Henri Ojakangas
Pasi Karjalainen

Bluetooth-anturi puhelinsovelluksella

RuuviTag-sensorin tiedon näyttäminen puhelinsovelluksessa

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutusohjelma
Marraskuu 2019

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Marraskuu 2019	**Tekijä/tekijät** Henri Ojakangas Pasi Karjalainen
Koulutusohjelma Tieto- ja viestintätekniikka		
Työn nimi Bluetooth-anturi puhelinsovelluksella		
Työn ohjaaja Sakari Männistö		**Sivumäärä** 27+3
Työelämäohjaaja		

Tämä opinnäytetyö käsitteli valmiin tuotteen luomista Bluetooth-anturista, pientietokoneesta sekä Android-puhelinsovelluksesta. Tavoitteena oli saada sensorin keräämä tieto näkymään puhelinsovelluksessa etäyhteyden avulla.

Opinnäytetyön alussa käydään ensin läpi käytettyjen laitteiden ja työkalujen taustoja ja ominaisuuksia lyhyesti, jotta lukijalle selviäisi hieman pohjatietoa käytetyistä eri laitteista ja työkaluista. Tämän pohjatiedon ymmärtäminen helpottaa lukijaa ymmärtämään projektin käytännön osuuden toteutusta.

Käytännön osuudessa käydään läpi projektin eri osien rakennus- ja ohjelmointivaiheista. Osiossa tutkitaan projektin eri työvaiheisiin, kuten suunnitteluun, ohjelmointiin ja testaamiseen liittyvää toimintaa. Selkeyden vuoksi projektin eri osa-alueet käydään läpi omissa osissaan, jotta lukija saa selvän kuvan jokaisen osan toiminnasta.

Projekti onnistui pääpiirteissään hyvin. Ainoastaan loppuosassa puhelinsovelluksen ja tietokannan tiedonsiirron välillä ilmeni ongelma, jota emme saaneet ratkaistua.

Asiasanat
android, Bluetooth-anturi, ohjelmointi, pientietokone, puhelinsovellus, mysql

ABSTRACT

Centria University of Applied Sciences	**Date** November 2019	**Author** Henri Ojakangas Pasi Karjalainen
Degree programme Information Technology		
Name of thesis AIR HUMIDITY METER WITH PHONE APPLICATION		
Instructor Sakari Männistö		**Pages** 27+3
Supervisor		

This thesis describes the building of a finished product which consists of a Bluetooth sensor, microcomputer and an Android phone application. The objective was to have the data collected by the sensor to be displayed in the phone application through a remote connection.

The beginning of the thesis shortly describes the history and features of the used tools and equipment, for the reader to have a better understanding of the background information of the tools and equipment used in this thesis. Understanding this background information aids in the understanding of the practical application of the thesis.

Building and programming stages of the project were explained in the practical application section. In this section the planning, programming and testing of the project were studied. For clarity, every component of the thesis has its own section for the reader to have a better understanding of their functions.

The project was mostly successful, the only thing that could not find a solution to was in the end of the project with transferring the database information to the Android phone application.

Key words
Android, Bluetooth, programming, microcomputer, phone application, MYSQL

KÄSITTEIDEN MÄÄRITTELY

BLUETOOTH	Lyhyen kantaman tiedonsiirtotekniikka
BUGI	Ohjelmointivirhe tietokoneohjelman lähdekoodissa
ETHERNET-PORTTI	Lähiverkkoliitäntä langallista verkkoyhteyttä varten
GPIO	General Purpose I/O, mikropiirin liityntä, jonka arvoa voidaan lukea/asettaa ohjelmallisesti
HDMI	High Definition Multimedia Interface, digitaalinen multimedialiitin
IDE	Integrated Development Environment, ohjelmointiympäristö
JSON	Javascript Object Notation, tiedostotyyppi näyttämään tietoa tekstinä
MICROSD	Mikrokokoinen Secure Digital muistikorttityyppi
PALVELIN	Tietokone, joka tarjoaa palveluja muille ohjelmille siinä olevan palvelinohjelmiston avulla
PINNI	Mikropiirin yksittäinen fyysinen liitin
RAM	Random Access Memory, tietokoneen keskusmuisti
SD	Secure Digital muistikortti
SSH	Secure Shell, salattuun tietoliikenteeseen tarkoitettu protokolla.
KOMENTOLIITTYMÄ	Komentosarjojen syöttämiseen tarkoitettu liittymä
TIETOKANTA	Tietokoneen muistissa sijaitseva tiedon varasto
USB	Universal Serial Bus, oheislaiteliitinstandardi
WLAN	Wireless Local Area Network, paikallinen langaton tietoverkko

KUVIOT

KUVAT

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on kehittää ja toteuttaa RuuviTag Bluetooth -anturin tiedon keräys ja näyttö mobiilisovelluksessa. Tuote kehitettäisiin kolmesta eri osasta: RuuviTag Bluetooth -anturi, Raspberry Pi -tietokone ja mobiilisovellus. RuuviTag Bluetooth -anturin lähettää tiedon tietokoneen kautta mobiilisovellukseen, joka näyttää tiedon graafisesti tai lukuna ja josta voi myös määrittää hälytysparametrit. Raspberry Pi -tietokoneella täytyy olla ohjelma, joka vastaanottaa sensorin lähettämän tiedon ja lähettää sen eteenpäin tietokantaan, josta mobiilisovellus saa tiedon kerättyä. Aluksi tietokoneelle asennettiin tietokannaksi InfluxDB ja tiedon vastaanottamiseen RuuviCollector-ohjelma, mutta erinäisten ongelmien, joita käymme läpi kappaleessa 9.1, myötä päädyimme käyttämään Node-RED-ohjelmaa ja MySQL-tietokantaa.

Opinnäytetyössä käymme läpi kaikki erilliset laitteet ja osiot alkaen Raspberry Pi -tietokoneesta ja sen lyhyestä historiasta. Raspberry Pi:stä jatkamme Node-RED-ohjelmaan ja sen historiaan sekä yleiseen käyttötarkoitukseen. Kolmantena käymme läpi hieman RuuviTag Bluetooth -anturin taustoja ja toimintaa. Tämän jälkeen kerromme vielä Android Studio -ohjelmointityökalun toiminnasta ja käytöstä, josta jatkamme lopuksi projektin käytännön toteutuksen kerrontaan. Kerronnassa käymme läpi jokaisen laitteen ja osa-alueen erikseen ja miten niitä on projektissa käytetty sekä minkälaisiin ongelmiin ja tilanteisiin törmäsimme projektia tehdessämme.

Tavoitteena opinnäytetyössä on saada toimiva tuote kehitettyä RuuviTag Bluetooth -anturista, tietokoneesta ja mobiilisovelluksesta sekä tutkia mahdollisuuksia saada näytettyä RuuviTag Bluetooth -anturin keräämä tieto puhelinsovelluksessa, joka näyttäisi kyseisen tiedon graafisesti. Halusimme myös tutkia, voiko tiedolle asettaa hälytysarvoja, joista sovellus hälyttäisi havaitessaan arvon muutoksen, ja kuinka helppoa on saada tietokannasta tieto Android puhelinsovellukseen.

2 RASPBERRY PI 3B

Raspberry Pi on luottokortin kokoinen mikrotietokone, jonka ensimmäisen version kehittivät brittiläinen Eben Upton ja myöhemmin Raspberry Pi Foundation -hyväntekeväisyysjärjestö. Raspberry Pi kehitettiin, koska haluttiin saada uusi edullinen tietokone harrastajille ja yleiseen oppimistarkoitukseen. Tämän tietokoneen käyttöjärjestelmäksi päätettiin käyttää ilmaista Linux-pohjaista käyttöjärjestelmää sen joustavuuden ja tuettavuuden takia. (Upton & Halfcree 2014, 3.)

Alkuperäinen Raspberry Pi -prototyyppi oli vain muistitikun kokoinen ja sisälsi yhden USB-liitännän ja yhden HDMI-liitännän ja microSD muistikorttipaikan. Julkaisuun mennessä Raspberry Pi kehittyi luottokortin kokoiseksi ja sisälsi kaksi USB-liitäntää, yhden HDMI:n, 3,5 mm audioliitännän, komposiittivideoliitännän, Ethernet-portin, GPIO-liitännän, SD-muistikorttipaikan sekä yhden microUSB-paikan virtalähdettä varten. Nykyisin Raspberry Pi Model 3 B:ssä on neljä USB-liitäntää, yksi HDMI, 3,5 mm audioliitäntä, Ethernet-portti, GPIO-liitäntä sekä kaksi 30-pinnistä oheislaiteliitäntää esimerkiksi näyttöä varten. (Hackitt & Cox 2012, 6–8.)

Kuten jo aiemmin kerrottiin, Raspberry Pi on hyvin pienikokoinen yleistietokone, jota on saatavilla kahta eri mallia: A ja B (KUVA 1). A-mallissa on vähemmän ominaisuuksia, ja se kuluttaa vähemmän virtaa. Tästä syystä sen hinta on myös halvempi kuin B-mallissa. Raspberry Pi:ssä voi käyttää useita eri käyttöjärjestelmiä, mutta näistä eniten käytettyjä ovat Linux-pohjaiset käyttöjärjestelmät. Raspberry Pi toimii kuin normaali pöytätietokone tai kannettava tietokone. Siihen voi liittää haluamansa oheislaitteen USB ja HDMI-liitännän avulla, esimerkiksi hiiren, monitorin sekä näppäimistön.

Raspberry Pi luotiin alun perin opetustarkoituksessa, jota käyttäen ihmiset voisivat oppia ohjelmoimaan. Raspberry Pi on hyvin monipuolinen, ja sen kehittämismahdollisuuksien ansiosta sitä voi käyttää muuhunkin kuin normaalina tietokoneena. Joitakin asioita, joihin Raspberry Pi:tä voidaan käyttää, ovat esimerkiksi robottien, erilaisten sensorien tai vaikka nettipalvelimen rakentaminen. (Upton & Halfcree 2014, 3.)

KUVA 1. Raspberry Pi 3B

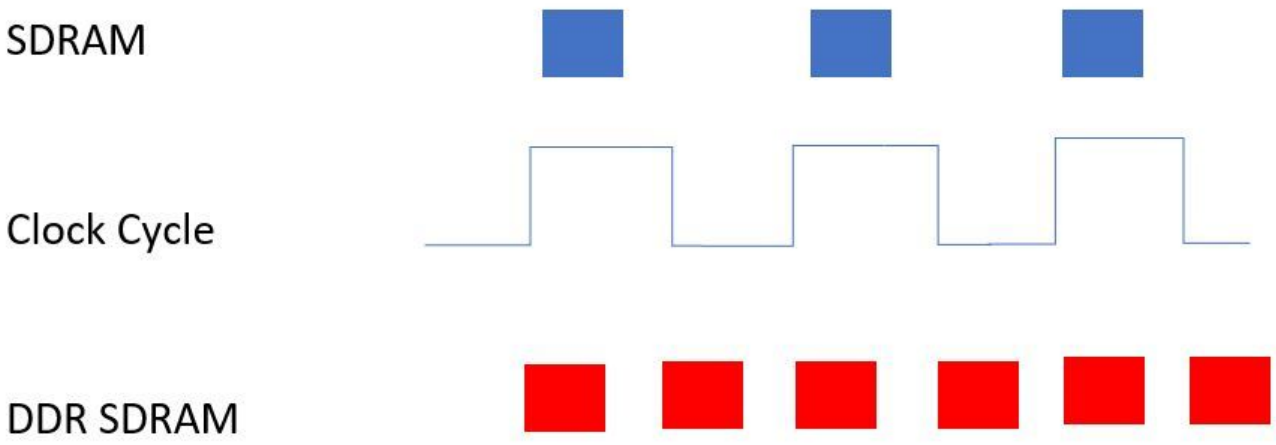
3 RASPBERRY PI 3B TEKNISET TIEDOT

Projektissa käytetty Raspberry Pi koostuu neliytimisestä ARM Cortex-A53 -prosessorista, Broadcom Videocore IV -näytönohjaimesta, yhden Gigatavun LPDDR2 RAM-muistista, 10/100 Ethernet ja 802.11n langattoman verkon adapterista, Bluetooth 4.1 -komponentista sekä ulkoisista lisälaiteporteista. (Barnes, R 2016.)

ARM Cortex-A53 -prosessori on vähävirtainen keskitason prosessori, joka on rakennettu ARMv8-A-arkkitehtuurilla. A53-prosessorissa on yhdestä neljään ydintä. Ytimien määrästä riippuen L1-välimuistin koko vaihtelee kahdeksan ja 64 kilotavun välillä ja L2-välimuisti vaihtelee 128 kilotavun ja 2 megatavun välillä. L1-välimuisti koostuu data- ja ohjelmavälimuistista, joiden tarkoitus on säilyttää ja välittää nopeasti tietoa ja ohjeita prosessorille ja prosessorilta. L2-välimuisti on hieman hitaampaa ja kaikkien ydinten yhteiskäytössä olevaa välimuistia. (ARM 2014.)

Videocore IV -näytönohjain perustuu neljään shader-prosessoriin, jotka soveltuvat erittäin hyvin pikseleiden prossessointiin. Videocore IV -näytönohjain käyttää ruutupohjaistsa pikselin hahmonnusta, mikä vähentää kuvan lataamiseen tarvittavaa kaistanleveyttä huomattavasti verrattuna välittömään hahmonnukseen. (BROADCOM 2013.)

Keskusmuistina on yksi gigatavu 900 megahertsin taajuksista LPDDR2-muistia. LPDDR2 tulee Low Power Double Data Rate sanoista, jotka tarkoittavat vähävirtaista kaksinkertaista tiedonsiirtomäärää. Kaksinkertainen tiedonsiirto tarkoittaa sitä, että tieto kulkee nousevan ja laskevan pulssijakson aikana (KUVA 2).



KUVA 2. DDR SDRAM siirtää tietoa sekä nousevan että laskevan pulssijakson aikana. (Mukaillen Thornton. 2018.)

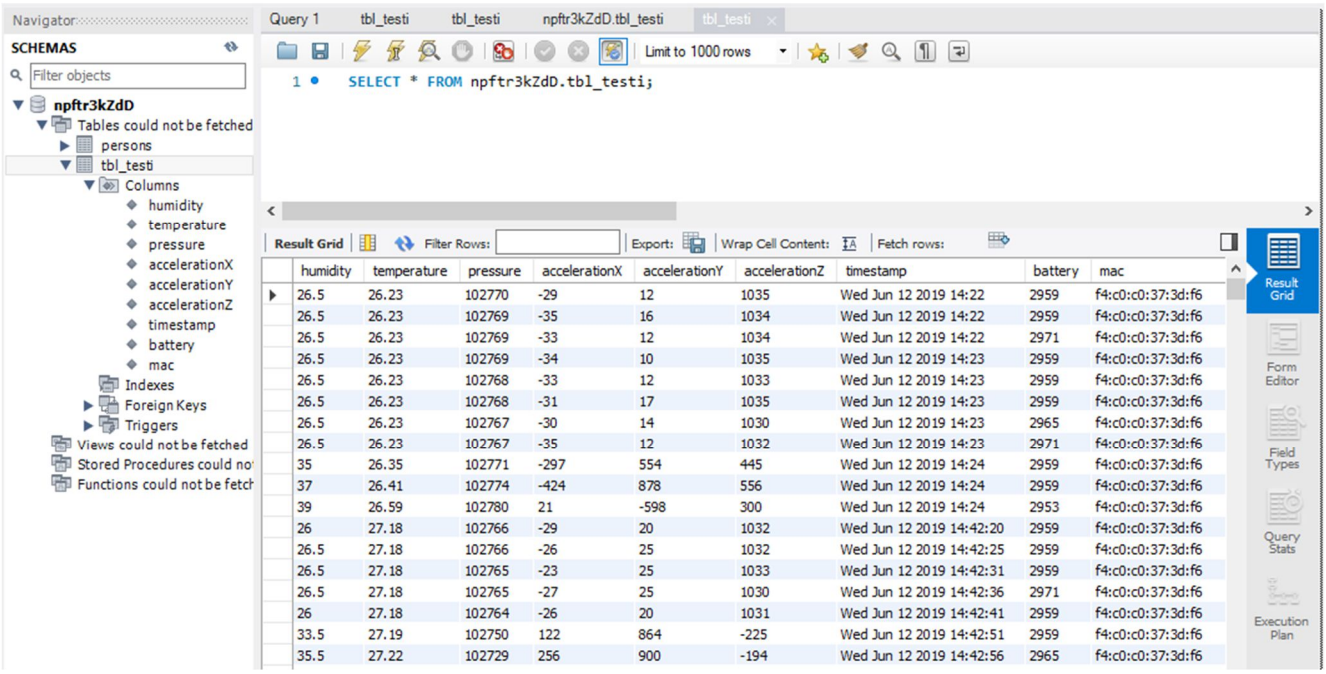
Langattoman verkon sovittimena Raspberry Pi 3B:ssä toimii 802.11n langattoman verkon standardi. 802.11n standardi toimii 2.4 ja 5 gigahertsin taajuudella, ja sen teoreettinen maksiminopeus on 300 megabittiä sekunnissa. 802.11n:n kaistanleveys on kahdestakymmenestä neljäänkymmeneen megahertsiin ja käyttää OFDM-modulaatiota. OFDM tulee sanoista Orthogonal Frequency-Division Multiplexing ja mahdollistaa tiedonsiirron useilla toisiaan häiritsemättömillä taajuuskanavilla yhtä aikaa. Antennitekniikkana 802.11n standardi käyttää MIMO (Multiple Input, Multiple Output) -teknolgiaa, jonka avulla on mahdollista käyttää ja koordinoida useaa eri radiosignaalia yhtä aikaa. MIMO mahdollistaa suuremman langattoman yhteyden toimintasäteen ja esteiden läpäisyn. (Abdelrahman, R. B. M., Mustafa, A. B. A. & Osman, A. A. 2015.)

Bluetooth-sovittimena toimii Bluetooth 4.1. Bluetooth 4.1 on päivitetty versio Bluetooth 4.0:sta, jonka mukana tuli paranneltuja ohjelmisto-ominaisuuksia. Näistä ominaisuuksista tärkeimmät olivat mobiiliverkkojen ja Bluetooth-verkkojen yhtäaikainen toiminta, Bluetooth-laitteiden ja -vastaanottimien automaattinen linkitys ja päivitys, sekä niiden sulavampi yhteinen tiedonsiirto. (Bluethooth Special Interest Group. 2013.)

4 MYSQL-TIETOKANTA

Relaatiotietokanta on tietokokoelma valmiiksi määritetyillä suhteilla. Nämä tiedot on järjestelty taulukkoon, jossa tieto on aseteltu riveihin ja sarakkeisiin. Sarakkeissa esitetään sarakkeen nimi ja tietotyyppi, ja riveissä esitetään kerätty arvo. (Amazon Web Services. 2019.) MySQL on avoimella lähdekoodilla toimiva relaatiotietokannan hallintaohjelma. SQL tulee sanoista Structured Query Language, joka tarkoittaa järjesteltyä rakenteista kyselykieltä. MySQL-tietokannanhallintaohjelma mahdollistaa tietokannan hallinnan verkossa sekä etäyhteyden avulla (KUVA 3).

Päädyimme käyttämään MySQL-tietokantaa osana projektiamme, koska se on yleisin tietokannanhallintaohjelmisto ja tästä syystä sen toiminnasta ja käytöstä on olemassa paljon tietoa ja ohjeita. Suunnitelmien mukaan tarkoituksemme oli käyttää InfluxDB-tietokantaa, mutta kun vaihdoimme RuuviCollectorin NodeRED-ohjelmaan, vaihdoimme samalla myös tietokantaa niiden yhteensopivuuksien vuoksi.



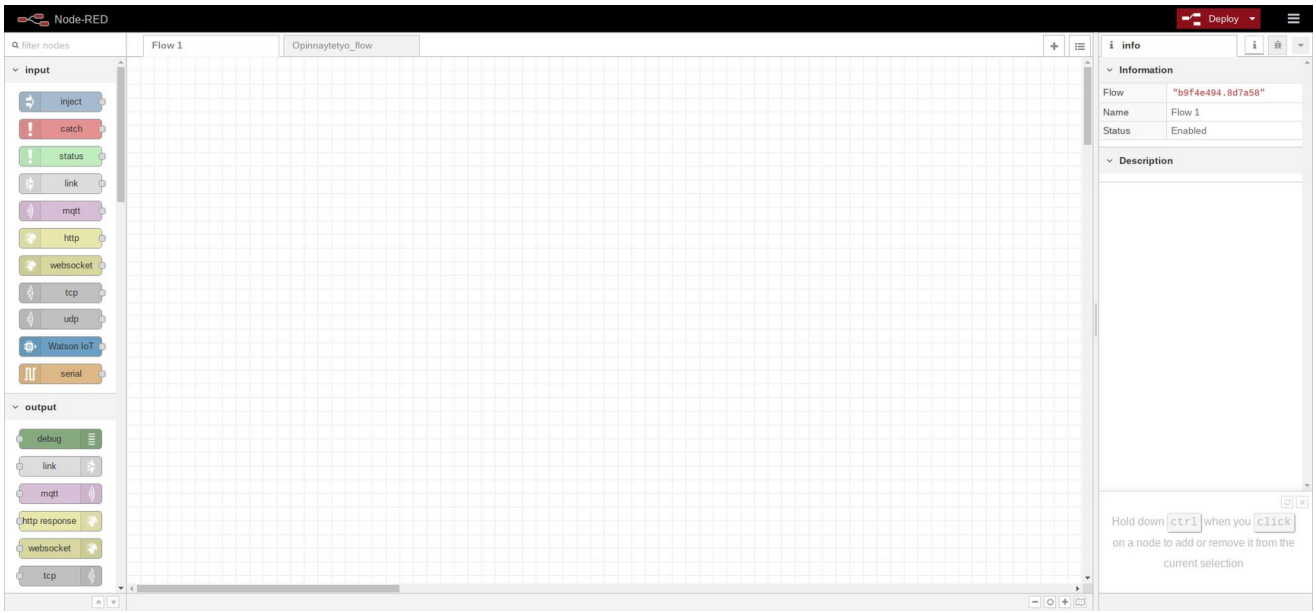
KUVA 3. MySQL Workbench -tietokannanhallintaohjelmisto

5 NODE-RED

Node-RED sai alkunsa Nick O'Learyn ja Dave Conway-Jonesin pienenä sivuprojektina. Aluksi projektin tarkoitus oli olla yksinkertainen ohjelmien visualisoinnin ja manipuloinnin soveltuvuusselvitys. Kun ohjelman laajuus kasvoi, siitä tuli avoimen lähdekoodin projekti syyskuussa 2013. Tästä ohjelmasta tuli myös yksi JS Foundationin ensimmäisistä projekteista. (OpenJS Foundation 2019.)

Node-RED on graafinen ohjelmaympäristö, jolla kyetään luomaan ohjelmia, jonka osat ovat visuaalisesti helposti luettavissa. Node-RED:ssä käytetään niin kutsuttuja "nodeja" jotka ovat erilaisia laatikoita, joilla on jokin tietty toiminta ohjelmassa, näille "nodeille" annetaan tietoa, jota ne käsittelevät ja sitten lähettävät eteenpäin seuraavalle "nodelle". Tämän visuaalisesti helpon tulkinnan takia ohjelma on yleisesti helpommin lähestyttävä kuin normaali ohjelma. Node-RED:stä näkee helposti ohjelman kaikki eri osat ja tapahtuman vain vilkaisemalla näkyvää kaaviota. (Ruuvi Lab 2017.)

Päädyimme käyttämään tässä opinnäytetyössä Node-RED ohjelmaa, koska aiemmassa ohjelmassa, jota käytimme, sisälsi myös jonkin verran bugeja, jotka johtivat moniin virheilmoituksiin ohjelmassa. Valitsimme Node-RED:n myös sen visuaalisesti helposti tulkittavan näkymän vuoksi ohjelmointia varten (KUVA 4). Tällä ohjelmalla kykenemme yhdistämään helposti eri laitteita yhteen esim. verkko-ominaisuuksien kanssa. Tällä ohjelmalla yhdistimme RuuviTagin ja Raspberry Pi 3b:n, jotta saamme RuuviTagin keräämän tiedon lähetettyä puhelinsovellukselle. Node-RED skannaa Bluetooth-laitteet, tässä tapauksessa RuuviTagin, ja kerää sen lähettämän tiedon, jonka se lähettää eteenpäin tietokantaan. Tietokannasta tieto välitetään eteenpäin puhelinsovellukseen, josta tiedon tarkastelu on helppoa.



KUVA 4. Node-RED-ohjelmointiympäristö

6 RUUVITAG

RuuviTag on IP67 sertifikaatilla vesitiivis Bluetooth-sensori, jonka on kehittänyt suomalainen startup-yritys Ruuvi (KUVA 5). RuuviTag alkoi ideana kehittää avoimen lähdekoodin Bluetooth-lähetintä. Alkuun RuuviTag:n kehitys oli vain pienikokoinen projekti, mutta saatuaan paljon kysyntää tuotteelle kehittäjät tekivät RuuviTag:lle KickStarter-joukkorahoituskampanjan vuonna 2016. KickStarter.com-sivuston joukkorahoituskampanjan avulla RuuviTag sai paljon näkyvyyttä ja keräsi noin 170 000 Yhdysvaltojen dollaria. (Jämsä & Tulabadi 2016)

RuuviTag on kehitetty avoimella lähdekoodilla, ja tästä syystä sitä kyetään käyttämään moniin erilaisiin tarkoituksiin. Avoimen lähdekoodin avulla käyttäjät voivat muokata RuuviTagia omiin tarpeisiinsa sekä jakaa kehittämiään projekteja muiden käyttäjien kanssa. RuuviTag mittaa lämpötilaa, ilmankosteutta ja ilmanpainetta, siinä on vaihdettava pitkäkestoinen paristo, jonka avulla sen ylläpidon kustannukset ovat vähäisiä. RuuviTag toimii –40 celsiusasteesta +85 celsiusasteeseen saakka. Kaikista ominaisuuksistaan huolimatta RuuviTag:n virrankulutus on vähäistä. Tässä opinnäytetyö projektissa käytämme ruuvitagia kosteuden mittarina, jonka lähettämä data kerätään tietokantaan bluetoothin avulla ja lähetetään eteenpäin käyttäen WLAN:a.

KUVA 5. RuuviTag ja sen kotelo

7 RUUVITAG TEKNISET TIEDOT

Projektissa käytetty RuuviTag bluetooth -anturi koostuu Nordic Semiconductor nRF52832 järjestelmäpiiristä, STMicroelectronics LIS2DH12 kiihtyvyysanturista, Bosch BME 280 ympäristöanturista, NFC -A tag antennista, 100mAh CR2477 patterista, kahdesta painikkeesta, kahdesta LED:stä, 45 millimetrin leveästä piirilevystä, 52 millimetrin leveästä kotelosta sekä pitkän kantaman radioantennista. (Ruuvi.com 2019.)

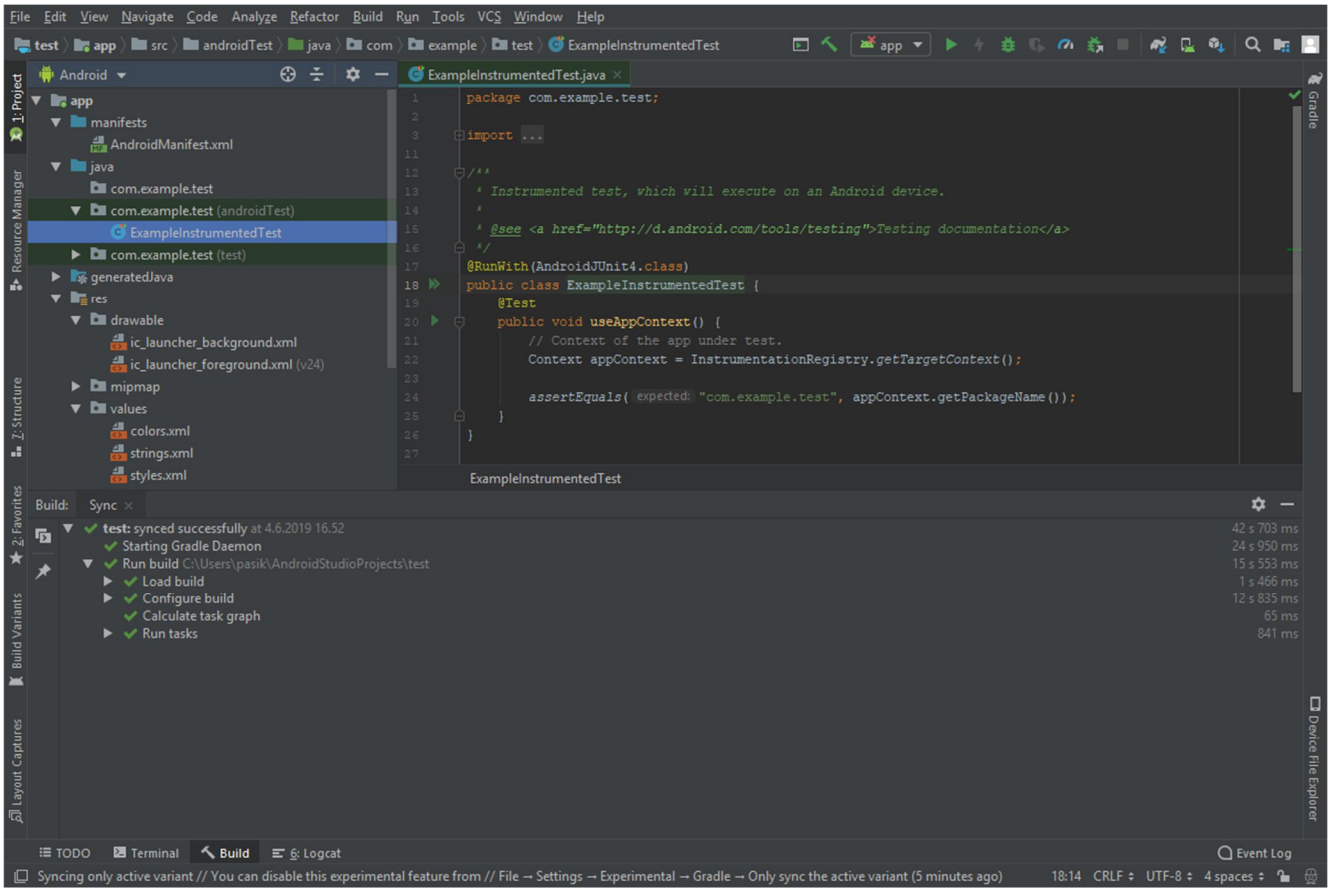
Nordic puolijohde nRF52832 järjestelmäpiiri on keskitason nRF52-sarjan järjestelmäpiiri. nRF52832 tukee monia protokollia ja samanaikaisuuteen kykenevä järjestelmäpiiri. Piirissä tuettuja protokollia ovat muiden muassa Bluetooth 5, Bluetooth mesh ja ANT. Järjestelmä piiri on rakennettu ARM Cortex -M4 suorittimen ympärille, jonka liukulukuyksikkö toimii 64 megahertsin taajuudella. Piirissä on myös NFC-A-tunniste, jota käytetään yksinkertaistettujen pariliitosten kanssa. Piirissä on tuki myös PDM- ja I2S-liikennöintiä käyttäville oheislaitteille. nRf52832-järjestelmäpiiri tukee myös useita eri protokollia, kuten Bluetooth 5, Wirepas Connectivity, Mira OS ja Quuppa. (Nordic Semiconductor 2019.)

LIDS2DH12 on erittäin pienitehoinen korkean suorituskyvyn kolmiakselinen lineaarinen kiihtyvyysanturi. Anturin virrankulutus on erittäin pieni, ja se pystyy mittaamaan kiihtyvyyttä 1–5.3 kilohertsin taajuudella. Anturin avulla kyetään mittaamaan liikettä ja lämpötilaa. (STMicroelectronics 2019.)

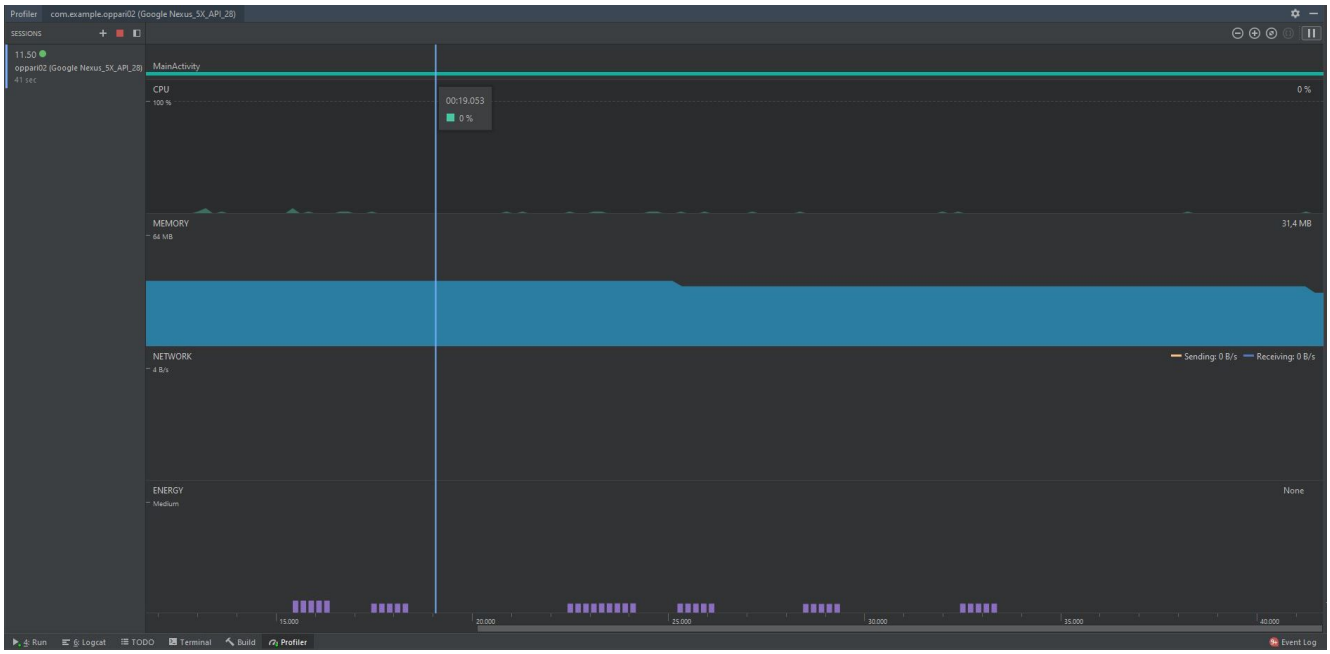
Bosch BME 280 ympäristöanturi, joka on tarkoituksella kehitetty mobiilisovelluksia varten, jossa koolla ja pienellä virrankulutuksella on suuri merkitys. Anturiin on yhdistetty lämpö-, ilmanpaine- ja ilmankosteusanturit kahdeksanpinniseen metallikanteen. Anturin ilmanpaineen mittaus toimii 300–1100 hPa:n väliltä ja lämpötilan mittaus toimii -40 ja 85 ºC välillä. Anturin syöttöjännite on 1.2–3.6 voltin väliltä. Anturin keskimääräinen virrankulutus on 1.8–3.6 mikroampeeria yhden hertsin toimintataajuudella. (Bosch Sensortec 2019)

8 ANDROID STUDIO

Android Studio on virallinen IDE-ohjelmointityökalu Android-sovelluksille (KUVA 6). Android Studiossa on saatavilla koodieditori, jolla luodaan sovellusten ohjelmat. Editorin tuetut koodi kielet ovat Java, Kotlin ja C/C++. Ohjelmassa on saatavilla myös asetelmaosio, jolla voi muokata sovelluksen ulkonäköä ja asettelua. Android Studio ohjelmaan kuuluu myös puhelinemulaattori sovellusten testaamiseen ohjelman sisällä. (Android Studio 2019) Tällä ohjelmalla kyetään luomaan sovelluksia, jotka toimivat useilla eri Android versioilla. Android Studiossa on profilointia varten myös työkalu, jolla voidaan katsastella reaaliaikaisia tilastoja tehtyjen sovellusten toiminnasta. Esimerkkinä tästä on muistin käyttö (KUVA 7) (Android Studio 2019).



KUVA 6. Esimerkki Android Studiosta



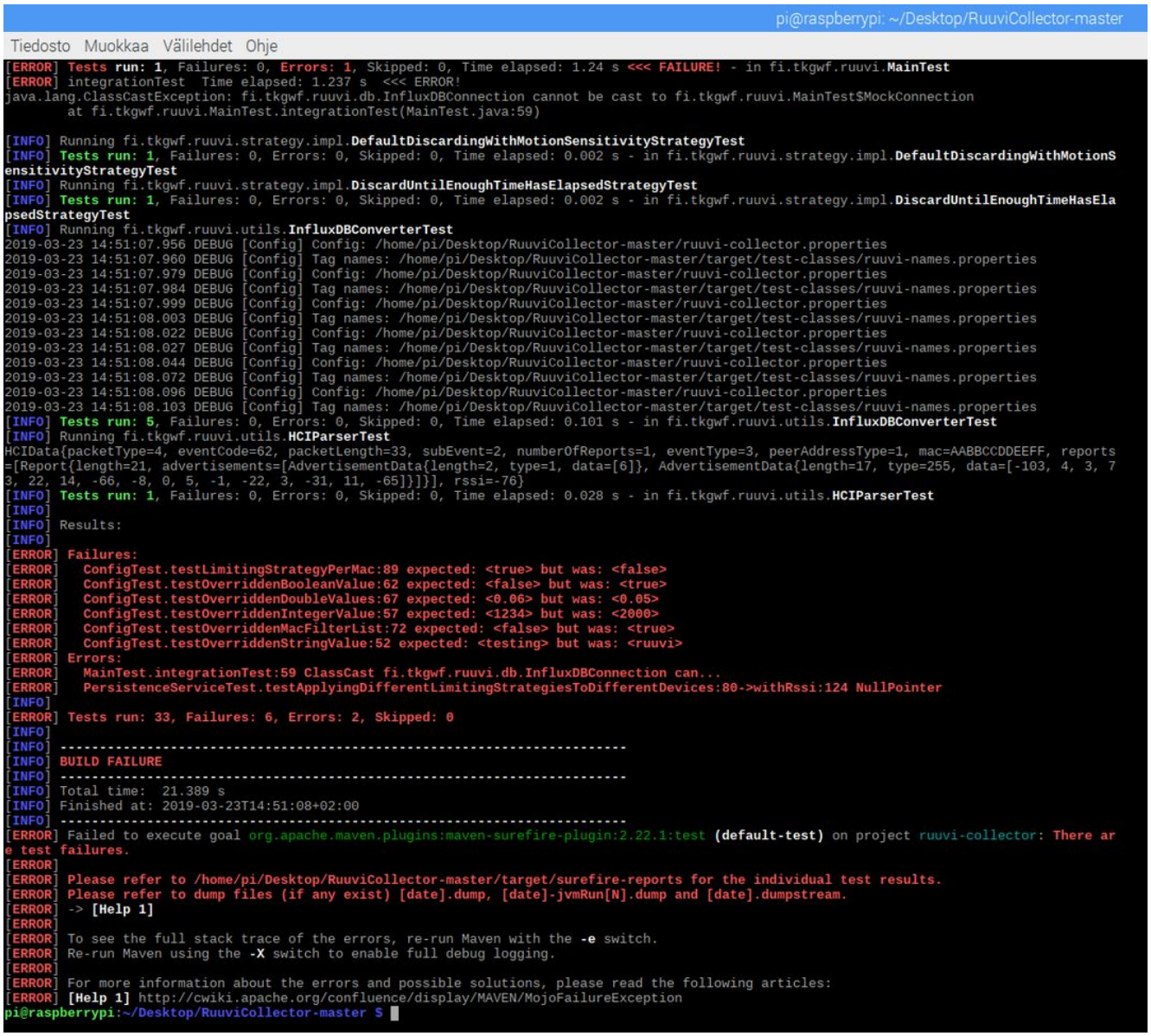
KUVA 7. Android Profiler-profilointityökalu

9 KÄYTÄNNÖN TOTEUTUS

9.1 Suunnittelu

Aloitimme opinnäytetyöprojektin suunnittelun rakentamalla UML-kaavioita projektin kokonaisuudesta, jotta saisimme konkreettisemman kuvan projektin kokonaisuuden toiminnasta (Liite-1). Suunnitteluvaiheen jälkeen aloitimme Raspberry Pi:n ohjelmoimisen projektia varten. Projektin ensimmäisessä versiossa tarkoituksemme oli käyttää RuuviCollector-ohjelmaa ja InfluxDB-tietokantaa. Projektin edetessä kuitenkin huomasimme, että RuuviCollector oli vielä kehitysvaiheessa, ja se sisälsi jonkin verran bugeja, jotka johtivat erinäisiin virheilmoituksiin (KUVA 8). Yritimme aluksi ratkaista ongelmia, joita nämä bugit tuottivat, mutta lopulta päätimme ottaa käyttöön Node-RED-ohjelman RuuviCollectorin sijasta sen suoraviivaisuuden vuoksi. Node-RED:n kanssa oli myös helpompi ottaa käyttöön MySQL-tietokanta, joten vaihdoimme myös InfluxDB:n siihen.

Teimme myös puhelinsovellukselle omat UML-kaaviot sen suunnitellusta toiminnasta. Tältä pohjalta tutkimme, miten erinäiset siirtymävalikot ja sivun vaihdokset toimivat Android-sovelluksessa. Löysimme useita erilaisia tapoja ja valikkoratkaisuja, mutta päädyimme käyttämään valikkoa, jonka saa auki valikkopainikkeella tai vetämällä sormella näytön vasemmasta reunasta. Aukinaisesta sivuvalikosta pystyisi avaamaan sivuja, jotka esittävät RuuviTagin keräämän tiedon, esim. ilmankosteuden tai ilmanlämpötilan. Näistä sivuista voisi myös valita RuuviTagin, jos niitä on useampi. Tällä tavoin sovelluksesta kykenisi katsomaan haluamansa RuuviTagin, tai RuuviTagin sijainnin tiedon.



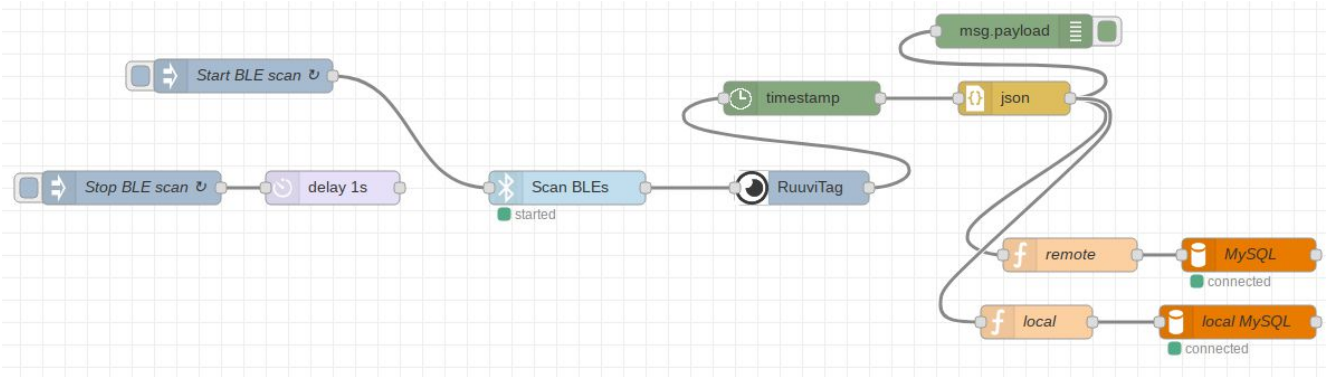
KUVA 8. Bugeista johtuvia RuuviCollector-virheilmoituksia komentoliittymässä

9.2 Ohjelmointi

Tässä luvussa tarkastelemme projektin eri osioiden ohjelmointia, jossa ohjelmointi kattaa sekä säädöt ja asennukset että ohjelman kirjoituksen. Käymme läpi Raspberry Pi:n asennuksen ja sen sisäisten asioiden ohjelmoinnin, Android-puhelinsovelluksen ohjelmoinnin sekä tietokannan ohjelmoinnin.

9.2.1 Raspberry Pi

Raspberry Pi 3b-laitteeseen asensimme Raspbian-käyttöjärjestelmän, joka toimii tämän työn pohjana. Raspbianiin asensimme tiedonkeruuta varten Node-RED-ohjelman ja tietokannaksi MySQL-tietokannan. Node-RED tarvitsi muutaman lisäosan, jotta sillä kyettiin skannaamaan Bluetooth-laitteita. Näiden lisäosien asennus toi ohjelmaan Bluetooth-ominaisuuden, jonka avulla saimme luettua Ruuvitagin lähettämää dataa. Rakensimme Node-RED flow-ohjelman kahdestatoista eri osasta (KUVA 9).



KUVA 9. Opinnäytetyön Node-RED flow-ohjelma

Node-RED:n käyttö aloitettiin asentamalla se ja siihen kuuluvia lisäosia eli uusia noodeja, Raspbianista löytyvän komentoliittymän avulla. Node-RED:n asennus onnistui, mutta lisäosien asennuksessa ilmeni lieviä ongelmia. Ongelmana oli esimerkiksi, että meiltä puuttui tiettyjä oikeuksia muuttaa tiedostoja. Löysimme kuitenkin Node-RED:n sisältä itsestään lisäosien haun ja asennuksen, joita käyttämällä onnistuimme asentamaan halutut lisäosat. Tarvitsemamme lisäosat olivat Bluetooth-skanneri, RuuviTagin virallinen lisäosa, MySQL-lisäosa sekä lisäosa, joka osaa tallentaa ajan ja päivämäärän.

Aloitimme rakentaa flow-ohjelmaa projektiin Node-RED:ssä asentamalla ensin Bluetooth-skannerin, jota ei ollut valmiina asennettu Node-RED:n. Skannerin asennuksen jälkeen lähdimme kokoamaan omaa kaaviota erinäisten Ruuvi-foorumin artikkelien ja kommenttien avulla. Ruuvin foorumeilta ja ohjeista löysimme RuuviTag-noodin käytön ohjeet sekä miten sitä oli käytetty eri käyttötarkoituksissa. Näistä muokkasimme omaan projektiimme flow-ohjelman.

Projektin flow-ohjelmassa Bluetooth-skanneri saa käskyn käynnistyä ja pysähtyä samaan aikaan, mutta pysähtymiskäsky tulee parin sekunnin viiveellä, jonka avulla saimme skannausta hitaammaksi, jotta

saadut tulokset olisivat helpommin tulkittavissa. Kun skannaus on alkanut, RuuviTag lisäosa alkaa toimimaan ja kerää RuuviTagin lähettämän tiedon ja lähettää sen eteenpäin JSON-muuntimeen, joka muuttaa saadun tiedon JSON-ohjelmointikielelle (KUVIO 1). Tästä eteenpäin tieto lähetetään eteenpäin funktionoodille, josta tieto menee eteenpäin MySQL-noodille, joka tallentaa tiedon MySQL-tietokantoihin.

[{"ilmankosteus":"47.5"},{"Päivämäärä":"Fri Jul 05 2019"},
{"aika":"16:42:39"},{"MAC-osoite":"f4:c0:c0:37:3d:f6"}]

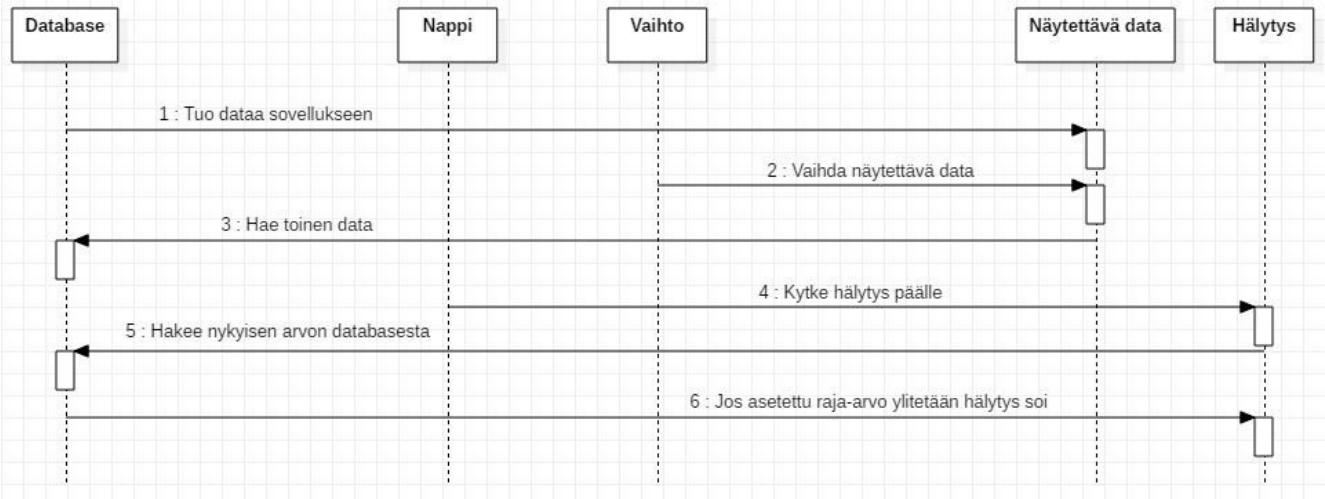
KUVIO 1. Tietokannasta saatava tieto JSON-tietona

Kun saimme kerättyä Ruuvitagin lähettämän tiedon Node-RED ohjelmaa käyttämällä, aloimme asentaa MySql-verkkotietokantaa, jota aioimme aluksi käyttää skannatun tiedon varastoimiseen. Tässä vaiheessa kuitenkin törmäsimme MySql-verkkopalvelimen kanssa pieneen ongelmaan, koska toisella kerralla, kun työstimme projektia ja yritimme käyttää tätä kyseistä sivustoa, se ei jostain syystä toiminut, joten loimme myös varmuuden vuoksi Raspberry Pi:n sisälle tietokannan, mikäli verkkopalvelin ei olisi tulevaisuudessa jostain syystä käytettävissä.

Työstessämme projektia ilmeni kuitenkin, että tiedonsiirto verkkopalvelimelta PHP-komentosarjaa käyttäen puhelinsovellukseen ei onnistunut puuttuvien oikeuksien myötä. Tämän takia otimme pääkäyttöön Raspberry Pi:n sisään luomamme tietokannan, josta kykenemme käyttämään PHP-komentosarjoja tiedonsiirtoon.

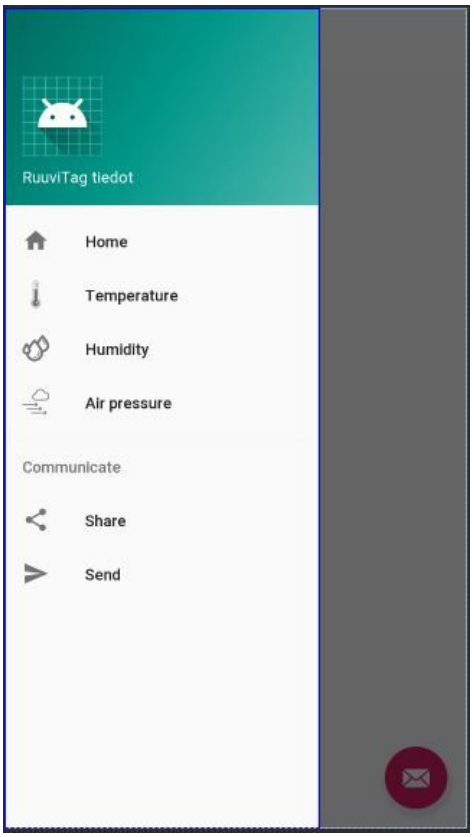
9.2.2 Android-puhelinsovellus

Sovellus tehtiin Android Studio –ohjelmalla Java-ohjelmointikielellä. Sovelluksen käyttöliittymän suunnittelu toteutettiin erinäisiä UML-kaavioita ja muita puhelinsovelluksia apuna käyttäen (KUVIO 2). Aluksi tutkimme sivunvaihtoa pyyhkäisyliikettä käyttäen ja saimme sen toimimaan. Tämän jälkeen tutkimme mahdollisuuksia asetusvalikon käyttöönottoon, ja päädyimme valikkoon, jonka saa pyyhkäisemällä vasemmasta reunasta, sekä vasemman yläkulman valikkonäppäimestä. Nämä ominaisuudet rakennettiin eri osioissa eri malleja käyttäen, ja tästä syystä ominaisuudet eivät toimineet keskenään erinäisten päällekkäisyyksien vuoksi (Liite 2).



KUVIO 2. Puhelinsovelluksen toiminnan sekvenssikaavio

Löysimme Android Studiosta valmiin valikkomallipohjan, johon aloimme rakentaa uudestaan pyyhkäisysivunvaihtoa. Luovuimme sivunvaihdosta pyyhkäisyllä ominaisuuksien päällekkäisyyksien vuoksi ja otimme käyttöön vain navigointivalikon, jonka saa vasemmasta reunasta pyyhkäisemällä, sekä vasemman yläkulman valikkopainikkeesta (KUVA 10).



KUVA 10. Puhelinsovelluksen navigointivalikko

Sovellus toimii käytännössä siten, että kaikissa sovelluksen sivuissa on sama asetelma, mutta kun avataan uusi sivu, sen sisältö vaihtuu myös. Tämä toimii siten, että ensimmäisen sivun sisältö kytkeytyy pois päältä ja seuraavan sivun sisältö kytkeytyy päälle. Kun saimme tietokannan toimimaan, aloimme työstämään sovelluksen tiedon graafista osiota, jolla saisimme sensorin keräämän tiedon näytettyä sovelluksen eri sivuilla.

Ohjelma toteutettiin neljällä Java-tiedostolla: MainActivity, TemperatureActivity, HumidityActivity sekä AirPressureActivity. MainActivityssä luodaan sovelluksen etusivu sekä navigointivalikko. MainActivityä käytetään myös jokaisen Activity-tiedoston pohjana, jonka avulla navigointivalikko sekä sisältö saadaan muihinkin sivuihin. Temperature-, Humidity- sekä AirPressureActivity ovat samankaltaisia sisällöltään, ja niistä vain vaihdetaan sisällön näkyvyys (KUVA 11).

```
public class temperatureActivity extends MainActivity
        implements NavigationView.OnNavigationItemSelectedListener


{
    @Override
    protected void onStart () {
        super.onStart();

        View m = findViewById(R.id.textView);    //tämä "poistaa" etusivun tekstin/sisällön
        m.setVisibility(View.GONE);    //tämä "poistaa" etusivun tekstin/sisällön

        View t = findViewById(R.id.textTemperature);
        t.setVisibility(View.VISIBLE);

        View button = findViewById(R.id.refreshButton);
        button.setVisibility(View.VISIBLE);
    }
```

KUVA 11. Esimerkki Activity-tiedostosta (TemperatureActivity.java)

Sovelluksen sivunvaihto toteutettiin Switch case-funktiolla, jossa jokainen ”case” edustaa tiettyä id:tä, jonka perusteella ajetaan tietty funktio, jos ”case” toteutuu. Funktiossa määritetään Intent, jonka avulla haluttu Activity saadaan avattua sovelluksessa (KUVA 12).

```
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();
    switch (id) {
        case R.id.nav_home:
            launchMainActivity();
            break;
        case R.id.nav_temperature:
            launchTemperatureActivity();
            break;
        case R.id.nav_humidity:
            launchHumidityActivity();
            break;
        case R.id.nav_airPressure:
            launchAirPressureActivity();
            break;
    }

    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}

private void launchMainActivity(){
    Intent home = new Intent ( packageContext: this, MainActivity.class);
    startActivity(home);
}

private void launchTemperatureActivity(){
    Intent temperature = new Intent( packageContext: this, temperatureActivity.class);
    startActivity(temperature);
}

private void launchHumidityActivity(){
    Intent humidity = new Intent ( packageContext: this, humidityActivity.class);
    startActivity(humidity);
}

private void launchAirPressureActivity(){
    Intent airPressure = new Intent ( packageContext: this, airPressureActivity.class);
    startActivity(airPressure);
}
```

KUVA 12. Switch-case ja funktio sivun vaihdolle

Sisältö ja sen asettelu määritellään content_main.xml ja app_bar_main.xml tiedostojen avulla. App_bar_main.xml toteuttaa sovelluksen ulkoasun, eli käytännössä taustan, sovelluksen yläreunan työkalupalkin, jossa sijaitsevat menunvalintanäppäin sekä sivun otsikko. App_bar_main tiedoston asettelussa käytettiin CoordinatorLayout-asettelua, jota yleisesti käytetään sivujen pohjana, jos käytössä on yksi tai useampi perijäsivu. Content_main:ssa määritetään kaikki sovelluksen sisällön sijainnit, sisältäen näppäimet ja tekstinäkymät. Content_main:ssa on käytössä ConstraintLayout-asettelu, jolla jokaiselle sivun sisällölle saadaan määritettyä tarkka sijainti.

9.2.3 Tietokannan ohjelmointi

Tietokantana käytimme MySQL-tietokantaa remotemysql.com-palvelussa sekä Raspberry Pi:n sisällä olevaa tietokantaa. Tietokannan verkkopalveluun teimme niiden tarjoaman käyttöliittymän avulla ja Raspberry Pi:n sisäisen tietokannan tietokoneen komentoliittymän komentoja käyttäen. Komentoina käytimme ”CREATE TABLE”-komentoa, jossa määritellään jokaisen tietokannan taulukon sarakkeet ja niiden tietotyypit, esimerkiksi meidän tapauksessamme JSON. Node-RED:llä ajoimme funktion, joka lisäsi RuuviTagin keräämän tiedon tietokantaan luotuun taulukkoon oikeassa järjestyksessä (KUVA 13). Funktiossa esiintyvä ”msg.payload.[muuttuja]” on RuuviTagin keräämä tieto, joka tuodaan taulukkoon samassa järjestyksessä jossa sarakkeiden nimetkin esitettiin.

```
msg.topic = "INSERT INTO tbl_testi
(humidity, temperature, pressure, accelerationX, accelerationY, accelerationZ, timestamp, battery, mac)"+"
VALUES ('" + msg.payload.humidity + "','" + msg.payload.temperature + "', '" + msg.payload.pressure + "',
'" + msg.payload.accelerationX + "', '" + msg.payload.accelerationY + "', '" + msg.payload.accelerationZ + "',
'" + msg.mydate + " " + msg.mytimes + "',  '" + msg.payload.battery + "', '" + msg.payload.mac + "');"
return msg;
```

KUVA 13. Tietokannan luomisfunktio Node-RED:ssä

Kun saimme tietokannat rakennettua, aloimme tehdä PHP-komentosarjoja. Käytimme db_connect.php-komentosarjaa ensin saadaksemme yhteyden tietokantaan, jonka jälkeen teimme jokaiselle haluamallemme tietotyypille oman PHP-komentosarjan, esimerkiksi nouda_temperature.php, joka hakee tuoreimman arvon tietokannan lämpötilasarakkeesta. Lisäsimme myös jokaiseen tiedonhakukomentosarjaan päivämäärän haun, jonka perusteella saamme tietää, että haettu tieto on uusin RuuviTagin keräämä arvo (KUVA 14).

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

humidity	temperature	pressure	accelerationX	accelerationY	accelerationZ	timestamp	battery	mac
26.5	26.23	102770	-29	12	1035	Wed Jun 12 2019 14:22	2959	f4:c0:c0:37:3d:f6
26.5	26.23	102769	-35	16	1034	Wed Jun 12 2019 14:22	2959	f4:c0:c0:37:3d:f6
26.5	26.23	102769	-33	12	1034	Wed Jun 12 2019 14:22	2971	f4:c0:c0:37:3d:f6
26.5	26.23	102769	-34	10	1035	Wed Jun 12 2019 14:23	2959	f4:c0:c0:37:3d:f6
26.5	26.23	102768	-33	12	1033	Wed Jun 12 2019 14:23	2959	f4:c0:c0:37:3d:f6
26.5	26.23	102768	-31	17	1035	Wed Jun 12 2019 14:23	2959	f4:c0:c0:37:3d:f6
26.5	26.23	102767	-30	14	1030	Wed Jun 12 2019 14:23	2965	f4:c0:c0:37:3d:f6
26.5	26.23	102767	-35	12	1032	Wed Jun 12 2019 14:23	2971	f4:c0:c0:37:3d:f6
35	26.35	102771	-297	554	445	Wed Jun 12 2019 14:24	2959	f4:c0:c0:37:3d:f6
37	26.41	102774	-424	878	556	Wed Jun 12 2019 14:24	2959	f4:c0:c0:37:3d:f6
39	26.59	102780	21	-598	300	Wed Jun 12 2019 14:24	2953	f4:c0:c0:37:3d:f6
26	27.18	102766	-29	20	1032	Wed Jun 12 2019 14:42:20	2959	f4:c0:c0:37:3d:f6
26.5	27.18	102766	-26	25	1032	Wed Jun 12 2019 14:42:25	2959	f4:c0:c0:37:3d:f6
26.5	27.18	102765	-23	25	1033	Wed Jun 12 2019 14:42:31	2959	f4:c0:c0:37:3d:f6
26.5	27.18	102765	-27	25	1030	Wed Jun 12 2019 14:42:36	2971	f4:c0:c0:37:3d:f6
26	27.18	102764	-26	20	1031	Wed Jun 12 2019 14:42:41	2959	f4:c0:c0:37:3d:f6
33.5	27.19	102750	122	864	-225	Wed Jun 12 2019 14:42:51	2959	f4:c0:c0:37:3d:f6
35.5	27.22	102729	256	900	-194	Wed Jun 12 2019 14:42:56	2965	f4:c0:c0:37:3d:f6
37	27.27	102756	254	861	-299	Wed Jun 12 2019 14:43:01	2965	f4:c0:c0:37:3d:f6
38.5	27.34	102752	603	1099	-37	Wed Jun 12 2019 14:43:06	2959	f4:c0:c0:37:3d:f6

Selain

localhost/fetch_stuff.php

{"success":1,"data":[{"lämpötila":"27.34"},{"aika":"Wed Jun 12 2019 14:43:06"},{"MAC-osoite":"f4:c0:c0:37:3d:f6"}]}

```php
<?php
include 'db_connect.php';

//Query to select temperature & timestamp & mac
$query = "SELECT temperature, timestamp, mac FROM tbl_testi ORDER BY timestamp DESC LIMIT 1";
$result = array();
$tempArray = array();
$timeArray = array();
$macArray = array();
$response = array();

//Prepare the query
if($stmt = $con->prepare($query)){
    $stmt->execute();

    //Bind the fetched data to $temperature
    $stmt->bind_result($temperature, $timestamp, $mac);

    //Fetch 1 row at a time
    while($stmt->fetch()){
        //Populate the arrays
        $tempArray["l&auml;mp&ouml;tila"] = $temperature;
        $timeArray["aika"] = $timestamp;
        $macArray["MAC-osoite"] = $mac;
        $result[]=$tempArray;
        $result[]=$timeArray;
        $result[]=$macArray;
    }
    $stmt->close();
    $response["success"] = 1;
    $response["data"] = $result;
    }else{
    //Some error while fetching data
    $response["error"] = 0;
    $response["message"] = mysqli_error($con);}
//Display JSON response
echo json_encode($response);

?>
```

KUVA 14. Tietokanta, selaimen antama JSON-tieto sekä PHP-komentosarja tiedonkeruuseen

9.3 Testaus ja käyttöönotto

Projektin jokaista osaa testattiin aluksi omana kokonaisuutena niiden toimivuuden selvittämiseksi. Jokaisen osan valmistuttua osat yhdistettiin isommaksi kokonaisuudeksi. Ensimmäisenä osana kokosimme yhteen kaikki RuuviTagiin ja Raspberry Pi -tietokoneeseen liittyvät osa-alueet. Raspberry Pi:hin etsimme meidän käyttötarkoituksiimme parhaiten soveltuvaa käyttöjärjestelmää ja muutamien kokeilujen ja testauksien jälkeen päädyimme Raspbian-käyttöjärjestelmään. Tämä käyttöjärjestelmä on varta vasten tehty Raspberry Pi -tietokonetta varten, joten esimerkiksi sen toiminnasta löytyi hyvin tietoa. Käyttöjärjestelmän asennuksen jälkeen testasimme aikaisemmin mainittua RuuviCollector-ohjelmaa, jonka jälkeen vaihdoimme RuuviCollectorin Node-RED-ohjelmaan.

Node-RED-ohjelma vaati perehtymistä sen käyttöön ja opetteluun, mutta kokeilujen ja käyttöönotto-ohjeiden avulla saimme tehtyä siihen yksinkertaisen esimerkin Flow-ohjelmasta, jotta näimme että Node-RED toimi halutulla tavalla. Tältä pohjalta aloimme rakentaa projektiin oikeaa flow-ohjelmaa, johon löysimme paljon apua RuuviTagin foorumeilta. Ensin kokeilimme foorumeilta saadun tiedon avulla yleispätevää RuuviTag-skanneria, jotta näimme, että Node-RED kykenee keräämään halutun tiedon, ja että RuuviTagit toimivat ja lähettävät tiedon oikeassa muodossa (Ruuvi.com 2017). Tässä vaiheessa, kun olimme rakentaneet yleispätevän flow-ohjelman, se lähetti lokiin RuuviTagin dataa useita kertoja sekunnissa. Tätä lähdimme hidastamaan rakentamalla viivytyksen tiedon lähetykseen. Saimme tiedon tulemaan viiden sekunnin viiveellä, jonka jälkeen pystyimme tulkitsemaan tietoa helpommin ja näkemään mahdolliset virheilmoitukset ennen kuin ne katosivat muun tiedon sekaan (Liite 3).

Saatuamme flow-ohjelman toimimaan halutulla tavalla yhden RuuviTagin kanssa, halusimme myös nähdä, onko käytössä useampi kuin yksi RuuviTag. Tässä testauksessa ei ilmennyt mitään ongelmaa, vaan rakentamamme flow-ohjelma onnistui nappaamaan molempien RuuviTagien keräämän tiedon. Tämän pohjalta aloimme rakentaa MySQL-sulautusta flow-ohjelmaan. MySQL-tietokannan kanssa kokeilimme paikallista tietokantaa, valmiiksi omistamaamme nettisivua tietokannalla sekä ulkopuolista MySQL-verkkopalvelinpalvelua. Näistä aluksi totesimme paikallisen tietokannan ja ulkopuolisen MySQL-verkkopalvelinpalvelun Remotemysql.com:in toimivan parhaiten, mutta käyttötarkoituksiimme Raspberry Pi:n sisäinen MySQL-tietokanta toimisi parhaiten. Tässä vaiheessa kävi ilmi, että Node-REDin Bluetooth-skannaus loppuu yllättäen satunnaisin ajoin flow-ohjelman käynnistämisestä. Tätä yritimme ratkaista poistamalla viivytysnoodin rakentamastamme flow-

ohjelmasta. Löysimme Ruuvin keskustelupalstalta, että emme olleet ainoita, joilla oli tämä kyseinen ongelma, mutta emme myöskään löytäneet ongelmaan yksittäistä ratkaisua. Kokeilimme keskustelupalstalla ilmenneitä ratkaisuvaihtoehtoja, mutta mikään ei näyttänyt vaikuttavan ongelmaan. Tässä vaiheessa asetimme myös Raspberry Pi:hin etäyhteysmahdollisuuden SSH:n avulla. Etäyhteys Raspberry Pi:hin saatiin ensiksi laittamalla Raspberry Pi:stä etäyhteys päälle. Tämän voi tehdä joko, Raspberryn työpöydän kautta Raspberryn konfigurointi-ikkunasta kytkemällä SSH päälle, tai terminaalista käyttämällä komentoja ”sudo raspi-config” ja valitsemalla ”Interfacing Options” ja ”SSH”. Tämän jälkeen modeemin asetuksista tuli avata tietyt nettiyhteyteen tarvittavat reitittimen portit, mikä mahdollistaa tiedon saannin internetin välityksellä. Avasimme reitittimestä portit 22 ja 80, joiden avulla saimme etäyhteyden Raspberryyn. Raspberry Pi:n sisäisten ohjelmien ollessa tässä pisteessä, keskityimme projektin puhelinsovelluspuoleen enemmän.

Ensimmäiseksi puhelinsovelluksessa testasimme, kuinka saisimme sivunvaihdon toimimaan niin, että se toteutuisi sormen pyyhkäisyllä. Eräiden ohjeiden avulla saimme kyseisen ominaisuuden toimimaan, minkä jälkeen aloimme rakentaa navigointivalikkoa puhelinsovellukselle (Ranjoni 2017). Ensin rakensimme navigointivalikon itse ohjeiden mukaan, jotta saisimme paremman käsityksen sen toiminnasta. Navigointivalikon rakentamisen jälkeen huomasimme, että sivunvaihdos pyyhkäisyllä ei enää toiminutkaan erinäisten ohjelman päällekkäisyyksien vuoksi. Tästä syystä palasimme taaksepäin ja otimme käyttöön Android Studiossa olevan valmispohjan navigointivalikolle, jossa emme huomanneet paljoakaan eroa rakentamaamme navigointivalikkoon, mutta ajattelimme, että valmispohja voisi olla vakaampi sovelluksen toimivuuden puolesta. Yritimme vielä tähän pohjaan rakentaa pyyhkäisyominaisuutta siinä onnistumatta, mutta totesimme, että navigointivalikon sivunvaihdos käy tarpeisiimme.

Seuraavaksi teimme tyhjät Activity-tiedostot RuuviTag:n keräämille eri tiedoille. Näihin halusimme saada jokaiseen oman tietyn arvon näkymään, esimerkiksi TemperatureActivityyn tulisi RuuviTagien lähettämä lämpötila-arvot. Tätä aloimme tekemään luomalla jokaiselle tietotyypille oman sisältötiedoston, joka loisi oman sivun jokaiselle tietotyypille omalla sisällöllään. Tässä pian huomasimme päällekkäisyyksiä, koska jokainen Activity-tiedosto näytti päällimäisenä MainActivity-tiedoston sisällön. Tähän emme löytäneet hyvää ratkaisua, joka olisi toiminut haluamallamme tavalla. Tästä syystä päädyimme tekemään yhden sisältötiedoston, johon laitoimme jokaiselle Activity:lle oman sisällön, mutta asetimme siinä myös sisällön automaattisesti ”gone”-arvolle, mikä tarkoittaa sitä, että sisältö ei ole olemassa ennen kuin sitä kutsutaan. Tämä tapahtuu jokaisella sivulla sille sisällölle, joka

halutaan näyttää kyseisellä sivulla. Esimerkiksi lämpötilasivulla ohjelma osaa näyttää vain lämpötilan arvot, eikä minkään muun sivun sisältöä.

Saatuamme tyhjät sivut ja etäyhteyden Raspberry Pi-tietokoneeseen aloimme tutkimaan, miten saisimme toteutettua tiedon näyttämisen tyhjillä sivuilla. Löysimme lukuisia esimerkkejä listan luomiseen, joka näyttäisi tietokannassa olevan tiedon listana. Testasimme näitä esimerkkejä omina sovelluksinaan, niin kuin ne oli ohjeessa tehty, ja ne toimivat ongelmitta. Näistä esimerkeistä käytimme pääasiassa kahta esimerkkiä. Toisessa kirjoittaja luo ensin JSON-muotoisen ilmoituksen, josta kirjoittaja muokkaa tiedon listaan (Khan 2019). Toisessa esimerkissä kirjoittaja ensin tekee listanäkymän ja JSON-hakufunktion, jossa funktio hakee kerättyjen arvojen nimillä tietoa. Tämän jälkeen ohjelma tulostaa JSON-tiedot ja kuvat listanäkymään. (Parsania 2018.)

Nähtyämme kuinka sovellukset toimivat, aloimme rakentaa ohjelmia omiin tarpeisiin, mutta tässä törmäsimme kuitenkin ongelmaan, jossa muokkaamamme ohjelma ei tuntemattomasta syystä saa noudettua meidän PHP-komentosarjassamme olevaa tietoa ja näytettyä sitä. Tätä yritimme ratkaista muokkaamalla meidän komentosarjaamme samaan muotoon kuin ohjeessa olevat, sekä tutkimalla, onko meidän ohjelmassamme tai nettilinkissä jotain ohjeista poikkeavaa. Tätä miettiessämme tulimme siihen johtopäätökseen, että ongelma luultavammin johtuu PHP-komentosarjasta tai käyttämästämme linkistä, sillä sovellus toimii muuten oikein ja on löytämiemme ohjeiden mukainen. Ensimmäisessä esimerkissä, jota käytimme, saimme näkyviin sovelluksen ilmoituksen, joka näyttää sovelluksen koodissa käytetyn nettilinkin sisällön. Sovelluksemme saa myös luotua listanäkymän, mutta ei saa tuotua tämän nettilinkin sisältöä siihen.

Epäilimme että nettilinkkimme on pääsyyllinen toimimattomuuteen, sillä vain yksi löytämistämme esimerkeistä käytti HTTP-siirtoprotokollaa ja muut nykyaikaisemmat esimerkit käyttivät HTTPS-siirtoprotokollaa, joka mahdollistaa suojatun tiedonsiirron. Otimme selvää HTTPS-protokollan toiminnasta ja miten sellaisen voisi ottaa käyttöön tähän projektiin, mutta selvisi, että HTTPS-protokollan saa kahdella tavalla: ostamalla sertifikaatin palveluntarjoajalta, tai käyttämällä Let’s Encrypt:n ilmaista ei-kaupallista sertifikaattia, johon tarvitsee SSH-yhteyden ja sudo-komentorivikomennon oikeudet. Käyttämämme palveluntarjoajamme ei tarjonnut meille SSH-yhteyttä, tästä syystä projektin loppuosa jäi tähän pisteeseen ja emmekä saaneet tiedonsiirtoa puhelinsovelluksen listanäkymään.

10 YHTEENVETO

Tämä opinnäytetyöprojekti oli tutkimusmielessä erittäin mielenkiintoinen toteuttaa, sillä emme suunnitteluvaiheessa tienneet, kuinka laitteet toimivat keskenään. Halusimme saada selville, onko tällainen projekti näillä laitteella mahdollista teknisesti toteuttaa. Onneksi suurin osa projektista luonnistui erittäin hyvin. Laitteet toimivat hyvin keskenään, ja niiden kokoonpano yhdeksi kokonaisuudeksi oli yleisesti helppoa pois sulkien pienet ohjelmien keskeneräisyydet ja tukiongelmat. Nämä ongelmat kuitenkin lisäsivät tietotaitoamme niin tietoteknisten ongelmien ratkaisemiseen kuin tiedonhakuun. Tästä syystä jouduimme perehtymään laitteisiin ja sovelluksiin entistä enemmän.

Projektia tehdessämme oma tietotaitomme ja kokemuksemme tietoteknistä tuotekehitystä kohtaan kasvoi ja saimme lisää itsevarmuutta työntekoon muuttuvassa projektiympäristössä. Etsiessämme lisää tietoa projektin osa-alueista, saimme samalla tietää osa-alueiden eri käyttökohteista enemmän, mikä helpottaa ongelmanratkaisukykyä, kun tietää, miten ja mitä käyttäen ongelmia voi ratkaista. Vaikka projekti muuttui alustavista suunnitelmista, sen yllättävän laajuuden mukaan lopputulos onnistui kuitenkin mielestämme pääpiirteissään tyydyttävästi, varsinkin vähäisen kokemuksemme perusteella tällaisten projektien suhteen.

Mielestämme RuuviTagin käyttäminen ja sen asetusten kokoonpano oli helppoa, mutta alun ongelmien myötä myös hieman stressaavaa. Kuitenkin ratkaistuamme ongelmat ja siirtyessämme NodeREDin käyttöön, RuuviTagin käyttäminen helpottui huomattavasti. Puhelinsovelluksen kehittäminen oli hyvin vähäisellä kokemuksella omalla tavallaan stressaavaa, mutta koska käytimme Java-ohjelmointikieltä, ongelmiimme löytyi helposti tietoa ja mahdollisia ratkaisuja. Tietokantojen rakentaminen oli mieluista ja emmekä törmänneet sen kanssa suurempiin ongelmiin. Projektin tekeminen kokonaisuudessa on ollut mielenkiintoista, ja sitä tehdessä olemme kyenneet seuraamaan omaa kehitystämme ja osaamistamme opinnäytetyön aiheesta. Mielestämme projekti on ollut hyvin opettavainen ja monipuolinen, sillä projektia tehdessä siihen on tullut enemmän osa-alueita kuin osasimme suunnitteluvaiheessa odottaa. Projektissa selvästi on vielä kehitettäviä asioita, joita olisi jatkossa mielenkiintoista lähteä tutkimaan ja ratkaisemaan. Kun projekti olikin odotettua laajempi, se loi eräänlaisia suorituspaineita saada projekti valmiiksi ennen alustavaa eräpäivää.

LÄHTEET

Amazon Web Services. 2019. What is a Relational Database? Saatavissa: https://aws.amazon.com/relational-database/. Viitattu 16.6.2019.

ARM. 2014. ARM Cortex-A53 MPCore Processor Revision: r0p2 Technical Reference Manual. Saatavissa: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0500d/DDI0500D_cortex_a53_r0p2_trm.pdf. Viitattu 7.9.2019

Bluetooth Special Interest Group. 2013. Quick Reference Guide. Saatavissa: https://web.archive.org/web/20140330135810/https://www.bluetooth.org/en-us/specification/adopted-specifications. Viitattu 7.9.2019

Bosch Sensortec. 2019. BME280 Integrated Environmental Units Technical data. Saatavissa: https://www.bosch-sensortec.com/bst/products/all_products/bme280. Viitattu 7.9.2019

BROADCOM. 2013. VideoCore IV 3D Architecture Reference Guide Saatavissa: https://web.archive.org/web/20160803202903/https://www.broadcom.com/docs/support/videocore/VideoCoreIV-AG100-R.pdf. Viitattu 7.9.2019

Android Studio. Measure App Performance. Saatavissa: https://developer.android.com/studio/profile/android-profiler. Viitattu 1.7.2019.

Android Studio. Meet Android Studio. Saatavissa: https://developer.android.com/studio/intro. Viitattu 14.6.2019.

Hackitt, B. N., Cox, T. 2012. The MagPi 01. Saatavissa: https://www.raspberrypi.org/magpi-issues/MagPi01.pdf. Viitattu 14.6.2019.

Abdelrahman, R. B. M., Mustafa, A. B. A. & Osman, A. A. 2015. A Comparison between IEEE 802.11a, b, g, n and ac Standards. Saatavissa: http://www.iosrjournals.org/iosr-jce/papers/Vol17-issue5/Version-3/D017532629.pdf. Viitattu 7.9.2019

Jämsä, L., Tulabadi, S. 2016. Ruuvitag – Open-Source Bluetooth Sensor Beacon.. Saatavissa: https://www.kickstarter.com/projects/463050344/ruuvitag-open-source-bluetooth-sensor-beacon. Viitattu 16.6.2019.

Khan, B. 2019. JSON Parsing in Android – Fetching From MySQL Database. Saatavissa: https://www.simplifiedcoding.net/json-parsing-in-android/. Viitattu 1.7.2019.

MySQL. 2019. What is MySQL? Saatavissa: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Viitattu 16.6.2019.

Nordic Semiconductor. nRF52832 Flexible, efficient Bluetooth 5 and Bluetooth mesh multiprotocol SoC. 2019. Saatavissa: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52832. Viitattu 7.9.2019

OpenJS Foundation. Node-RED: About. Saatavissa: https://nodered.org/about/. Viitattu 16.6.2019.

Parsania, H. 2018. Android ListView Using Volley | Parse JSON Listview With Image And Text. Saatavissa: https://demonuts.com/android-listview-using-volley/. Viitattu 1.7.2019.

Ranjoni, A. 2017. Swipe Between Activities Android Studio. Saatavissa: https://www.youtube.com/watch?v=kKqZoS4THnY. Viitattu 27.6.2019.

Ruuvi Lab. 2017. Flow-based Programming for the Internet of Things. Saatavissa: https://lab.ruuvi.com/node-red/. Viitattu 1.7.2019.

Ruuvi.com. 2017 Node-RED node. Saatavissa: https://f.ruuvi.com/t/node-red-node/451 Viitattu 1.7.2019

Ruuvi.com. 2019. RuuviTag Technical Specifications. Saatavissa: https://ruuvi.com/files/ruuvitag-tech-spec-2019-7.pdf. Viitattu 7.9.2019

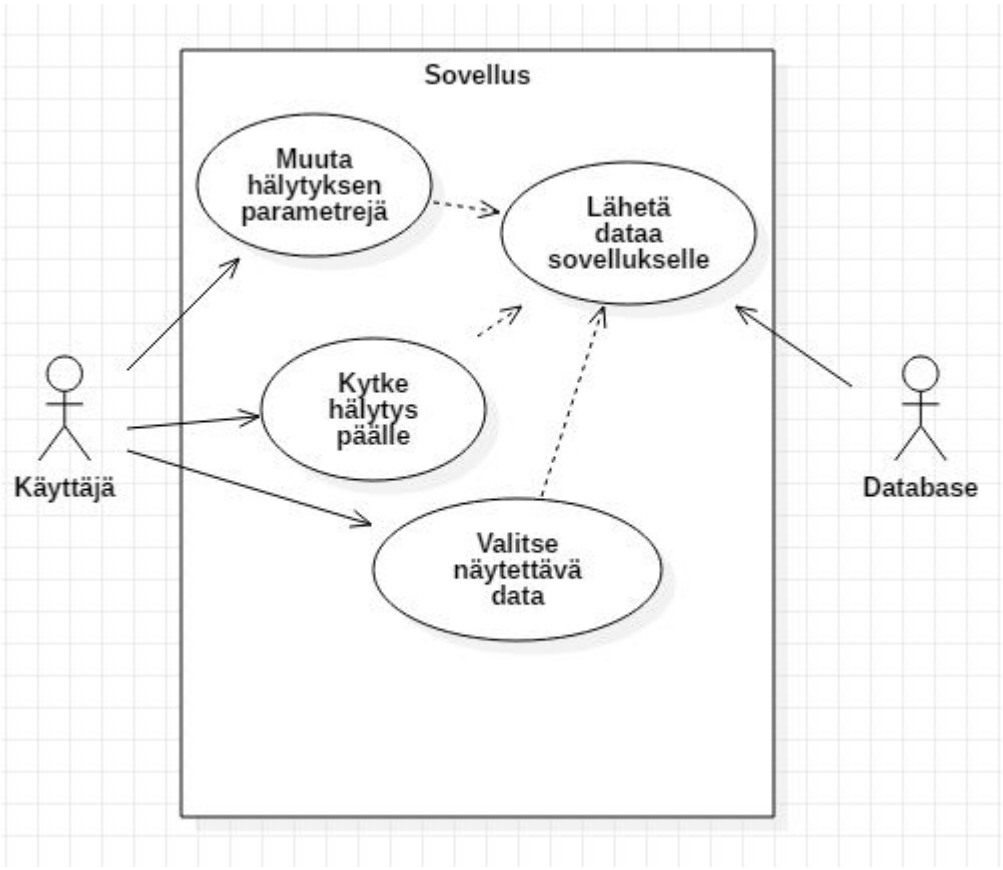
STMicroelectronics. LIS2DH12. 2019. Saatavissa:https://www.st.com/content/st_com/en/products/mems-and-sensors/accelerometers/lis2dh12.html. Viitattu 7.9.2019

Barnes, R. 2016. Raspberry Pi 3: Specs Benchmarks & Testing. Saatavissa: https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/ Viitattu 7.9.2019

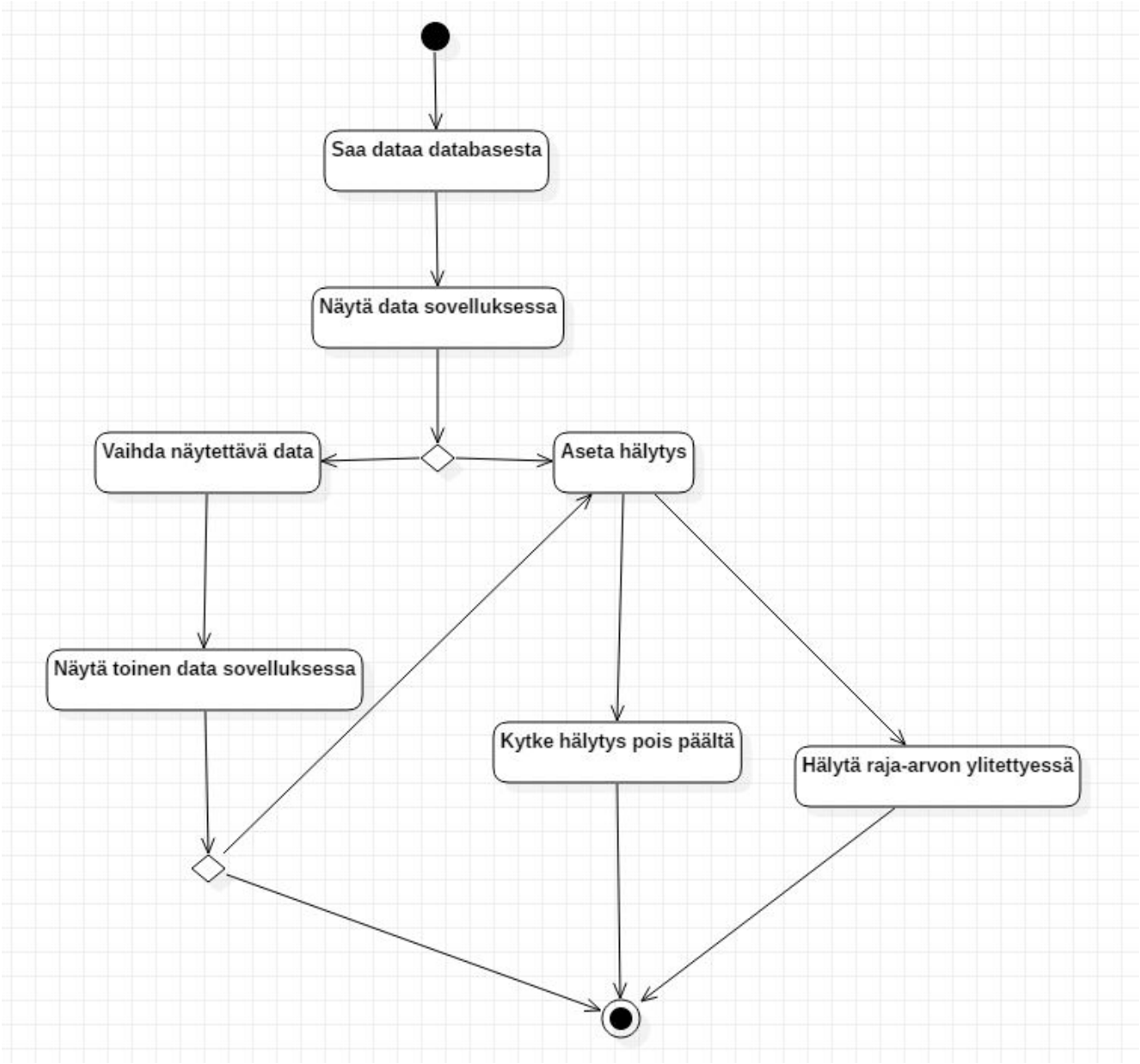
Thornton, S. 2018. What is DDR (Double Data Rate) Memory and SDRAM memory. Saatavissa: https://www.microcontrollertips.com/understanding-ddr-sdram-faq/. Viitattu 7.8.2019

Upton, E., Halfcree, G. 2014. Raspberry Pi User Guide. 3., uudistettu painos. Chichester: WILEY.

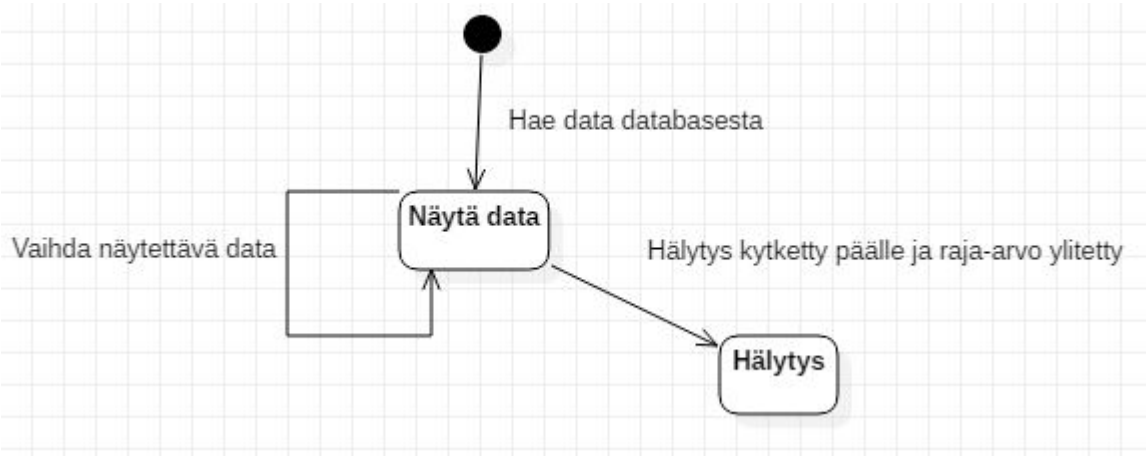
Teimme projektin aluksi neljä eri UML-kaaviota projektin suunnittelun avuksi. Yksi näistä UML-kaavioista on sovelluksen sequence kaavio, jonka kuva löytyy opinnäytetyön Android-puhelinsovellus kappaleesta.



KUVIO 3. Use case kaavio.



KUVIO 4. Activity kaavio



KUVIO 5. Statechart kaavio

Näistä kaavioista saimme luotua yleiskuvan projektin suunnitellusta toiminnasta. Etenkin Activity kaavioista näkee tarkemmin puhelinsovelluksen alustavan toiminnan suunnittelun. Näiden kaavioiden pohjalta kykenimme jaottelemaan projektia helpommin hallitaviin osa-alueisiin, joita lähdimme rakentamaan yksitellen.

Puhelinsovelluksen ohjelmoinnissa lähdimme ensimmäiseksi toteuttamaan siirtymä toimintoa toiselle sivulle näytön oikeasta reunasta pyyhkäisemällä. Löysimme netistä toimivat ohjeet tämän toiminnan toteutukseen ja käytimme niitä.

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MotionEvent;

public class MainActivity extends AppCompatActivity {

    float x1,x2,y1,y2;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


    }
    public boolean onTouchEvent(MotionEvent touchEvent){
        switch(touchEvent.getAction()){
            case MotionEvent.ACTION_DOWN:
                x1 = touchEvent.getX();
                y1 = touchEvent.getY();
                break;
            case MotionEvent.ACTION_UP:
                x2 = touchEvent.getX();
                y2 = touchEvent.getY();
                if(x1 < x2){
                Intent i = new Intent( packageContext: MainActivity.this, SwipeLeft.class);
                startActivity(i);
            }else if(x1 > x2){
                Intent i = new Intent( packageContext: MainActivity.this, SwipeRight.class);
                startActivity(i);
            }
            break;
        }
        return false;
    }
}
```

Kuva 15. Toimiva sivunvaihto pyyhkäisyllä

Tässä vaiheessa lähdimme toteuttamaan valikko ruutua, jonka saisi auki valikko painikkeesta ja näytön vasemman reunan pyyhkäisystä auki. Päädyimme käyttämään Android Studiossa olevaa pohjaa, jossa

navigointivalikko on tehty valmiiksi. Navigointivalikko oli kuitenkin ristiriidassa sivun vaihdon pyyhkäisy toiminnan kanssa, kun yhdistimme valikon aukaisu toiminnon ja sivunvaihdon emme enää onnistuneet vaihtamaan sivua oikeasta reunasta pyyhkäisemällä. Yritimme ratkaista tätä ongelmaa tuloksetta, joten luovuimme sivun vaihdosta pyyhkäisemällä ja tyydyimme valikossa olevan painikkeen käyttöön tässä toiminnossa.

Node-RED ohjelmasta löysimme ajastin “noden” jota käytimme viivytyksen luomiseen. Valitsemalla tämän “noden” pääsimme muokkaamaan sen arvoja. Muokkasimme arvoiksi viiden sekunnin viivytyksen, jonka avulla kerättyä tietoa pystyi paremmin tulkitsemaan. Myös muiden ”nodejen” mahdollisten arvojen muokkaaminen toimii samalla tavalla. Hyvänä esimerkkinä tästä on tietokannan ohjelmointi kappaleessa oleva kuva 13 jossa näkyy msg.payload ”noden” sisään luomamme funktio.