jolloin palvelin alkaa kuuntelemaan koodissa määriteltyä porttia joka tässä esimerkkitapauksessa on 2998.

Nyt data pystytään visualisoimaan. Käytetään seuraavaksi esimerkkiohjelmaa temperaturetest.html joka löytyy liitteestä 2. Liite tulee tallentaa HTML-muodossa, jolloin sen klikkaaminen aukaisee selaimeen uuden sivun, johon piirtyy kuvaaja halutun pöydän datasta. Kuvasta 5 löytyy esimerkkikoodin piirtämä kuvaaja sensorin mittaamasta lämpötilasta ajan funktiona.



Kuva 5. Sensorin mittaaman lämpötilan arvo ajan funktiona

Esimerkkikoodi on suunniteltu siten, että HTML-koodi hakee tietokannasta sensorin mittaaman lämpötilan sekä lämpötilaa vastaavan ajanhetken. Näiden tietojen pohjalta koodi piirtää selaimeen sensorin mittaaman lämpötilan ajan funktiona.

Koodi toimii toteutuspohjana ja sitä on helppo muokata omiin käyttötarkoituksiin sopivaksi. Perusperiaate on, että JavaScript lähettää koodissa määritellystä tietokannasta dataa. Pöydästä riippumatta koodin pystyy muokkaamaan siten, että se hakee juuri halutusta datasta. HTML-koodissa JavaScriptiltä pyydetään tiettyä dataa, joka selaimeen piirtyy. Jos esimerkiksi halutaan hakea ilmapainetta tietystä pöydästä, tulee JavaScriptiin määritellä tietokanta sekä pöytä, josta tieto halutaan hakea. Tämän jälkeen HTML tulee muokata siten, että se kysyy oikeita tietoja. Lopuksi selvyyden vuoksi kannattaa muokata myös muutamia nimiä HTML-koodissa, ettei ilmanpainekuvaajan nimeksi tule esimerkiksi Room Temperature.

Esimerkkikoodin rivillä 45 on määritelty tietokanta, josta haluttu data haetaan. Kuvassa 6 tietokannaksi on määritelty 5gtn_iprotoxi_db. Tätä riviä muokkaamalla pystytään vaihtamaan haluttua tietokantaa.

```
var pool = mysql.createPool ({
    connectionLimit: 30,
        host: "localhost",
        user: "admin",
        password: "5gtnadmin",
        database: "5gtn_iprotoxi_db"
}); // mysql.createPool
```

Kuva 6. Tietokannan määrittelevä rivi koodista

Rivillä 74 on määritelty mitä tietoa tietokannasta haetaan. Kuvan 7 tapauksessa aikaisemmin määritellyn tietokannan pöydästä 5gtn_iprotoxi_test_db haetaan lämpötilan arvo (temperature) sekä niitä vastaavat ajanhetket (td). Näitä tietoja HTML-koodi käyttää kuvaajan luomiseen. Mikäli pöytää tai mitattua arvoa halutaan vaihtaa, voi sen toteuttaa muokkaamalla tätä kyseistä riviä.

```
if (qParam == 1)
    query_str = 'SELECT date_format(created_at,"%Y-%m-%dT%T") AS td, sensor_id AS id, temperature as tmp FROM 5gtn_iprotoxi_test_db';
```

Kuva 7. Haettavan datan määrittelevä rivi koodissa

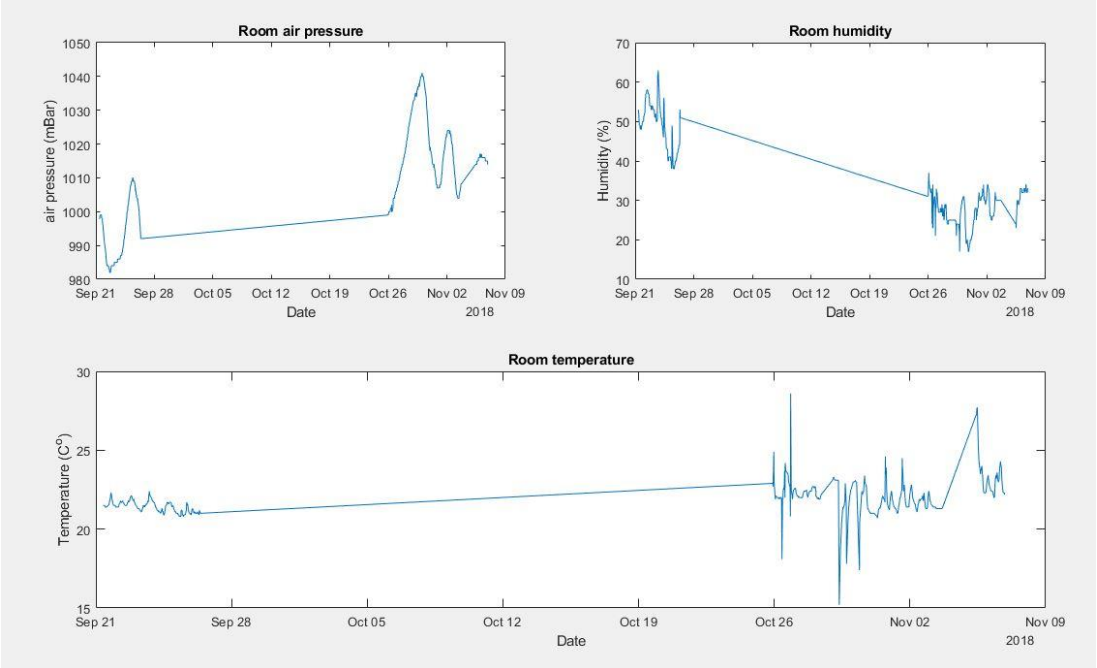
3.3. Datan tarkempi analysointi

Sensoridatan esittäminen kuvaajalla on helposti ymmärrettävä ja yleisin tapa tiedon esittämiseen. Dataan sisältyy kuitenkin myös paljon muuta. Lämpötilasta saattaa esimerkiksi olla tärkeää tietää lämpötilan vaihteluväli. Tässä esimerkissä analysoidaan tarkemmin sensoridataa ja sitä mitä se pitää sisällään.

3.3.1 Datan analysointi MATLAB-ympäristössä

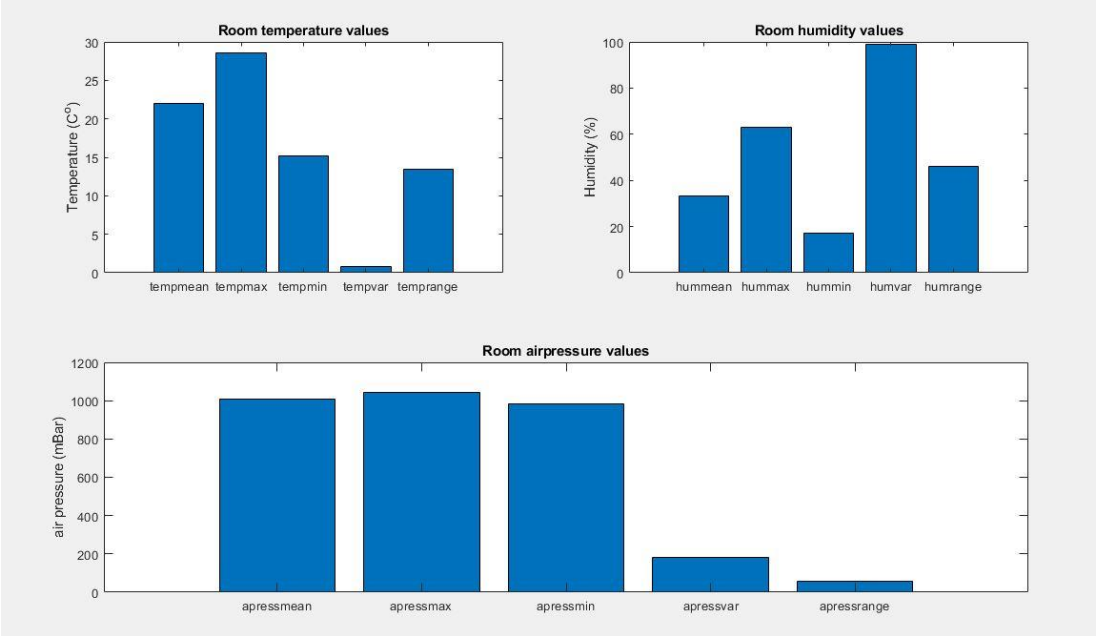
Helpoin tapa datan analysointiin paikallisesti on tuoda data tietokannasta tiedostona omalle tietokoneelle. Tämä onnistuu HeidiSQL-ohjelman avulla. Etsitään tietokanta, joka tietokoneelle halutaan tuoda, jonka jälkeen valitaan yläpalkista Työkalu → Vie vasteririvit. Tässä kohtaan pystytään valitsemaan mihin data tuodaan ja missä muodossa. Lopuksi painetaan valmis.

Esimerkin data tuotiin CSV-tiedostona. CSV-tiedostomuotoa on helppoa lukea ja useat ohjelmat sisältävät valmiita työkalujen sen parsimiseen. Kuvasta 8 löytyy tietokannasta tuodun sensoridatan mittaukset lämpötilasta, ilmakosteudesta sekä ilmanpaineesta. Kuvaajien keskellä olevat viivat johtuvat siitä, että sensori on ollut sammuneena kuukauden ajan. Esimerkissä käytetty MATLAB-koodi löytyy liitteestä 3.



Kuva 8. Sensoridatan visualisointi MATLAB-ympäristössä

Data sisältää paljon muutakin tietoa kuin lämpötilan tai ilmanpaineen tietyllä ajanhetkellä. Kun dataa on riittävästi, voidaan siitä selvittää esimerkiksi lämpötilan vaihtelu tai varianssi eli tutkia kuinka paljon arvot poikkeavat oletusarvosta. Kuvassa 9 on selvitetty tarkempaa tietoa samasta datasta mistä aikaisemmat kuvaajat luotiin. Kuvasta löytyy mittausarvojen keskiarvo, maksimi, minimi, varianssi sekä vaihteluväli.



Kuva 9. Datan tarkempi analysointi

Tietyssä tilanteessa tämä voi olla tärkeää tietoa. Kuvaajasta nähdään, että lämpötila pysyy 20 asteen tuntumassa lähes koko ajan. Tilanteen mukaan voi kuitenkin olla, että lämpötila ei esimerkiksi saa missään nimessä käydä alle 10 asteen, jolloin datan tarkempi analysointi on tarpeen.

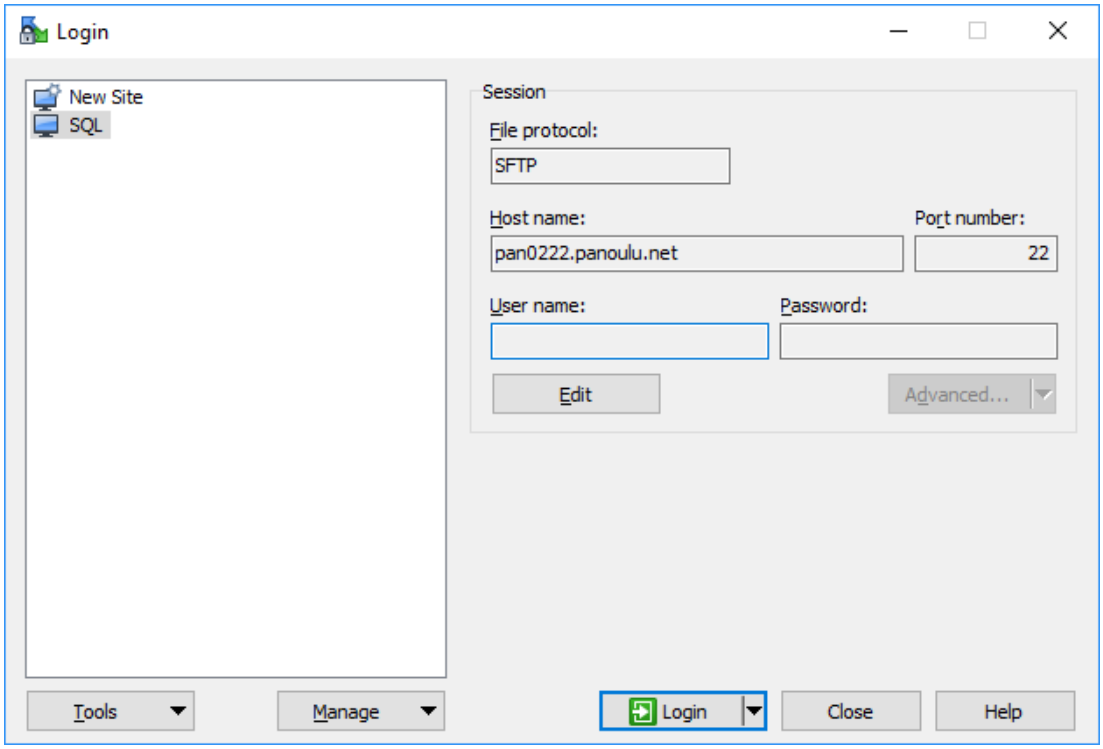
3.4. Ongelmatilanteet

3.4.1. WinSCP

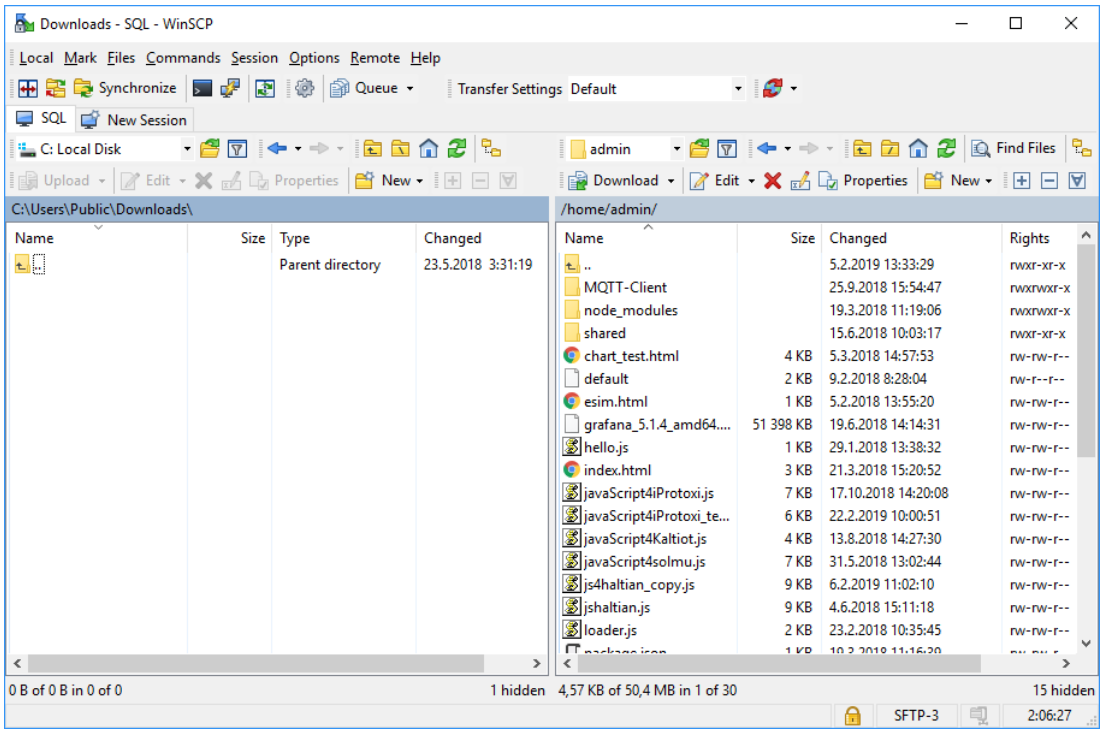
Mikäli palvelin ei suostu käynnistämään haluttua ohjelmaa niin saattaa olla, että kyseistä ohjelmaa ei löydy käyttäjän profiilista. Tämä voidaan tarkistaa syöttämällä komento *ls* PuTTY:n terminaaliin, joka tulostaa listan kaikista tiedostoista käyttäjän profiilissa.

Haluttu ohjelma voidaan siirtää palvelimelle käyttäjän profiiliin manuaalisesti WinSCP:n avulla. WinSCP on ilmainen ohjelma, jolla tietokoneelta voidaan lisätä paikallisia tiedostoja palvelimelle sekä tuoda tiedostoja palvelimelta.

Alkuun WinSCP tulee yhdistää palvelimeen syöttämällä isäntäosoite sekä portin numero, jotka näkyvät kuvassa 10. Tämän jälkeen painetaan *Login*-painiketta, jonka jälkeen ohjelma kysyy käyttäjätunnusta ja salasanaa palvelimelle. Kun nämä on syötetty, aukeaa kuvan 11 mukainen ikkuna, jossa voidaan hakea haluttu tiedosto tietokoneelta ja siirtää se käyttäjän profiiliin palvelimelle.



Kuva 10. WinSCP:n sisäänkirjautuminen



Kuva 11. WinSCP:n käyttöliittymä

3.4.2 Kirjastojen puuttuminen

Ohjelmaa suoritettaessa palvelimella on mahdollista, että kaikkia moduuleita ei löydy käyttäjän profiilista. Jos esimerkiksi moduuli *request* puuttuu, ilmenee tämä virheilmoituksena *Error: Cannot find module ”request”* silloin, kun palvelimella yritetään suorittaa ohjelmaa, jonka suoritus vaatii tämän moduulin. Moduulin pystyy asentamaan PuTTY:n terminaalin kautta kirjoittamalla komento *npm install request --save.* Nyt moduuli asentuu käyttäjän profiiliin eikä edellä mainittua virheilmoitusta enää tule. Käyttäjän profiilista saattaa ensimmäinen käyttökerralla puuttua useampi moduuli, josta johtuen ohjelmaa suoritusta joutuu yrittämään useamman kerran, ennen kuin on saanut käytyä läpi kaikki moduulit, jotka puuttuvat. Jos käyttäjä haluaa selvittää mitkä kaikki moduulit esimerkki tarvitsee, voi koodissa käytettävät moduulit tarkistaa itse koodin alusta. Koodin alussa on määritelty kaikki tarvittavat moduulit. Esimerkiksi esimerkkikoodin riviltä 5 löytyy rivi *var bodyParser = require("body-parser");* joka ilmaisee sen, että koodi tarvitsee *body-parser* -moduulin toimiakseen.

3.5 Yhteenveto käyttäjälle

Tässä kappaleessa käytiin läpi, miten tietokantaan päästään käsiksi sekä esiteltiin erilaisia menetelmiä visualisoida ja analysoida dataa. Valmiit esimerkit ja työkalut helpottavat alkuun pääsemistä tietokannan ja sensoridatan käytön osalta. Järjestelmään voi kuitenkin tehdä myös omia sovellutuksia ja työkaluja. Kuvaan 12 on listattu olennaisimmat työkalut sensorien datan käyttöön liittyen. Listassa käydään läpi eri toteutusten huonoja ja hyviä puolia, joiden avulla on helpompi päätellä mikä työkalu missäkin tilanteessa toimii parhaiten. Täytyy muistaa, että listassa on mainittu vain osa mahdollisista ohjelmista ja että lista on vain suuntaa antava käyttäjälle.

Työkalut	Hyvät puolet	Huonot puolet
JavaScript + HTML	- Valmiita toteutuksia tarjolla. - Valmiiden toteutuksien muokkaaminen mahdollista omaa käyttötarkoitusta varten.	- Vaatii tietämystä koodauksesta. - Vaatiin PuTTY:n käynnissäolon.
Python	- Soveltuu Python -ohjelmointikieltä osaaville.	- Ei valmiita toteutuksia saatavilla.
MATLAB	- Tarkka analyysi datasta. - Paljon sisäänrakennettuja funktioita helpottamassa analysointia.	- Vaatii MATLAB -lisenssin. - Vaatii datan paikallisena tiedostona.
Valmistajien Pilvipalvelut	- Helppo ja tehokas. - Sopii hyvin kuluttajille.	- Epäkäytännöllinen jos usean laitevalmistajan laitteita käytössä. - Data ulkoisen yhtiön palvelimella.

Kuva 12. Esimerkkilista käytettävistä ohjelmista

Kuten aikaisemmin on mainittu, järjestelmä on avoin ja siihen on ulkopuolisilla mahdollisuus päästä tutustumaan ja hyötymään. Mikäli järjestelmään haluaa päästä käsiksi ja tutustumaan, Oulun yliopiston Centre for Wireless Communications (CWC) 5G testiverkon henkilöstöön kannattaa ottaa yhteyttä. CWC:n henkilöstöstä löytyvät järjestelmää hallinnoivat henkilöt ja heidän kauttaan on mahdollista saada tunnukset palvelimelle. Heiltä löytyvät myös tarvittavat esimerkit ja työkalut järjestelmän hyödyntämiseen.

Tässä työssä esitellyt esimerkit ovat tehty kyseistä tietokantaa käyttäen, joten ne ovat suoraan yhteensopivia sen kanssa. Osa integroiduista sensoreista on toteutettu opiskelijoiden tekeminä kandidaatin töinä eli myös fyysisen integroinnin toteutuksesta on tarjolla esimerkkejä. Mikäli tästä on kiinnostunut, kannattaa myös tästä aiheesta kysyä henkilöstöltä. Osa töistä löytyy myös Oulun yliopiston kandidaattitöiden kannasta.

4 POHDINTA

Työn alkuperäinen tarkoitus saatiin omasta mielestäni toteutettua. Työn tarkoituksena oli kartoittaa yliopiston IoT-toteutuksen eri osa-alueet sekä käydä läpi esimerkkejä järjestelmän käytöstä.

Nyt dataa kerätään vain yliopiston kampusalueella eikä siitä juuri yhteistä hyötyä ole. Sensorien keräämää tietoa ei ole integroitu minkään toisen järjestelmän kanssa. Esimerkiksi tällä hetkellä Oulun Yliopiston Tellus -tilasta kerätyn sensoridatan voisi liittää opiskelijoiden käytössä olevaan Tuudo-sovellukseen, josta näkisi kyseisen tilan huoneenlämmön, ilmankosteuden sekä huoneilman hiilidioksidipitoisuuden.
Tämä vaihtoehto on tietenkin melko yksinkertainen ja sitä on helppo lähteä kehittämään pidemmälle. Suurin osa sensoreista lähettää itsestään paikkatietoa, jonka avulla niiden fyysinen sijainti pystytään helposti selvittämään. Kun sensoreita on paljon eri puolilla, saadaan luotua reaaliaikainen yleiskuva ympäristöstä.

Reaaliaikaisen lämpötilan tai ilmankosteuden tietäminen on tietenkin yleisin käyttökohde tämäntapaiselle toteutukselle. Datasta on kuitenkin muutakin hyötyä, kun sitä analysoidaan pidemmälle ja osataan käyttää oikein. Esimerkiksi liian korkea hiilidioksidipitoisuus sisäilmassa aiheuttaa väsymystä ja päänsärkyä, joka johtaa työtehon alenemiseen.[3] Jos sensorin lähettämästä datasta huomataan, että tietyn tilan hiilidioksidipitoisuus nousee toistuvasti suositellun rajan yläpuolelle niin tiedetään, että kyseisen tilan ilmanvaihtoa tulee tehostaa.

Yliopiston tarjoama sensoridata on avointa dataa ja näin ollen kaikille tarjottavissa. Tulevaisuudessa järjestelmää pystytään kehittämään pidemmälle ja se on esimerkiksi mahdollista integroida toisen avoimen datan kanssa. Oulun kaupunki ylläpitää omaa avoimen datan lähdettä Oulun kaupungin dataportaali.[4] Portaali ei toistaiseksi tarjoa reaaliaikaista dataa vaan informaatioita kaupungin julkisiin palveluihin liittyen. Tällaiset avoimen datan palvelut olisi mahdollista yhdistää toisiinsa, jolloin kaikki reaaliaikainen tieto ja informaatio kaupungista löytyisi samasta osoitteesta. Järjestelmä pystyisi bussiaikataulujen ja kaupungin ajankohtaisten tapahtumien lisäksi tarjoamaan reaaliaikaista tietoa kaupungista. Eri asuinalueiden lämpötilan tai esimerkiksi merenpinnan korkeuden pystyisi jokainen asukas käydä tarkistamassa ilman, että paikanpäälle joutuisi itse menemään. Osa datasta voisi luonteensa takia olla salattua, sillä ajatuksella, että ei kyseinen data kaikille kuulu. Esimerkiksi tietyn peruskoulun sisäilman laadusta sekä lämpötilasta voisi olla tarjolla tietoa, jonka saisi samasta portaalista selville käyttäjätunnuksen takaa.[4]

5 YHTEENVETO

Tässä työssä käytiin läpi Oulun yliopiston 5G testiverkon IoT-sensorien arkkitehtuuria, hallinnointia ja käyttöä. Työssä esiteltiin erilaisia toimintamalleja, miten datasta pystytään hyötymään ja miten järjestelmää voidaan käyttää hyödyksi.

Työn tarkoituksena oli auttaa lukijaa ymmärtämään mistä järjestelmä koostuu ja miten siitä voidaan hyötyä. Työssä käydyt esimerkit datan käytöstä ja hyödyntämisestä ovat yleispäteviä eli niiden muokkaaminen omaan tarpeen mukaan on tehty helpoksi.

6 LÄHTEET

[1] https://5gtn.fi/ ’’5GTN – 5G Test Network’’ Luettu 8.4.2019

[2] ”How 5G works?”, luettu 01.03.2019
http://www.emfexplained.info/?ID=25916

[3] https://www.sisailmayhdistys.fi/Terveelliset-tilat/Sisailmasto/Kemialliset-epapuhtaudet ’’Hiilidioksidi (CO2)’’ Luettu 22.04.2019

[4] https://data.ouka.fi/fi/

7 LIITTEET

Liite 1. Node.js esimerkkiohjelma: web_queries.js

```
// Test script for managing sensor data stream

const util     = require('util');
var cors       = require('cors');
var _          = require('underscore');
var bodyParser = require("body-parser");
var dateTime   = require('node-datetime');
var html       = require('html');
var CronJob    = require('cron').CronJob;
var convert    = require('xml-js');
var mysql      = require('mysql');
var express    = require('express');
var request    = require('request');
var http       = require('http');
var https      = require('https');
var moment     = require('moment');
var app        = express();
var json2csv   = require('json2csv');

var port = 2998;

// A script for adding timestamps to console messages

var origlog = console.log;

console.log = function( obj, ...placeholders ){
                        var dt = dateTime.create();
                        var formatted = dt.format('Y-m-d H:M:S');
                        if ( typeof obj === 'string' )
    placeholders.unshift("[" + formatted + "] " + obj );
  else
  {
```

```
        // This handles console.log( object )
        placeholders.unshift( obj );
        placeholders.unshift("[" + formatted + "] %j" );
    }
    origlog.apply( this, placeholders );
};

var pool = mysql.createPool ({
                        connectionLimit: 30,
        host: "localhost",
        user: "admin",
        password: "5gtnadmin",
        database: "5gtn_iprotoxi_db"
}); // mysql.createPool

function handleDate (fromDay, param) {
                        var dt = new Date(fromDay);
                        var yy = dt.getFullYear();
                        var mm = ('0'+ (dt.getMonth()+1)).slice(-2);
                        var dd = ('0'+ dt.getDate()).slice(-2);
                        var ret;

                        if (param == 0)       {
                                                ret = """+yy+'-'+mm+'-'+dd+""";
                        }
                        else {
                                                ret = """+yy+'-'+mm+'-'+dd+'%'+""";
                        }

                        return ret;

} // handleDate

// Java script for read sensor data from MySQL

function readSensorData (res, qParam, fromDay, toDay, resType) {
                        var query_str;
                        var tmp1 = handleDate(fromDay, 1);
                        var tmp2 = handleDate(toDay, 1);

                        if (qParam == 1)
                                                query_str = 'SELECT date_format(created_at,"%Y-%m-%dT%T") AS td, sensor_id
AS id, temperature as tmp FROM 5gtn_iprotoxi_test_db';
                        else if (qParam == 2) {
                                                query_str = 'SELECT date_format(created_at,"%Y-%m-%dT%T") AS td, sensor_id
AS id, lat, lng FROM 5gtn_gps_test_db WHERE created_at BETWEEN \'+tmp1+\' AND \'+tmp2\";
                        }
                        else if (qParam == 3) {
```

```
                query_str = 'SELECT lat, lng, sensor_id, temperature FROM 5gtn_gps_test_db WHERE sensor_id = \'101\'";
        }
        else if (qParam == 4) {
                query_str = 'SELECT date_format(created_at,"%Y-%m-%dT%T") AS td, sensor_id AS id, temperature AS tmp, humidity AS hum, air_pressure AS ap FROM 5gtn_gps_test_db WHERE created_at BETWEEN '+tmp1+' AND '+tmp2;
        }
        else if (qParam == 5) {
                query_str = 'SELECT date_format(created_at,"%Y-%m-%dT%T") AS td, sensor_id AS id, temperature AS tmp, humidity AS hum, air_pressure AS ap FROM 5gtn_gps_test_db WHERE created_at BETWEEN '+tmp1+' AND now()';
        }
        else if (qParam == 6) {
                query_str = 'SELECT lat, lng, sensor_id FROM 5gtn_gps_test_db WHERE sensor_id = \'20\'";
        }

        pool.getConnection(function(err, con) {
                con.query ({sql: query_str, timeout: 10000}, function(err,rows)

                {
                        con.release();
                        if(err)
                                console.log('Error:'+err.code);
                                // throw err;
                        else {
                                if (rows == '')
                                        console.log('Empty:\n');
                                else
                                        console.log('Found\n');
                                        sendResponse (res, rows, resType);
                        }
                });
        });
}

app.get('/5gtn/read/:fromDay/:toDay/:qParam', function (req, res) {
        var fromDay = req.params.fromDay;
        var toDay   = req.params.toDay;
        var resType = 1;
        var qParam  = req.params.qParam;

        console.log("Query #"+qParam+" from day "+fromDay+" to "+toDay);
        readSensorData (res, qParam, fromDay, toDay, resType);
})
```

```
// *** an app to listen a port and print to console

app.listen(port, function () {
                        console.log('Server started on port '+port)
})

app.use(function(req, res, next) {
                        res.header("Access-Control-Allow-Origin", "*");
                        res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
                        res.header("Access-Control-Allow-Methods", "POST, GET, PUT, DELETE, OPTIONS");
                        next();
});

function sendResponse (res, response, resType) {
                        if (resType == 1) {
                                                res.type('application/json');
                                                res.header("Access-Control-Allow-Origin", "*");
                                                res.header("Access-Control-Allow-Headers", "Content-Type, X-Requested-With");
                                                res.header("Access-Control-Allow-Methods", "POST, GET, PUT, DELETE,
OPTIONS");
                                                res.header("Access-Control-Max-Age", "1800");
                                                res.contentType('application/json');
                                                res.json(response);
                        }
                        else if (resType == 2) {
                                                res.setHeader('Content-disposition', 'attachment; filename=data.csv');
                                                res.setHeader('Content-type', 'text/csv');
                                                var csv = json2csv({data: response, quotes: ''});
                                                res.type('text/csv');
//                                              res.end('');
                                                res.send(csv);
                        }

} // sendResponse
```

Liite 2. Datan visualisointi: data_visualize.html

```
<html>
  <head>
    <title>Bosch XDX Temperature</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script type = "text/javascript" src = "https://www.gstatic.com/charts/loader.js"></script>
    <script type = "text/javascript">
      google.charts.load('current', {packages: ['corechart','line']});
    </script>
  </head>
  <body>
```

```
    <div id="chart_container" style="height: 600; width: 800;">
    </div>
  </body>
</html>
<script>
  var json_url = "https://pan0222.panoulu.net/5gtn/read/'2018-05-18'/'2018-06-19'/1";
  $(document).ready(function(){
    function drawCharts(){
      $.getJSON(json_url, function(data){
        var json_array = [];
        var chart_data = new google.visualization.DataTable();
        chart_data.addColumn('string', 'Date');
        chart_data.addColumn('number', 'Temperature');

        $.each(data, function (index, item) {
          json_array.push([item.td,item.tmp]);
        });

        for(i=0;i<json_array.length;i+=1){
          if (json_array[i][1] < 50){
            chart_data.addRow([json_array[i][0],json_array[i][1]]);
          }
        }

        var chart_options = {'title' : 'Room temperature ( C )',
          hAxis: {
            title: "Time"
          },
          vAxis: {
            title: 'Temperature'
          },
          'width':2000,
          'height':600
        };

        var chart = new google.visualization.LineChart(document.getElementById('chart_container'));
        chart.draw(chart_data, chart_options);
      });
    }
    google.charts.setOnLoadCallback(drawCharts);
  });
</script>
```

Liite 3. Datan tarkempi analysointi: Sensordatavalues.m

```
%Create table from CSV-file
Data = readtable('sensoriexcel.csv');


%Create Matrices from table data
temp = Data{:, 'temperature'}';
date = datetime(Data{:, 'created_at'}, 'InputFormat', 'dd.MM.yyyy HH:mm')';
hum = Data{:, 'humidity'}';
apress = Data{:, 'air_pressure'}';

%Plot coordinates from sensor data
subplot(2, 2, [3, 4])
plot(date, temp)
title('Room temperature')
xlabel('Date')
ylabel('Temperature (C^o)')
subplot(2, 2, 2)
plot(date, hum)
title('Room humidity')
xlabel('Date')
ylabel('Humidity (%)')
subplot(2, 2, 1)
plot(date, apress)
title('Room air pressure')
xlabel('Date')
ylabel('air pressure (mBar)')

%Calculate spesific information from sensor data
tempmean = mean(temp);
tempmax = max(temp);
tempmin = min(temp);
tempvar = var(temp);
temprange = tempmax - tempmin;
tempvalues = [tempmean tempmax tempmin tempvar temprange]
hummean = mean(hum);
hummax = max(hum);
hummin = min(hum);
humvar = var(hum);
humvalues = [hummean hummax hummin humvar humrange]
humrange = hummax - hummin;
apressmean = mean(apress);
apressmax = max(apress);
apressmin = min(apress);
apressvar = var(apress);
apressrange = apressmax - apressmin;
apressvalues = [apressmean apressmax apressmin apressvar apressrange]
```

```
%Plot bars from sensor data
figure(2);
subplot(2, 2, 1)
bar(tempvalues)
xticklabels({'tempmean' 'tempmax' 'tempmin' 'tempvar' 'temprange'})
title('Room temperature values')
ylabel('Temperature (C^o)')
subplot(2, 2, 2)
bar(humvalues)
xticklabels({'hummean' 'hummax' 'hummin' 'humvar' 'humrange'})
title('Room humidity values')
ylabel('Humidity (%)')
subplot(2, 2, [3, 4])
bar(apressvalues)
xticklabels({'apressmean' 'apressmax' 'apressmin' 'apressvar' 'apressrange'})
title('Room airpressure values')
ylabel('air pressure (mBar)')
```