Tampereen ammattikorkeakoulu

Bluetooth-kommunikaatio Android-sovelluksella pelto-koneen kanssa

Kalle Lindevall

OPINNÄYTETYÖ
Toukokuu 2022

Tietojenkäsittelyn tutkinto-ohjelma
Web-palvelut

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Web-palvelut

LINDEVALL, KALLE:
Bluetooth-kommunikaatio Android-sovelluksella peltokoneen kanssa

Opinnäytetyö 40 sivua, joista liitteitä 0 sivua
Toukokuu 2022

Opinnäytetyön toimeksiantajana toimi ohjelmistokehitykseen keskittynyt Wapice Oy. Työn tavoitteena oli kehittää osa mobiilisovellusta, joka kommunikoi Bluetoothin välityksellä peltokoneeseen asennetun laitekappaleen kanssa. Peltokoneesta lähetetty ja käsitelty data näytettiin samalla sekä siihen yhdistetyn puhelimen ruudulla että myös Wapice Oy:n omassa IoT-TICKET-IoT-alustassa. Näin dataa pystyttiin seuraamaan myös etänä ja tarkastelemaan muodostuneita tilastoja. Kehityksessä käytettiin luottamuksellista tietoa, joten tietyt yksityiskohdat, kuten vastaanotetun datan tarkempi muoto, ei ilmene opinnäytetyössä.

Opinnäytetyön alussa käsitellään yleistä tietoa Bluetoothista ja vertaillaan sen eri versioita. Mobiilisovelluksen kehittämisestä työ kattaa Bluetooth-kommunikaation ja vastaanotetun datan käsittelyn osuuden. Tiedon jatkamisen IoT-alustaan toteutti toinen taho osana kyseistä mobiilisovelluskehitysprojektia.

Valmistunut mobiilisovellus kommunikoi onnistuneesti dataa lähettävän laitteen kanssa sekä lähetti saadun datan IoT-alustaan, josta seuraaminen onnistui mutkattomasti. Sovellusta on mahdollista jatkokehittää, muun muassa lisäämällä ominaisuus kommunikoida yhdistetyn laitteen kanssa, esimerkiksi toteuttaen yhteyden kannalta oleelliset alustukset.

Asiasanat: bluetooth, android, javascript, typescript, react native

ABSTRACT

Tampere University of Applied Sciences
Business Information Systems
Web Services

LINDEVALL, KALLE:
Bluetooth Communication with Agricultural Machinery in an Android Application

Bachelor's thesis 40 pages, appendices 0 pages
May 2022

The commissioner of the thesis was a software company, Wapice Ltd. The goal of the thesis was to develop a part of a mobile application which communicates with a device installed in an agricultural machine via Bluetooth. The received and later processed data from the machine is shown both on the receiving mobile application, and in Wapice Ltd's IoT-TICKET IoT platform. Thus, the data could be monitored remotely, and the charts generated based on it further observed. Some confidential information was also used during the development, so certain details, such as the specific format of the received data, do not appear in the thesis.

The thesis starts by introducing general information about Bluetooth and comparing its different versions. In the actual mobile application development part, the work covers the parts of Bluetooth communication and the handling of the received data. Forwarding the data to the cloud was carried out by another party as part of that mobile application development project.

The completed mobile application communicated with the device successfully and sent the data forward to the cloud, where it could be observed with ease. The application is suited for further development, such as, adding the feature to communicate with the connected device, in order to make the initialisations necessary for the connection, for example.

Key words: bluetooth, android, javascript, typescript, react native

SISÄLLYS

LYHENTEET JA TERMIT

backend	ohjelmistotuotannon termi, sovelluksen/verkkosivun datanhallintaan keskittynyt taso
BLE	Bluetooth Low Energy
Bluetooth	Langattomaan tiedonsiirtoon ja kommunikaatioon kehitetty radioaaltoja hyödyntävä tekniikka
Bluetooth Classic	Yleiskattava termi Bluetooth-versioille ennen Bluetooth Low Energy-version kehittämistä
Bluetooth Low Energy	Bluetoothin moderni, vähemmän virtaa kuluttava versio
CAN-väylä	Controller Area Network -väylä, mikro-ohjainten välillä käytetty kommunikointimenetelmä
frontend	Ohjelmistotuotannon termi, sovelluksen/verkkosivun käyttöliittymän ohjelmointiin keskittynyt taso
IoT	Internet of Things, esineiden internet. Useiden laitteiden kytkeminen internettiin, esimerkiksi monitorointia varten—älykellot, -valot, muut sensorit
JavaScript	Yksi suurimpia ohjelmointikieliä. Käytetään yleisimmin web-ohjelmoinnissa
pikoverkko	Bluetooth-laitteiden välisen yhteyden muodostama verkko

React	Metan, entisen Facebookin, ylläpitämä avoimen lähdekoodin JavaScript-kirjasto. Suoraviivaistaa ja mahdollistaa monimutkaisempien käyttöliittymien luomista
React Native	React-kirjaston mobiiliohjelmointiin keskittyvä versio
TypeScript	Microsoftin ylläpitämä, JavaScript-ohjelmointikielen päälle rakentuva ohjelmointikieli. Tarjoaa sovelluksiin muun muassa vahvan tyypityksen

1 JOHDANTO

Vaikka Bluetooth onkin nykyisin varmasti miltei jokaiselle tuttu käsite, on sen toimintaperiaatteet ja historia kuitenkin useimmille täysin vierasta. Opinnäytetyössä käsitellään Bluetoothin yleistä historiaa ja sen ominaisuuksia, sekä toteuttaa hiukan tarkempaa vertailua tämän aiempien versioiden ja nykyisen BLE-version välillä, muun muassa tuoden esiin näille soveltuvat käyttötarkoitukset.

Opinnäytetyössä on tekninen osuus, joka kattaa osuuden laajemmasta Android-mobiilisovelluskehitysprojektista. Opinnäytetyössä toteutetaan sovellukseen ominaisuus Bluetooth-kommunikaatiolle ja vastaanotettujen viestien käsittelylle oikeanlaiseen muotoon. Toteutus oli pääosin sovelluksen käyttöliittymään, frontendiin, painottuvaa.

Opinnäytetyön tekijä työskenteli opinnäytettä tehdessään ensimmäistä kertaa mobiiliohjelmoinnin parissa, hyödyntäessä Bluetoothia sovelluskehityksessä, ja ylipäätään perehtyessä kyseiseen viestintäteknologiaan tarkemmin, joten tämä vaikutti myös varsin sopivalta aiheelta opinnäytetyölle. Työn toimeksiantajana toimi sovelluskehitykseen keskittyvä, pääosin teollisuuden parissa toimiva, Wapice Oy.

2 BLUETOOTH

Bluetooth on edullinen ja virrankulutuksen kannalta kevyt, lyhyen kantaman, radioaaltoihin pohjautuva, langaton viestintäteknologia (Zeadally, Siddiqui & Baig 2019). Bluetoothia käytetään nykymaailmassa lukemattomissa eri laitteissa, yleisimpiä käyttötarkoituksia ovat esimerkiksi langattomat hiiret sekä kuulokkeet ja muu äänentoisto. Seuraavaksi perehdytään Bluetoothin yleiseen historiaan ja tämän toiminnallisuuksiin. Bluetoothin versioista tarkemmin myöhemmin omassa kappaleessaan.

2.1 Historia

Bluetoothin kehitys aloitettiin vuonna 1994 ruotsalaisella Ericssonilla, pääosin alankomaalaisen Jacobus "Jaap" Cornelis Haartsenin toimesta. Kehitys aloitettiin muun muassa korvaamaan vuonna 1960 käyttöönotettu langallinen RS-232 telekommunikaatio-standardi. (Triggs 2021.) Kehitys käynnistyi osittain myös siksi, koska olisi paljon suotuisampaa olla vain yksi keskeinen lyhyen kantaman langattoman viestinnän standardi, eikä useampaa erillään toimivaa—Intelillä oli käytössään tuolloin kehitteillä olevasta teknologiastaan nimitys ”Business-RF”, Ericssonilla ”MC-Link”, sekä Nokialla ”Low Power-RF”. (Kardach 2008.)

2.1.1 Bluetooth SIG

Välttääkseen teknologioiden turhan hajanaisuuden yhtiöt tapasivat vuonna 1996 Ericssonilla tavoitteenaan päästä sopuun yhteisen standardin kannalta, ja muodostivat Bluetooth Special Interest Groupin (SIG) tavoitteenaan ylläpitää Bluetoothin kehitystä. Muodostamishetkellä organisaation kuuluivat Ericsson, IBM, Intel, Nokia sekä Toshiba. (Kardach 2008.) Ensimmäisen vuoden loppuun mennessä organisaation perustamisesta siihen kuului jo yli 400 jäsentä. Nykyisin jäseniä on yli 30 000. (Pothitos 2017.)

2.1.2 Nimi ja logo

Bluetoothin nimi sekä logo sisältävät yllättävän mielenkiintoista historiaa. Bluetooth on saanut nimensä 900-luvulla eläneen viikinkikuningas Harald Sinihampaan mukaan. Hän yhdisti suuren osan Skandinaviaa, kuten Norjan ja Tanskan, yhtenäiseksi kuningaskunnaksi—kuten juuri kehitteillä olevan teknologian tavoitteena onkin yhdistää monet hajanaiset viestintäteknologiat yhtenäisiksi. Nimen oli tarkoitus alun perin olla vain väliaikainen. Virallisina vaihtoehtoina nimelle olivat PAN (Personal Area Networking) sekä RadioWire. Näistä ensimmäinen olikin jo laajemmin muussa käytössä, ja jälkimmäisen tavaramerkin rekisteröinnissä olisi kestänyt liian pitkään, joten nimi Bluetooth jäi lopulta ainoaksi osuvaksi vaihtoehdoksi. (Bluetooth SIG, Inc. 2022c.)

Myös Bluetoothin logolla on kytkökset viikinkikuningas Sinihampaaseen, sillä siihen on yhdistetty hänen nimikirjaimensa hagall (ᚼ) ja bjarkan (ᛒ), jotka kuuluvat nuoremman furtharkin riimumerkistöön (Bluetooth SIG, Inc. 2022c). Bluetoothin logo on esitetty kuvassa 1.

KUVA 1. Bluetooth-logo (Bluetooth SIG, Inc. 2022b).

2.2 Radio

Bluetooth toimii langattomasti radioaaltoja hyödyntäen, 2400–2483.5 MHz:n taajuudella. Taajuusalue kuuluu maailmanlaajuisesti avoimeen ISM-taajuusalueeseen (Industrial, Scientific and Medical), eli tämän käyttöön ei tarvita maakohtaisia erillisiä lupia. (Gupta 2016.)

Välttääkseen virhesignaalit muiden ISM-taajuusalueella toimivien laitteiden kanssa, Bluetooth hyödyntää jo vuonna 1942 esitettyä frequency-hopping spread spectrum (FHSS) -teknologiaan, eli taajuushyppelyä. Tätä hyödyntäen laitteiden välille muodostettu Bluetooth yhteys hyppii taajuudelta toiselle vältellen ruuhkaisimpia taajuuksia. (Pothitos 2017.) Taajuushyppelyyn Bluetoothilla on käytössään kanavia 79 kappaletta, jotka erotellaan 1 MHz välein. Hyppely tapahtuu 1600 kertaa sekunnissa puolisatunnaisesti. (Gupta 2016.)

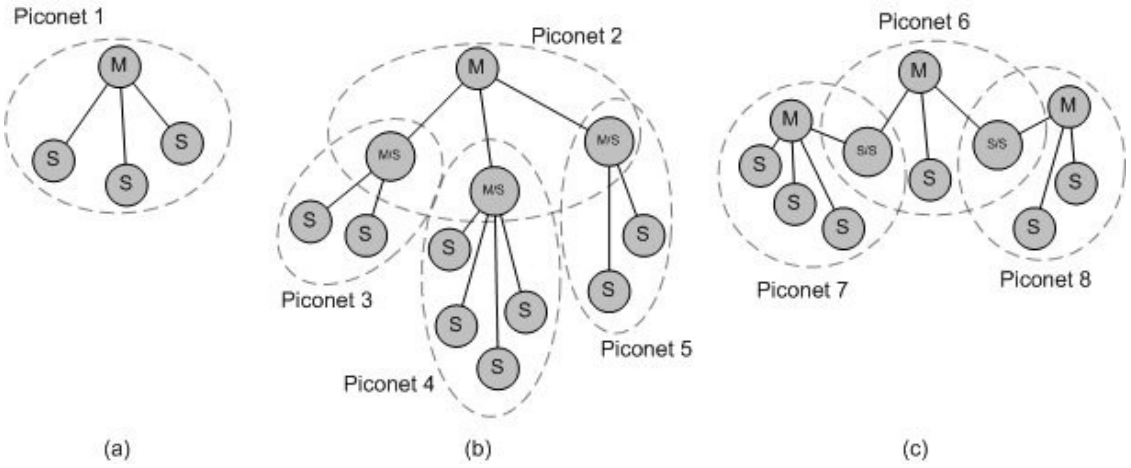
Bluetooth tukee yhteyksiä jopa sataan metriin asti, vaikkakin tätä tyypillisesti käytetäänkin huomattavasti lyhyemmillä kantamilla. Bluetooth-laitteet jaotellaan kolmeen eri luokkaan laitteen lähetystehon perusteella: mitä suurempi lähetysteho, sitä suurempi kantama. Ensimmäisen luokan laitteisiin kuuluvat laitteet, joilla on tehoa enintään 100 mW, toiseen luokkaan kuuluvilla 2.4 mW ja kolmanteen luokkaan kuuluvilla 1 mW. (Gupta 2016.)

2.3 Verkko

Bluetooth-laitteiden muodostamaa verkkoa kutsutaan pikoverkoksi (engl. piconet). Pikoverkossa toimii yksi isäntälaite (engl. master), johon voi olla yhdistettynä enimmillään seitsemän orjalaitetta (engl. slave). Yhteyden aloittava laite toimii muodostuneen pikoverkon mestarilaitteena. Laitteet voivat kuitenkin vaihdella roolejaan, riippuen siitä kuinka moneen pikoverkkoon nämä kuuluvat. Kommunikaatio useamman pikoverkon välillä muodostaa hajaverkon (engl. scatternet). (Zeadally ym. 2019.) Bluetooth-pikoverkkojen topologiasta tarkemmin omassa alaluvussaan.

Taajuushyppelyä hyödyntäen useammat yksittäiset Bluetooth-pikoverkot voivat toimia itsenäisesti lähellä toisiaan sotkematta toistensa yhteyksiä. Pikoverkkojen laitteet käyttävät yhteyksissään näiden mestarilaitteiden määrittämiä taajuuksia. (Gupta 2016.)

Kuviossa 1 esimerkkejä erilaisista piko- ja hajaverkoista. Kuviossa M-kirjain kuvastaa mestarilaitetta ja S-kirjain orjalaitetta. Kuvion esimerkki A on yksittäinen pikoverkko, jossa selkeät mestari- ja orjalaitteet. Esimerkki B on useamman pikoverkon mestari-/orjalaitteilla yhdistetty hajaverkko. Esimerkki C on useamman pikoverkon jakavilla orjalaitteilla yhdistetty hajaverkko.



KUVIO 1. Esimerkkejä piko- ja hajaverkoista (Fraser, James & Thiel 2007).

2.4 Topologia

Topologia, tässä tapauksessa verkkotopologia, on kuinka fyysiset ja loogiset laitteet sekä yhteydet ovat määritelty käytetyssä verkossa. Tietyt verkkotopologiat toimivat eri käyttökohteissa paremmin kuin toisissa, joten oikeanlaisen topologian rakentaminen on hyvin keskeistä verkon parhaimman suorituskyvyn ja käytettävyyden saavuttamisessa. (Gillis 2021.)

Kehityksen myötä vuosien saatossa, Bluetooth tukee nykyisin kolmea erilaista verkkotopologiaa näin soveltuen moniin eri käyttötarkoituksiin. Bluetoothia voidaan hyvin hyödyntää vain kahden laitteen väliseen yhteyteen tai laajempaan useamman laitteen väliseen verkkoon.

Kahden laitteen muodostavassa kaksisuuntaisessa (1:1) verkossa käytetään point-to-point-topologiaa. Tämä soveltuu moneen eri käyttötarkoitukseen yksinkertaisuutensa ansiosta, kuten kaiuttimiin ja kuulokkeisiin, sekä myöhemmin

Bluetooth Low Energy -version parannuksien myötä muun muassa seuranta-antureihin ja PC-oheislaitteisiin. Verkon muodostaessa yksi ja useampi laite, käytetään kaksisuuntaista one-to-many (1:m) broadcast-topologiaa. Tämä mahdollistaa tiedon lähettämisen useammalle Bluetooth-laitteelle verkossa, soveltuen muun muassa paikkakohtaisten tietojen lähettämiseen. Verkon muodostaessa moni useamman laitteen verkko, käytetään many-to-many (m:m) mesh-topologiaa, mahdollistaen suurien verkkojen muodostamisen. Tämä soveltuu muun muassa seurantaan ja hallinnointiin—tilanteisiin, joissa jopa tuhansien laitteiden täytyy kommunikoida keskenään luotettavasti. Mesh-topologia on käytettävissä Bluetooth Low Energy versioissa. (Bluetooth SIG, Inc. 2022d.)

2.5 Tiedonsiirto

Bluetooth-laitteiden välinen viestintä tapahtuu tämän 79:ää eri kanavaa pitkiin, pikoverkon isäntälaitteen säätelemää taajuushyppelyä hyödyntäen, kanavan muuttuessa 1600 kertaa sekunnissa. Itse tiedonsiirto tapahtuu lähettämällä datasta pilkotun yksittäisen datapaketin lähettäminen tietyssä aikaikkunassa, slotissa. (Gupta 2016.) Yhteyden yksi kellosykli on 312.5 mikrosekuntia ja tiedonsiirtoon käytetty slot on kaksi sykliä, 625 mikrosekuntia. Tiedonsiirto synkronoidaan isäntälaitteen käyttämän kellon mukaan. (Zeadally ym. 2019.)

Datapaketti voidaan lähettää tämän koon mukaan joko yhden, kolmen tai viiden slotin mittaisena. Mestarilaitteet lähettävät paketteja aina parillisilla sloteilla ja vastaanottavat aina parittomilla sloteilla ja orjalaitteet päinvastoin. Orjalaitteet siirtävät datan pikoverkossa aina mestarilaitteen kautta. (Gupta 2016.)

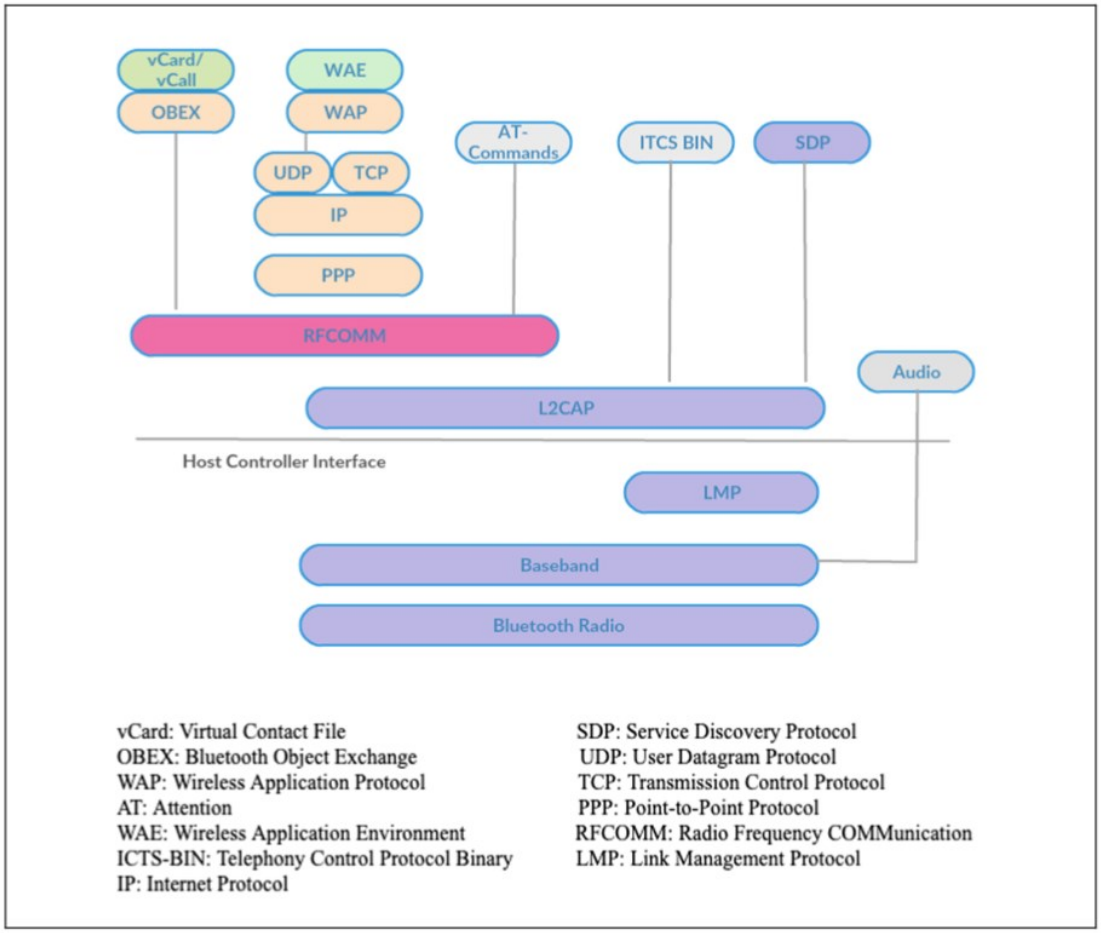
Bluetoothin ensimmäisissä versioissa tiedonsiirtonopeus, eli niin sanottu Basic Rate (BR), oli enintään 721 Kbps. Bluetooth 2.0 version myötä saapui Enhanced Data Rate (EDR), joka mahdollisti tuen jopa 2.1 Mbps yhteysnopeuksille. Bluetooth 3.0 version High Speed (HS) mahdollisti 24 Mbps nopeudet. (Gupta 2016.) Bluetoothin versioista vielä tarkemmin myöhemmin omassa kappaleessaan.

2.6 Protokollat

Bluetooth koostuu useammista protokollista, eli käytännössä kasoista erilaisia sääntöjä/standardeja, joiden pohjalta Bluetooth-laitteet määrittelevät kuinka toimia pikoverkossa (Zeadally ym. 2019). Poiketen OSI- ja TCP/IP-verkkojen toimintatavoista, Bluetooth-laitteiden ei tarvitse hyödyntää jokaista protokollaa yhteyksissään (Gopikrishnan 2020).

Bluetoothin protokollat voidaan jaotella kahteen eri tasoon, alempaan niin sanottuun controller (ohjain) -tasoon, sekä ylempään host (isäntä) -tasoon, joita jakaa Host Controller Interface (HCI). Alemman tason protokollat ovat vastuussa Bluetoothin toiminnan kannalta keskeisistä ominaisuuksista, kuten uusien Bluetooth-laitteiden etsinnästä ja yhteyden muodostamisesta. Host-tason protokollat puolestaan hyödyntävät alemman tason protokollia monimutkaisempien toiminnallisuuksien toteuttamiseen, kuten esimerkiksi isompien datapakettien purkamiseen ja uudelleen kokoamiseen sekä musiikin suoratoistoon. (Gupta 2016.)

Bluetooth pääosin operoi seitsemää protokollaa hyödyntäen: Radio protokolla, Baseband protokolla, Radio Frequency Communication (RFCOMM), Service Discovery Protocol (SDP), Link Management Protocol (LMP), Logical Link Control and Adaptation Layer Protocol (L2CAP) ja Host Controller Interface (HCI) protokolla. (Zeadally ym. 2019.) Bluetoothin protokollapino esitelty kuvassa 2.



KUVA 2. Bluetoothin protokollapino (Zeadally ym. 2019).

Aiemmin mainituista seitsemästä protokollasta, protokollapinon alimmainen ja tärkein protokolla, Bluetooth Radio, on vastuussa lähetettävien radioaaltojen hallinnoimisesta. Se myös määrittää muun muassa yhteydessä käytetyt taajuudet sekä taajuushyppelyn tarkemmat tiedot. Tästä ylempi protokolla, Baseband, hoitaa Bluetooth-laitteiden etsinnän sekä tiedonkäsittelyn ja mestari-/orjalaitteiden määrityksen. Seuraava protokolla, Link Management Protocol (LMP), on vastuussa yhteyden muodostamisesta, hallinnoimisesta, varmentamisesta sekä suojauksesta. (Gopikrishnan 2020.) Host Controller Interface (HCI), vaikkakin valinnainen ja vaadittu ainoastaan Bluetooth-toteutuksissa, joissa host- ja controller-tasot toimivat omilla prosessoreillaan, on eri kerrosten välisessä viestinnässä erittäin olennainen protokolla. Kommunikaatio HCI:tä pitkin tapahtuu pakettimuotoisena, ja host-tason lähettäessä viestin alaspäin, vastaa controller-taso tähän asynkronisesti. (Gupta 2016.) Logical Link Control and Adaptation Protocol (L2CAP) toimii alemman ja ylemmän protokollatason välissä, huolehtien näiden keskinäisestä viestinnästä. Ylemmän tason protokol-

la Service Discovery Protocol (SDP) mahdollistaa erilaisten palvelujen löytämisen yhdistetystä Bluetooth-laitteesta. Radio Frequency Communications (RFComm) protokollan avulla Bluetooth-laite kykenee emuloimaan virtuaalisen sarjaportin yhteyksiinsä, täten poistaen tarpeen kaapeleilta. (Gopikrishnan 2020.)

2.7 Profiilit

Bluetooth-laitteet hyödyntävät erilaisia profiileja, jotka määrittelevät mitä protokolleja laitteiden kuuluisi käyttää näiden välisessä yhteydessä. Yhteyden molemmilla laitteilla pitää olla käytössään sama profiili. Kaikki laitteet eivät tue kaikkia profiileja. (Sony Electronics Inc. 2022.) Esimerkkejä erilaisista profiileista taulukossa 1.

TAULUKKO 1 Erilaisia Bluetooth-profiileja (Sony Electronics Inc. 2022).

Profiilin nimi	**Ominaisuudet**	**Käyttötarkoitukset**
Advanced Audio Distribution Profile (A2DP)	Musiikin suoratoisto	Kuulokkeet, kulkuneuvojen laitteisto, tietokoneet, mobiililaitteet
Basic Imaging Profile (BIP)	Kuvien siirtäminen	Mobiililaitteet, tulostimet
Human Interface Device Profile (HID)	Laitteiden langaton yhdistäminen	Hiiret, näppäimistöt, mobiililaitteet
Object PushProfile (OPP)	Tiedonsiirto laitteiden välillä	Mobiililaitteet, tietokoneet, kulkuneuvojen laitteisto
Personal Area Network Profile (PAN)	Pienenerkon muodostaminen	Tietokoneet, tukiasemat, mobiililaitteet

2.8 Tietoturva

Jo Bluetoothin alkuvaiheilta asti on tietoturva ollut yksi keskeisimpiä asioita tämän kehityksessä. Bluetoothin monikäyttöisyys on samalla sekä hyöty että haitta, sillä joustavan tietoturvallisuuden rakentaminen samalla hankaloituu—mikä sopii parhaiten ja mihin käyttöympäristöön? (Gehrmann, Persson & Smeets 2004.) Ohessa lyhyt yleiskatsaus Bluetoothin tietoturvapalveluihin.

Bluetooth-standardiin kuuluu yhteensä viisi erilaista tietoturvan peruspalvelua: todentaminen (engl. authentication), luottamuksellisuus (engl. confidentiality), valtuuttaminen (engl. authorization), viestien yhtenäisyys (engl. message integrity) sekä parittaminen/kiinnittäminen (engl. pairing/bonding). Todentamisella vahvistetaan yhteydessä olevat laitteet näiden osoitteiden mukaan. Luottamuksellisuudella varmistetaan tiedon yhtenäisyys huolehtimalla, että vain lähetettyyn tietoon valtuutetut laitteet pääsevät tähän käsiksi. Valtuuttamisella hallitaan, että laitteet voivat käyttää vain näille oikeutettuja palveluita ja resursseja yhteydessä. Viestien yhtenäisyydellä pidetään huolta, että laitteiden välillä lähetetyt viestit eivät ole muuttuneet siirron aikana. Parittamisella muodostetaan talteen yhteyksille omat salaiset avaimet, joidenka pohjalta laitteet kykenevät tunnistamaan toisensa myöhemmissä yhteyksissä. (Padgette ym. 2017.)

3 BLUETOOTH-VERSIOT

Julkaisustaan vuonna 1999 Bluetooth on kehittynyt vuosi vuodelta yhä tehokkaammaksi ja luotettavammaksi. Keskeisimpiä parannuksia on tullut muun muassa yhteysnopeuksiin, tietoturvaan sekä virrankulutukseen. Seuraavaksi yleiskatsaus Bluetoothin eri versioihin vuosien varrelta.

3.1 Bluetooth 1.0–1.2

Vaikkakin Bluetooth 1.0 julkaistiin vuonna 1999, ensimmäinen tätä hyödyntävä matkapuhelin tuotiin markkinoille vasta 2001. Bluetooth 1.0, tarkemmin markkinoille tuodun puhelimen käyttämän 1.0b version, eli niin sanotun Basic Raten (BR), tiedonsiirtonopeus oli enintään 732.2 Kbps ja kantama 10 metriä. Bluetooth-versio 1.1 tuotiin markkinoille vuonna 2001 mukanaan muun muassa tuki useamman laitteen yhteyksille sekä yhteyksille salaamattomia kanavia pitkin. Vuonna 2003 tulleen versio 1.2:n myötä saapui yleisten parannuksien lisäksi Adaptive Frequency Hopping (AFH), joka vähensi Bluetoothin sotkeutumista muiden taajuuksien kanssa, parantaen yhteyksien laatua. (Hildenbrand & Maring 2019; Zeadally ym. 2019.)

3.2 Bluetooth 2.0–2.1

Bluetoothin versio 2.0 julkaistiin vuonna 2004 (Zeadally ym. 2019). Uusi versio toi mukanaan Enhanced Data Rate (EDR) -ominaisuuden, joka muun muassa mahdollisti teoreettisen 3.0 Mbps tiedonsiirtonopeuden. Versio 2.0 ylti käytännössä kuitenkin vain 2.1 Mbps nopeuksiin, joka kuitenkin oli yli kaksinkertainen aiempiin nopeuksiin verrattuna. EDR paransi myös Bluetoothin virrankulutusta. Versio 2.1 julkistettiin vuonna 2007, joka suoraviivaisti laitteiden parittamista. (Hildenbrand & Maring 2019.)

3.3 Bluetooth 3.0

Vuonna 2009 markkinoille tuotu versio 3.0 mahdollisti entistä nopeamman tiedonsiirtonopeuden tämän high-speed (HS) -väylää pitkin. Tämän avulla Bluetooth kykeni teoreettisesti jopa 24 Mbps nopeuksiin, mutta tämä vaati kuitenkin avukseen Wi-Fi-yhteyden. (Hildenbrand & Maring 2019.)

3.4 Bluetooth 4.0–4.2

Versio 4.0 julkistettiin kesällä 2010 ja tuotiin markkinoille saman vuoden lokakuussa uuden Apple iPhone 4S -älypuhelimen myötä. 4.0-version suurena pääpainona oli tämän uusi, entistä vähemmän virtaa kuluttava, Bluetooth Low Energy (BLE) -teknologia. (Hildenbrand & Maring 2019.) LE-teknologia pohjautuu vähävirtaiseen langattomaan Wibree-teknologiaan, jota oli kehittämässä muun muassa Nokia. Wibree yhdistyi Bluetooth SIG:n alle vuonna 2010. Versiota kutsuttiin alun perin Bluetooth Smartiksi. (Get Connected Blog 2021.) Tiedonsiirtonopeudet pysyivät ennallaan, mutta yhteydet suojattiin nyt 128-bittisellä salauksella. Yksisuuntaiset yhteydet olivat nyt mahdollisia BLE:n avulla, kuten auttamaan sisätilojen navigoinnissa, mutta myös tämän huonompana puolena, seuranta- ja mainostietoja voitiin näin kerätä ilman käyttäjän suostumusta. Versio 4.1 julkaistiin loppuvuodesta 2013, parantaen Bluetooth-laitteiden kommunikaatiota 4G-radiosignaalien seassa, sekä mahdollistaen laitteiden toimivan samaan aikaan eri rooleissa, näin helpottaen laitteiden välistä samanaikaista kommunikaatiota. Vuoden päästä joulukuussa 2014 julkaistu versio 4.2 ehosti Bluetoothin tietoturvaa sekä esineiden internetin (engl. Internet of Things, IoT) toiminnallisuuksia. (Hildenbrand & Maring 2019.)

3.5 Bluetooth 5.0–5.3

Vuonna 2016 julkaistu Bluetoothin versio 5.0, SIG:n brändäämänä pelkkä ”Bluetooth 5”, oli iso edistysaskel Bluetooth-teknologiassa. Bluetoothin kantama parani huomattavasti, sekä yhteysnopeudet yltivät nyt jopa 50 Mbps nopeuksiin.

Myös esineiden internet oli keskeisessä osassa 5.0 päivitystä, muun muassa ehostaen virrankulutusta ja laitteiden yhteyksiä. (Hildenbrand & Maring 2019.) Vuonna 2019 julkistettu versio 5.1 paransi Bluetoothin paikallistamiseen liittyviä ominaisuuksia, kuten tunnistaa toisen laitteen suunta, esimerkiksi löytämään langattomat kuulokkeet. Tämä oli mahdollista kahta eri tapaa hyödyntäen, Angle of Arrival (AoA) sekä Angle of Departure (AoD). Ensimmäistä hyödyntämällä laite tutkii, yhden antennin lähettämää, vastaanotettua signaalia useamman antennin avulla, näin päätellen saadun signaalin tarkan suunnan. Jälkimmäinen toimii käytännössä päinvastoin: paikallistettava laite lähettää signaalin useampaa antennia hyödyntäen, ja tämä voidaan paikallistaa yhdellä antennilla tutkimalla vastaanotettujen signaalien eroavaisuudet. Loppuvuodesta 2019 Bluetooth SIG julkisti Bluetooth 5.2 -version, jonka tavoitteena oli pääosin parantaa Bluetoothin toiminnallisuuksia langattomien kuulokkeiden ja kaiuttuimien kanssa. Uutena ominaisuutena oli EATT-protokolla (Enhanced Attribute Protocol), joka mahdollisti useammat samanaikaiset lähetykset. Tämä lyhensi yhteyksien viiveitä ja aikaa, jossa yhteys vakautuu. Toinen uusi ominaisuus oli LE Power Control (LEPC), joka paransi virrankulutusta entuudestaan, antaen laitteille tuen dynaamiselle lähetystehon säätelylle. Kolmas uusi ominaisuus oli LE Audio, joka paransi Bluetooth-yhteyksien äänenlaatua. Kesällä 2021 julkistettu, Bluetoothin nykyinen, 5.3-versio keskittyi pääosin IoT-parannuksiin muun muassa nopeuttaen Bluetooth-laitteen siirtymää korkeamman suorituskyvyn tilasta alhaisempaan tilaan ja päinvastoin. (Gupta 2021.)

3.6 Bluetooth Classic ja BLE

Ennen versio 4.0:aa ei ollut tietoakaan Bluetooth Classicista tai muista erillisistä nimityksistä, mutta 4.0-version myötä saapunut Low Energy -teknologia jakoi teknologian kahteen pääversioon omine eroavaisuuksineen ja käyttötarkoituksineen: 4.0:aa edeltävä niin sanottu perinteinen (classic) Bluetooth ja tätä seuraava Bluetooth Low Energy. (Get Connected Blog 2021.)

Bluetooth Classicia käytetään nykyisin pääosin äänen suoratoistamiseen, eli toisin sanoen jatkuvaan tasaiseen datan lähetykseen, kun taas BLE keskittyy

pääosin yksittäisten IoT-laiteiden hetkittäisiin yhteyksiin. Classicia voidaan toki myös hyödyntää IoT-laitteiden parissa, mutta BLE soveltuu tähän huomattavasti paremmin pienemmän virrankulutuksensa ansiosta. Teknologiat poikkeavat myös huomattavasti topologioiltaan. Bluetooth Classic hyödyntää vain kahden laitteen välistä kaksisuuntaista point-to-point-topologiaa. BLE kykenee hyödyntämään tämän lisäksi myös one-to-many-topologiaa sekä many-to-many-topologiaa. Bluetoothin topologioista tarkemmin aiemmassa omassa kappaleessaan. (Get Connected Blog 2021.) Bluetooth Classicin yhteydet mahdollistavat hiukan laajemman väylän datan kulkuun korkeintaan 3 Mb/s bittinopeudella, BLE tukien väylää vain 2 Mb/s asti (Bluetooth SIG, Inc. 2022a). Bluetooth Classic ja BLE välisiin teknisempiin yksityiskohtiin perehtyvää vertailua kuvassa 3. Kaikkia kuvassa käsiteltäviä aiheita tai termejä ei ole käsitelty opinnäytetyössä.

	Bluetooth Low Energy (LE)	Bluetooth Classic
Frequency Band	2.4GHz ISM Band (2.402 – 2.480 GHz Utilized)	2.4GHz ISM Band (2.402 – 2.480 GHz Utilized)
Channels	40 channels with 2 MHz spacing (3 advertising channels/37 data channels)	79 channels with 1 MHz spacing
Channel Usage	Frequency-Hopping Spread Spectrum (FHSS)	Frequency-Hopping Spread Spectrum (FHSS)
Modulation	GFSK	GFSK, π/4 DQPSK, 8DPSK
Data Rate	LE 2M PHY: 2 Mb/s LE 1M PHY: 1 Mb/s LE Coded PHY (S=2): 500 Kb/s LE Coded PHY (S=8): 125 Kb/s	EDR PHY (8DPSK): 3 Mb/s EDR PHY (π/4 DQPSK): 2 Mb/s BR PHY (GFSK): 1 Mb/s
Tx Power*	≤ 100 mW (+20 dBm)	≤ 100 mW (+20 dBm)
Rx Sensitivity	LE 2M PHY: ≤-70 dBm LE 1M PHY: ≤-70 dBm LE Coded PHY (S=2): ≤-75 dBm LE Coded PHY (S=8): ≤-82 dBm	≤-70 dBm
Data Transports	Asynchronous Connection-oriented Isochronous Connection-oriented Asynchronous Connectionless Synchronous Connectionless Isochronous Connectionless	Asynchronous Connection-oriented Synchronous Connection-oriented
Communication Topologies	Point-to-Point (including piconet) Broadcast Mesh	Point-to-Point (including piconet)
Positioning Features	Presence: Advertising Direction: RSSI, HADM(Coming) Distance: Direction Finding (AoA/AoD)	None

* Devices shall not exceed the maximum allowed transmit power levels set by the regulatory bodies that have jurisdiction over the locales in which the device is to be sold or intended to operate. Implementers should be aware that the maximum transmit power level permitted under a given set of regulations might not be the same for all modulation modes.

KUVA 3 Bluetooth Low Energy ja Classic eroavaisuuksia (Bluetooth SIG, Inc. 2022a).

4 KEHITYSYMPÄRISTÖ

Tässä kappaleessa käsitellään opinnäytetyön käytännönosuuden projektin kehitysympäristöä. Kehitysympäristö kattaa niin työkalut ja alustat, kuin käytetyt ohjelmointikielet ja teknologiat. Käytännön osuus kehitettiin Android-alustalle Windows 10 -käyttöjärjestelmällä, joten esimerkiksi iOS-laitteelle ohjelmointi voi poiketa käytetyistä välineistään. Itse opinnäytetyö kirjoitettiin Microsoft Office Word -tekstinkäsittelyohjelmalla.

4.1 Projektissa käytetyt työkalut

Opinnäytetyön tekninen osuus toteutettiin pääosin Visual Studio Codea käyttämällä. Android Studiota käytettiin mahdollisiin Android-ohjelmointiin liittyviin paikkauksiin. Projektissa käytettiin opinnäytetyön toimeksiantajayrityksen projektikäytänteitä, kuten Git-versionhallintaa sekä BitBucket-alustaa Gitin hallinnointiin.

4.1.1 Visual Studio Code

Visual Studio Code (VS Code) on Microsoftin kehittämä avoimen lähdekoodin tekstinkäsittelyohjelma Windows-, Linux- sekä macOS-käyttöjärjestelmille. Ohjelma on kevyt ja hyvin muokattavissa oleva ladattavien lisäpakettien ansiosta, samalla tukien useita suurimpia ohjelmointikieliä. Näin käyttäjällä on hyvät mahdollisuudet tämän mukauttamiselle omiin käyttötarkoituksiinsa. (Microsoft 2022b.) Projektin koodi tehtiin pääosin VS Codella, lukuun ottamatta Android Studiolla tehtyjä tarvittavia muutamia muutoksia.

4.1.2 Android Studio

Android Studio on Android-mobiilisovelluksien kehitykseen keskittynyt tekstinkäsittelyohjelma. Visual Studio Coden monipuolisuuteen verrattuna Android Studio on keskittynyt pelkästään mobiiliohjelmointiin. Android Studiolla myös ominaisuus Android-laitteiden emuloinnille. (Android 2022.) Opinnäytetyön teknisen osuuden sovelluskehityksessä Android Studiota käytettiin Java-ohjelmointikieltä hyödyntämällä tiettyihin tarvittaviin muutoksiin. Muutoksista tarkemmin mobiilisovelluksen kehitykseen keskittyneessä kappaleessa.

4.1.3 Git, BitBucket

Git on versionhallintaohjelma. Tämä mahdollistaa nopean ja luotettavan lähdekoodin tallettamisen keskeiseen palvelimelle säilöttyyn hakemistoon, repositorioon, josta käyttäjät pystyvät kopioimaan itselleen paikallisen version muokattavakseen. Säilötyn koodin versioita pystytään hallitsemaan ja seuraamaan, joten esimerkiksi koodin viimeisimmän version mahdollisesti rikkoessa tietyn ominaisuuden ohjelmassa, voidaan aikaisempi versio aina palauttaa. (Spinellis 2012.) Projektissa käytettiin Atlassian-alustaan kuuluvaa Bitbucket-työkalua Git repositorioiden hallintaan (Atlassian 2022).

4.1.4 Laitteisto

Kehitystyö toteutettiin opinnäytetyön toimeksiantajalta saatua Samsung Galaxy S7 -älypuhelinta hyödyntäen, mahdollistaen sovelluksen testauksen fyysisellä laitteella pelkän tietokoneella käytettävän Android-emulaattorin sijaan. Työssä käytettiin myös toimeksiantajan asiakkaalta saatua fyysistä Bluetooth-simulaattoria, joka simuloi peltokoneelta lähetettävää dataa. Molemmat palautettiin opinnäytetyön toimeksiantajalle jo ennen opinnäytetyön kirjoitushetkeä, joten tarkempia tietoja, kuten puhelimen Android-käyttöjärjestelmäversiota, ei voitu tarkistaa.

4.2 Projektissa käytetyt tekniikat

Kehitystyössä käytettiin pääosin TypeScript -ohjelmointikieltä, mobiilisovelluskehitykseen tarkoitettua React Native -lisäkirjastoa hyödyntäen. Myös Java-ohjelmointikieltä käytettiin Android-ohjelmoinnissa, mutta tämä oli niin pienessä osassa kehitystyötä, joten tästä ole ei tarkempaa osiota tässä kappaleessa.

4.2.1 TypeScript

TypeScript (lyhennettynä TS) on ohjelmointikieli, joka rakentuu JavaScript-ohjelmointikielen päälle. Tavalliseen JavaScriptiin verrattuna TypeScript tarjoaa muun muassa ominaisuuden tyypitykselle. Tyypityksellä tarkoitetaan, kun koodissa käytettävät muuttujat määritetään tiettyyn muotoon, esimerkiksi numeroksi, kirjaimeksi tai kirjainjonoksi, minkä jälkeen näitä ei voida enää muuttaa toiseen. JavaScriptissä tyypityksen puute voi lopulta johtaa virheisiin koodissa: muuttuja, jolle on määritetty arvoksi numero, voidaan kuitenkin määrittää uudestaan esimerkiksi antamalle tälle arvoksi kirjaimen. TypeScriptillä kirjoitettu koodi kääntyy tätä ajaessa JavaScript-muotoiseksi, jonka ohjelma ymmärtää. Koodia kääntäessä TypeScript suorittaa virheentarkistuksia. (Microsoft 2022a.)

4.2.2 React Native

React Native on lisäkirjasto Java-/TypeScript-ohjelmointikielillä toteutettaviin frontend-kehitysprojekteihin. React Native pohjautuu suosittuun JavaScript-lisäkirjastoon, Reactiin. React Native mahdollistaa monipuolisempien ja laajempien käyttöliittymien toteuttamisen, sekä Android- että iOS-ympäristöissä. Tällä toteutettu sovellus hyödyntää mobiililaitteen käyttöliittymää täysin. Selainpohjaisella sovelluksella saattaa olla ongelmia esimerkiksi mobiililaitteen ruutuun skaalautumisessa tai kaikkia laitteen ominaisuuksia hyödyntäessä—eli niin sanotuissa natiiveissa toiminnallisuuksissa. Sovellus päivittyy reaaliajassa, kun koodiin tehdyt muutokset tallennetaan. React Nativea on käytetty muun muassa Instagramin, Skypen ja Teslan mobiilisovelluksissa. (Meta Platforms, Inc. 2022.)

Opinnäytetyön mobiilisovellus toteutettiin React Nativella. Sovellusta kehitettiin Android Studion Android-emulaattoria hyödyntämällä ja Samsung Galaxy S7 -älypuhelimella micro-USB-kaapelin välityksellä.

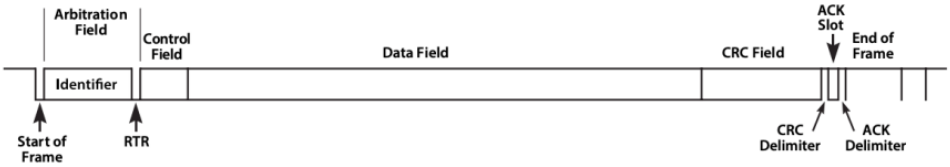
5 ANDROID-MOBIILISOVELLUS

Mobiilisovelluskehitysprojekti opinnäytetyön toimeksiantajayrityksessä käynnistettiin keväällä 2021 osana laajempaa hanketta. Projekti jatkui saman vuoden kesään asti, jolloin sovelluksen valmistuttua toimeksiantajan asiakas testautti tätä omilla mobiililaitteillaan ja peltokoneillaan. Opinnäytetyössä käsitellään sovelluksen pelkkiä Bluetooth-kommunikaatioon liittyviä toiminnallisuuksia, itse sovelluksen käyttöliittymän ja IoT-toiminnallisuuden toteutti toinen taho osana mobiilisovelluskehitysprojektia. Valmistunut sovellus toimitettiin asiakkaalle ladattavana APK-tiedostona (Android application PacKage). Sovellusta jouduttiin päivittämään asiakkaan kohdatessa ohjelmointivirheitä tai muita ongelmia sovelluksen käytössä. Valmistunutta sovellusta on vielä mahdollista jatkokehittää. Kuvia valmistuneen sovelluksen käyttöliittymästä ei valitettavasti esiinny opinnäytetyössä. Tässä kappaleessa käsitellään tarkemmin sovelluskehitysprojektin kulusta ja yksityiskohdista. On kuitenkin huomioitavaa, että projektissa käytettiin muun muassa projektin asiakkaalta saatua luottamuksellista aineistoa, joten tietyt yksityiskohdat on jätetty mainitsematta. Myös projektin asiakkaan sekä hankkeen nimet on jätetty mainitsematta.

5.1 Projektin pohjustus

Projektissa kehitettävältä sovellukselta kaivattiin langattoman yhteyden muodostamista ja viestintää mobiililaitteella peltokoneeseen kytkettävään laitekappaleen kanssa. Laite viestisi puhelimelle CAN-väylää (Controller Area Network) hyödyntämällä, jonka viestit olisivat heksadesimaalijärjestelmän muodossa. Heksadesimaalijärjestelmässä numerot lasketaan myös A-F kirjaimia hyödyntämällä, sisältäen yhteensä 16 numeroa nolla mukaan lukien, tavallisen desimaalijärjestelmän kymmenen luvun sijaan. (Kvaser 2022.) Projektissa vaihtoehtona viestinnälle oli suunniteltu Wi-Fiä Bluetoothin lisäksi, mutta Bluetooth oli lopulta suotuisampi toteutuksen ja käytettävyyden kannalta.

CAN-väylä on mikro-ohjainten, esimerkiksi kulkuneuvon moottorin ja muiden anturien, välillä käytetty kommunikointimenetelmä. CAN-väylän viestiketjut lähetetään jokaiselle laitteeseen kytketylle anturille. Viestit aloitetaan tunnisteella, joka määrittää anturit, joiden kuuluu vastaanottaa kyseinen viesti. Tunnistetta seuraa itse viestin runko, ja lopuksi muun muassa CRC-virheentarkistus sekä sensorin takaisin lähettämä viesti vastaanotetun viestin hyväksymisestä. Viestien eri osuudet koostuvat eri määristä bittejä ja tavuja. Näiden pohjalta voidaan purkaa viestien eri osuudet ja sisältö, tässä tapauksessa opinnäytetyön toimeksiantajan asiakkaan laatiman spesifikaation mukaan. (Kvaser 2022.) Esimerkki CAN-viestin rakenteesta esitetty kuvassa 4. Mobiilisovelluskehitysprojektissa käytettyä CAN-datan tarkempaa muotoa, kuten mitkä viestin osuudet vastaavat mitä luettavaa dataa, ei ilmoiteta opinnäytetyössä.



KUVA 4 Esimerkki CAN-väylän viestin rakenteesta (Kvaser 2022).

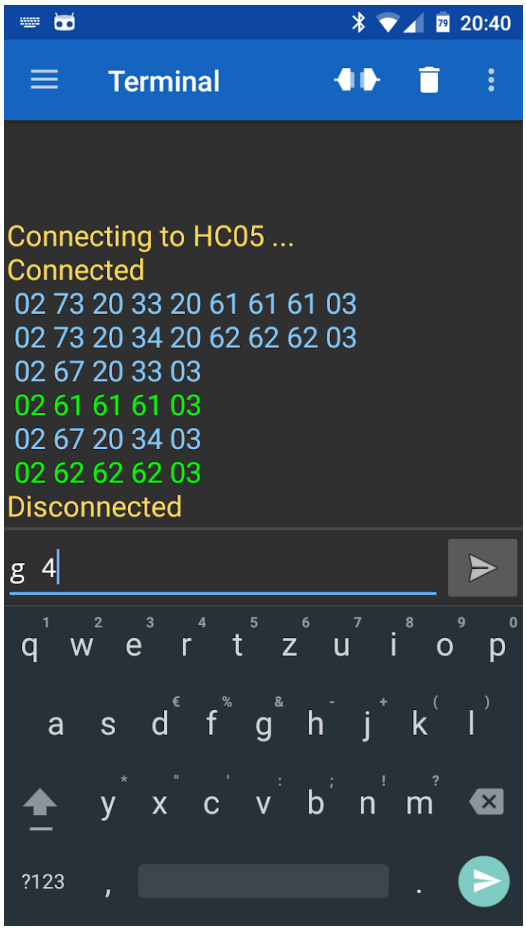
5.2 Bluetooth-kommunikaatio

React Nativeen on saatavilla useampia Bluetooth-kommunikaatioon keskittyviä lisäkirjastoja. Useimmat näistä kuitenkin käsittelevät BLE-kommunikaatiota, eivätkä mobiilisovelluskehitysprojektin kannalta kaivattua Bluetooth Classicia. Syitä BLE:n suosiolle myös sovelluskehityksessä voidaan arvioida johtuvan monien kehitettävien Bluetooth-sovelluksien käyttötarkoituksien keskittymisestä IoT-toiminnallisuuksiin—lyhyisiin ja nopeisiin yhteyksiin. Samalla myös alhaisempi virrankulutus soveltuu hyvin mobiilisovelluskehitykseen.

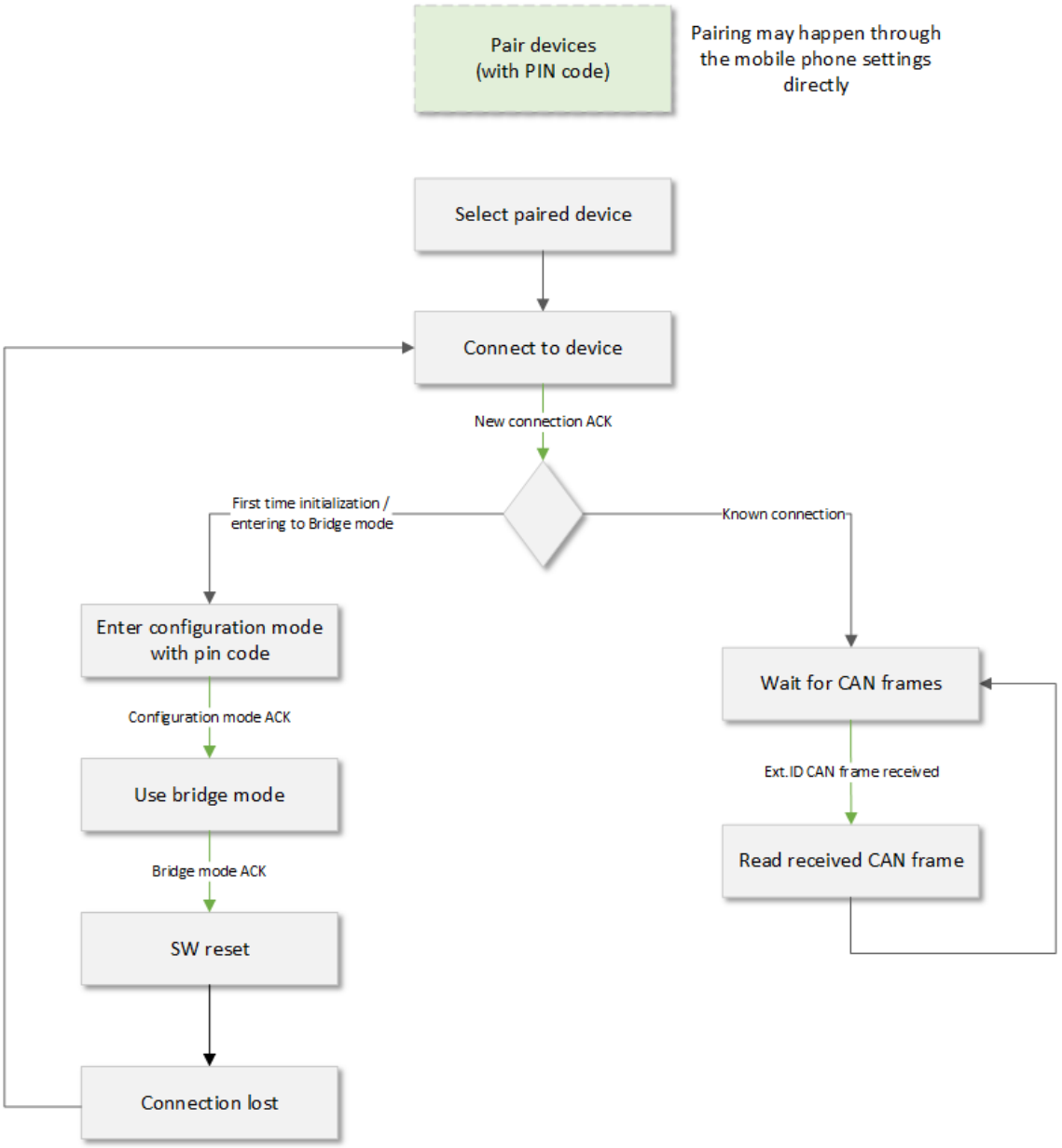
Bluetooth Classic -kommunikaatioon React Nativessa löytyi yksi soveltuva lisäkirjasto, React Native Bluetooth Classic. Kirjastolla on kattava dokumentaatio Android-käyttöjärjestelmän kannalta, mutta kirjaston iOS-laitteisiin keskittyvä

dokumentaatio on suppeampi. Dokumentaatio löytyy kirjaston kotisivuilta. (Davidson n.d.) Mobiilisovelluskehitysprojektin Bluetooth-toiminnallisuus toteutettiin pääosin hyödyntäen React Native Bluetooth Classic -kirjaston valmista demosovellusta, johon tehtiin projektin kannalta tarvittavia muutoksia.

Jotta Android-sovellus kommunikoisi tähän yhdistettävän Bluetooth-simulaattorin, ja myöhemmin peltokoneeseen asennetun laitekappaleen, kanssa oikein, laitteelle piti tehdä aluksi parituksen ja viestinnän kannalta olennaiset konfiguraatiot ja alustukset. Jo tässä vaiheessa tuli valitettavasti yksi React Native Bluetooth Classic -kirjaston ongelmista vastaan: kirjasto lähetti viestit yhdistettyyn Bluetooth-laitteeseen tekstimuodossa ja projektissa käytetyt laitteet ymmärsivät vastaanotetut viestit vain heksadesimaalimuotoisina CAN-viestiketjuina. Kirjastossa ei ollut virallista tukea heksadesimaalimuotoiselle viestinnälle, joten vaadittavat asetukset tehtiin Google Play -kaupasta ladattavalla Serial Bluetooth Terminal -Android-sovelluksella. Kuvassa 5 on esimerkkikuva mobiilisovelluksen käyttöliittymästä, jossa näkyvillä heksadesimaalimuotoista viestintää. Lähetettävät viestit muodostettiin asiakkaan laatiman spesifikaation pohjalta ja näiden mukaan laitteet paritettiin ja yhdistettävä laite siirrettiin dataa lähettävään tilaan. Vuokaavio laitteen alkukonfiguraatioista esitetty kuvassa 6.



KUVA 5 Serial Bluetooth Terminal -sovelluksen käyttöliittymä (Morich, K 2022).



KUVA 6 Bluetooth-laitteen parittamisen vaiheet CAN-viesteillä.

React Native Bluetooth Classic -kirjasto suoriutui hyvin itse Bluetooth-yhteyden kannalta liittyvien toiminnallisuuksien toteuttamisessa. Toinen kehitysprojektin kannalta keskeinen ongelma kuitenkin ilmeni mobiilisovelluksen kehityksen aikana, mistä kerrotaan vielä tarkemmin datan vastaanottamiseen ja käsittelyyn keskittyvässä kappaleessa.

React Nativella kehitettävä mobiilisovellus kykeni hyödyntämään mobiililaitteen omia toiminnallisuuksia, esimerkiksi yksi Bluetooth-kommunikaation kannalta keskeisiä ominaisuuksia oli vaatia käyttäjältä sijaintitietojen hyödyntämistä, joka oli samalla itse projektin kannalta keskeinen ominaisuus. Bluetooth-kirjasto

käytti sijaintitietoja Bluetooth-laitteiden etsintään. Kuva koodista sijaintitietojen pyyntöön on kuvassa 7. Kuvan koodissa näkyy React Nativen Android-käyttöjärjestelmän lupia pyytävä *PermissionsAndroid.request()*-funktio. Funktiossa käytetään parametria *PermissionsAndroid.PERMISSIONS.ACESS_FINE_LOCATION* vaatimaan käyttäjän sijaintitietojen käyttöönottoa. Funktio avaa käyttäjälle mobiilisovelluksen käyttöliittymään ponnahdusikkunan. Mikäli käyttäjä ei salli sijaintitietoja, Bluetooth-laitteita ei voida etsiä.

```
// https://reactnative.dev/docs/permissionsandroid (dangerous permissions)
const requestAccessFineLocationPermission = async () => {
    const granted = await PermissionsAndroid.request(PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION, {
        title: "Fine location access required for discovery",
        message: "In order to perform Bluetooth device discovery, you must enable/allow fine location access.",
        buttonNeutral: "Ask me later",
        buttonNegative: "Cancel",
        buttonPositive: "OK",
    });
    return granted === PermissionsAndroid.RESULTS.GRANTED;
};
```

KUVA 7 Funktio koodissa käyttäjän sijaintitietojen sallimiseen.

Käyttäjän myönnettyä sovellukselle sijaintitietojen käyttöönoton, React Native Bluetooth Classic -kirjaston yksi keskeisimmistä toiminnallisuuksista oli pyytää käyttäjältä lupaa Bluetoothin käyttöön. Bluetooth-kirjaston toiminnallisuuksia käytettiin kutsumalla ensiksi itse kirjastoa alkuliitteellä *RNBluetoothClassic*, jonka jälkeen pisteen erottamana käytettiin kirjastosta löytyviä funktioita. Lista kaikista kirjaston funktioista löytyy tämän kotisivuilta. Kuvassa 8 esitellään funktio Bluetoothin käyttöönoton pyytämisestä. Kuvassa kutsutaan kirjaston *requestBluetoothEnabled()*-funktiota, joka avaa käyttäjälle mobiilisovelluksen käyttöliittymään ponnahdusikkunan pyytäen lupaa Bluetoothin käyttöönottoon.

```
    const requestEnabled = async () => {
        try {
            const enable = await RNBluetoothClassic.requestBluetoothEnabled();
            dispatch(setBluetoothConnection(enable));
        } catch (error) {
            Alert.alert(`Error occurred while enabling bluetooth: ${error.message}`);
        }
    };
```

KUVA 8 Funktio koodissa Bluetoothin käytön sallimiseen.

Kun luvat sijaintitietojen ja Bluetoothin käyttöönottoon oli annettu käyttäjän toimesta, sovellus pystyi käynnistämään Bluetooth-laitteiden etsinnän. Löydetyt laitteet listattiin sovelluksen käyttöliittymään, joista käyttäjä pystyi valitsemaan mihin laitteeseen yhdistää. Kun käyttäjä valitsi laitteen listasta, sovellus aloitti prosessin yhteyden muodostamiselle.

Koodia Bluetooth-yhteyden muodostamisesta ja katkaisemisesta esitetty kuvassa 9. Kuvan *connect()*-funktio vastaa yhteyden muodostamisesta. Funktiossa hyödynnetään *device*-nimistä oliota, johon on tallennettu yhteyteen valitun laitteen tiedot. Jos laitteeseen ei ole vielä entuudestaan muodostettu yhteyttä, sovellus pyrkii muodostamaan uutta yhteyttä tähän kutsumalla *device*-olion sisältämää *connect()*-funktiota. Sovellus näyttää käyttäjälle ponnahdusikkunassa virheen, jos laitteeseen ei voida muodostaa yhteyttä. Jos laitteeseen yhdistäminen onnistuu, sovellus alkaa kuuntelemaan laitetta mahdolliselta vastaanotettavalta datalta. Yhteyden katkaisusta vastaa *disconnect()*-funktio, joka lopettaa laitteelta vastaanotettavan mahdollisen datan kuuntelun. Datan vastaanottamisesta ja käsittelystä seuraavaksi tarkemmin omassa kappaleessaan.

Kuvassa 10 on esillä kuvan 9 *connect()*- ja *disconnect()*-funktioiden lopussa esiintyvät, viestien lukemisen aloittamiseen ja lopettamiseen tarkoitetut, funktiot *startRead()* ja *stopRead().* Bluetooth-viestien lukeminen tapahtuu kuuntelijoita käyttämällä—sovellus kuuntelee yhteyttä jatkuvasti mahdollisten viestien varalta ja viestin vastaanotettuaan käsittelee tämän eteenpäin. Yhteyttä katkaistaessa *stopRead()*-funktio poistaa kuuntelijat käytöstä.

```
const connect = async () => {
    dispatch(setDisconnect(false));
    try {
        let connection = await device.isConnected();
        if (!connection) {
            console.Log(`Attempting connection to ${device.address}`);
            connection = await device.connect({
                CONNECTION_TYPE: "hex",
            });
            console.Log("Connection successful");
        } else {
            console.Log(`Connected to ${device.address}`);
        }
        setConnectionState(true);
        startRead();
    } catch (error) {
        console.Log(`Connection failed: ${error.message}`);
    }
};

const disconnect = async (disconnected?: boolean) => {
    try {
        if (!disconnected) {
            disconnected = await device.disconnect();
        }
        console.Log(`Disconnected from ${device.name}.`);
        setConnectionState(!disconnected);
    } catch (error) {
        console.Log(`Disconnect failed: ${error.message}`);
    }
    stopRead();
};
```

KUVA 9 Koodin funktiot Bluetooth-yhteyden muodostamiselle ja katkaisemiselle.

```
// Adds subscription listener for data received
const startRead = () => {
    console.Log("Starting read from device");
    // readSubscription = device.onDataReceived((data: BluetoothDeviceReadEvent) => onReceivedData(data));
    readSubscription = device.onDataReceived((data: BluetoothDeviceReadEvent) => setDataFromBl(data));
    disconnectSubscription = RNBluetoothClassic.onDeviceDisconnected(() => disconnect(true));
};

const stopRead = () => {
    console.Log("Stopping read from device");
    if (readSubscription) {
        readSubscription.remove();
    }
};
```

KUVA 10 Koodin funktiot viestinnän kuuntelun aloittamiselle ja lopettamiselle.

5.3 Datan vastaanottaminen ja käsittely

React Native Bluetooth Classic -kirjastossa tuli mobiilisovelluskehitysprojektin kannalta hyvin keskeinen ongelma vastaan: myös vastaanotettu data oli aina tekstimuodossa, vaikka tämä pitäisi vastaanottaa puhtaassa heksadesimaalimuodossa. Bluetooth-simulaattori lähetti dataa heksadesimaaleista muodostuvista viestiketjuista, jotka Bluetooth-kirjasto oletuksena käänsi tekstimuotoon. Vaikka Bluetooth-kirjaston käännöksen käänsikin takaisin heksadesimaalimuotoon, osa vastaanotetusta datasta oli kadonnut käännösten aikana. Heksadesimaalimuodosta kääntäessä tekstimuotoon jokaiselle datan pätkälle ei ole vastaavaa symbolia tekstimuodossa, ASCII-järjestelmässä, joten nämä korvattiin tietynlaisilla vakiomerkeillä. Vaikka data käännettiinkin takaisin heksadesimaaleiksi, se ei enää vastannut Bluetooth-simulaattorin lähettämää raakaa dataa. Tämä varmistettiin Serial Bluetooth Terminal -mobiilisovellusta hyödyntämällä, joka vastaanotti ongelmitta simulaattorilta asiakkaan spesifikaation mukaista dataa. Jotta sovellus vastaanottaisi datan puhtaana heksadesimaalina, mobiilisovelluksen backendiin ja Bluetooth-kirjastoon piti toteuttaa muutoksia Java-ohjelmointikielellä Android Studiossa. Kuvassa 11 on esitelty osa Bluetooth-kirjaston datan vastaanottamiseen keskittyvään tiedostoon/luokkaan tehtyä lisäystä datan käsittelyä varten. Kuvan funktio, parametrinaan *bytes*-taulukko, käy vastaanotetun datan läpi muuntaen tavallisessa ASCII-tekstimuodossa olevat tavut heksadesimaaleiksi. Tehdyn muutoksen jälkeen Bluetooth-simulaattorilta vastaanotettu data vastasi opinnäytetyön toimeksiantajan asiakkaan laatimaa spesifikaatiota.

```
private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
public static synchronized String bytesToHex(byte[] bytes) {
    char[] hexChars = new char[bytes.length * 2];
    for (int j = 0; j < bytes.length; j++) {
        int v = bytes[j] & 0xFF;
        hexChars[j * 2] = HEX_ARRAY[v >>> 4];
        hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
    }
    return new String(hexChars);
}
```

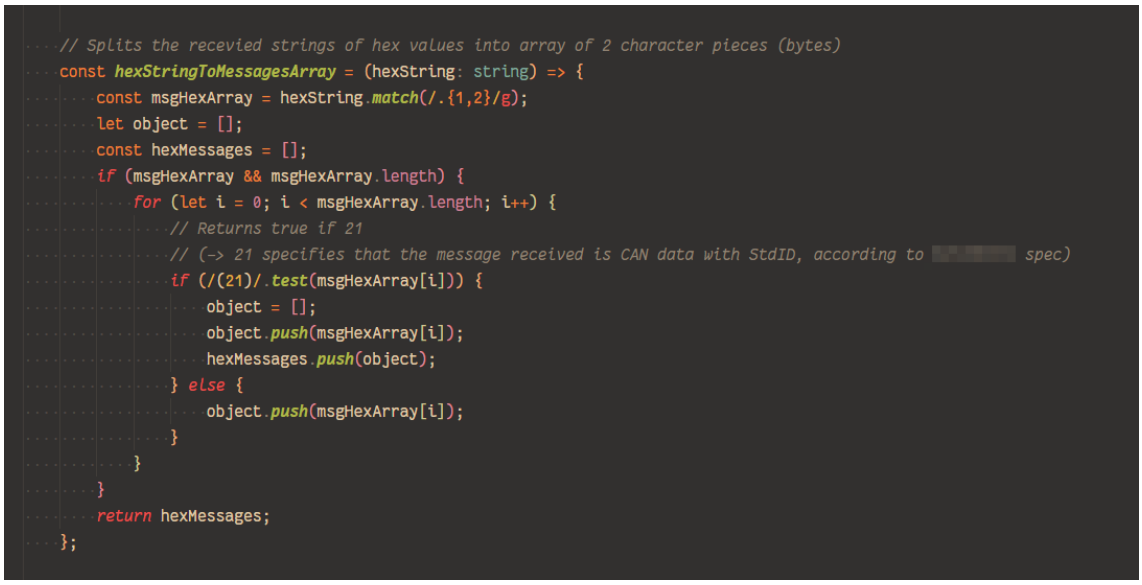
KUVA 11 Java-koodi Android Studiossa vastaanotettujen viestien kääntämiseen heksadesimaaleiksi.

Vastaanotetun, Java-koodin oikeaan muotoon käsittelemän, datan käsittely jatkui React Native -koodissa kuvassa 12 olevalla *onReceivedData()*-funktiolla. Funktio toimii yläfunktiona kolmelle muulle tämän sisällä kutsutulle funktiolle, *hexStringToMessagesArray()*, *crcCheckAndADataParsing()* ja *sendDatatoTicket().*

```
/* Handle the received data:
 * 1. Split the received string of hex values into array of 2 character pieces (bytes)
 * 2. Calculate and perform a CRC32 check on the array of hex values, form a data object
 * 3. Send the data object to IoT-Ticket
 */
const onReceivedData = async (event: BluetoothDeviceReadEvent) => {
    // console.log("Data received from device:", event);
    console.log("Received data...");
    const canMessages = hexStringToMessagesArray(event.data);
    const parsedData = crcCheckAndDataParsing(canMessages);
    sendDataToTicket(parsedData);
};
```

KUVA 12 Vastaanotetun datan käsittelystä vastaava funktio koodissa.

Kuvassa 12 esitetyn funktion alla oleva, kuvassa 13 esitetty, *hexStringToMessagesArray()*-funktio pilkkoi vastaanotetut pitkät heksadesimaali-viestijonot yksittäisiksi kahden luvun tavuiksi taulukkoon. Funktio *crcCheckAndDataParsing()* toteutti viestien parsinnan ja virheentarkistuksen mahdollisten korruptoituneiden viestien varalta. Funktiossa käytiin läpi *hexStringToMessagesArray()*-funktion käsittelemät tavut, tarkistaen mitä arvoja nämä vastasivat projektin asiakkaan spesifikaatiossa. Oikeisiin arvoihin parsittu vastaanotettu data lähetettiin IoT-TICKETtiin *sendDataToTicket()*-funktiolla. Opinnäytetyössä ei käsitellä tarkemmin sovelluksen IoT-toiminnallisuuksia.



```
// Splits the recevied strings of hex values into array of 2 character pieces (bytes)
const hexStringToMessagesArray = (hexString: string) => {
    const msgHexArray = hexString.match(/.{1,2}/g);
    let object = [];
    const hexMessages = [];
    if (msgHexArray && msgHexArray.length) {
        for (let i = 0; i < msgHexArray.length; i++) {
            // Returns true if 21
            // (-> 21 specifies that the message received is CAN data with StdID, according to [redacted] spec)
            if (/(21)/.test(msgHexArray[i])) {
                object = [];
                object.push(msgHexArray[i]);
                hexMessages.push(object);
            } else {
                object.push(msgHexArray[i]);
            }
        }
    }
    return hexMessages;
};
```

KUVA 13 Koodin funktio kääntämään yhtenäinen heksadesimaali-viestiketju omiksi kahden luvun tavuikseen.

6 POHDINTA

Opinnäytetyön tavoitteena oli tutkia langattoman viestintäteknologia Bluetoothin historiaa, versioita ja toiminnallisuuksia. Työn tarkoituksena oli toteuttaa ominaisuus Bluetooth-kommunikaatiolle opinnäytetyön toimeksiantaja Wapice Oy:n mobiilisovelluskehitysprojektin Android-sovellukseen. Mobiilisovelluskehitysprojekti alkoi alkuvuodesta 2021 ja jatkui saman vuoden kesään asti, jolloin toteutettua sovellusta testautettiin.

Bluetooth-kommunikaatio, joka on lopulta useimmille hyvinkin arkipäiväinen aihe, sisältää lopulta pitkän historian—nimensä ja logonsa kautta jo keskiajoille asti. Pelkän historian lisäksi, myös itse teknologian toiminnallisuudet ovat varmasti lähes kaikille tuntemattomat. Opinnäytetyössä perehdyttiin viestintäteknologian historiaan, eri versioihin sekä toiminnallisuuksiin. Tämän vanhempia versioita kattavaa Bluetooth Classicia yleisesti vertailtiin moderniin Bluetooth Low Energy -versioon. Vertailua osittain hyödynnettiin opinnäytetyön teknisen osuuden kehitystyössä. Opinnäytetyössä tutkittiin Bluetooth-teknologiaa selventääkseen lukijalle viestintäteknologian toimintaa, sekä lopuksi annettiin teknisen osuuden myötä esimerkki teknologian hyödyntämisestä.

Mobiilisovelluskehitysprojektiin oli suunniteltu Wi-Fi-teknologian hyödyntämistä, mutta Bluetooth lopulta arvioitiin suotuisammaksi projektin kannalta. Projektiin tehtiin tutkimusta Bluetoothin eri versioista, jonka pohjalta päädyttiin käyttämään Bluetooth Classic -versiota teknologiasta. Sovellusta olisi voitu kehittää Bluetooth Low Energy -versiota käyttämällä, mutta Classic oli tässä tapauksessa kuitenkin parempi. Sovelluksen kehityksessä olisi voitu käyttää muutakin viestintäteknologiaa kuin Bluetoothia, kuten esimerkiksi jo mainittua Wi-Fiä, sillä päätavoitteena olikin vain vastaanottaa mobiililaitteeseen peltokoneelta saatava data.

Kehitysprojektin lopputuote kattoi sovelluksen alustavat vaatimukset, mutta sovellusta olisi edelleen vielä mahdollista jatkokehittää. Esimerkiksi yhdistettävän laitteen alkukonfiguraatiot voitaisiin toteuttaa mobiilisovelluksessa. Tämä vaatisi

mobiilisovellukseen ominaisuuden myös heksadesimaalimuotoisille lähetettäville viesteille pelkkien vastaanotettavien lisäksi. Mobiilisovelluksen kehitystä ei ole uudestaan jatkettu opinnäytetyön kirjoitushetkellä.

LÄHTEET

Android. 2022. Meet Android Studio. Viitattu 1.4.2022. https://developer.android.com/studio/intro

Atlassian. 2022. Bitbucket. Viitattu. 23.4.2022. https://bitbucket.org/

Bluetooth SIG, Inc. 2022a. Bluetooth Technology Overview. Viitattu 19.4.2022. https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

Bluetooth SIG, Inc. 2022b. Brand Assets. Viitattu 29.3.2022. https://www.bluetooth.com/media/

Bluetooth SIG, Inc. 2022c. Origin of the Bluetooth Name. Viitattu 29.3.2022. https://www.bluetooth.com/about-us/bluetooth-origin/

Bluetooth SIG, Inc. 2022d. Topology Options. Viitattu 30.3.2022. https://www.bluetooth.com/learn-about-bluetooth/topology-options/

Davidson, K. n.d. React Native Bluetooth Classic. Viitattu 27.4.2022. https://kenjdavidson.com/react-native-bluetooth-classic/

Fraser, M. J., James, D. A. & Thiel, D. 2007. Innovative techniques for extending the range and node limits in Bluetooth based wireless sensor networks. Proceedings of SPIE - The International Society for Optical Engineering.

Gehrmann, C., Persson, J., Smeets & B. 2004. Bluetooth Security. Yhdysvallat, Norwood: Artech House.

Get Connected Blog. 2021. The Difference Between Classic Bluetooth and Bluetooth Low Energy. Viitattu 21.4.2022. https://blog.nordicsemi.com/getconnected/the-difference-between-classic-bluetooth-and-bluetooth-low-energy

Gillis, A. S. 2021. network topology. TechTarget. Viitattu 30.3.2022. https://www.techtarget.com/searchnetworking/definition/network-topology

Gopikrishnan, K. 2020. The Bluetooth Standard – A simple guide to the protocol for beginners. Technobyte. Verkkosivu. Luettu 10.4.2022. https://technobyte.org/bluetooth-standard-beginners-explained-guide/

Gupta, D. 2021. Bluetooth 5.1, 5.2 And 5.3: Features And Differences Between Versions. TechShali. Viitattu 30.3.2022. https://techshali.com/bluetooth-versions/

Gupta, N. K. 2016. Inside Bluetooth Low Energy. 2. painos. Yhdysvallat, Norwood: Artech House.

Hildenbrand J. & Maring J. 2019. A history of all the major Bluetooth releases and updates. Android Central. Viitattu 30.3.2022. https://www.androidcentral.com/history-major-bluetooth-releases-and-updates

Kardach, J. 2008. Tech History: How Bluetooth got its name. EE Times. Viitattu 16.3.2022. https://www.eetimes.com/tech-history-how-bluetooth-got-its-name/

Kvaser. 2022. The CAN Bus Protocol Tutorial. Viitattu 25.4.2022. https://www.kvaser.com/can-protocol-tutorial/

Meta Platforms, Inc. 2022. React Native. Viitattu 10.4.2022. https://reactnative.dev/

Microsoft. 2022a. TypeScript. Viitattu 10.4.2022. https://www.typescriptlang.org/

Microsoft. 2022b. Visual Studio Code. Viitattu 1.4.2022. https://code.visualstudio.com/

Morich, K. 2022. Serial Bluetooth Terminal. Google Play. Viitattu 30.4.2022. https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&gl=US

Padgette, J., Bahr, J., Batra, M., Holtmann, M., Smithbey, R., Chen, L. & Scarfone, K. 2017. Päivitetty 2022. Guide to Bluetooth Security. National Institute of Standards and Technology. Viitattu 15.4.2022. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2-upd1.pdf

Pothitos, A. 2017. The History of Bluetooth. Mobile Industry Review. Viitattu 8.2.2022. https://www.mobileindustryreview.com/2017/08/the-history-of-bluetooth.html

Sony Electronics Inc. 2022. Bluetooth Wireless Technology Profiles and Descriptions. Viitattu 14.4.2022. https://www.sony.com/electronics/support/articles/00007501

Spinellis, D. 2012. Git. IEEE Software 29 (3), 100–101.

Triggs, R. 2021. A little history of Bluetooth. Android Authority. Viitattu 21.3.2022. https://www.androidauthority.com/history-bluetooth-explained-846345/

Zeadally, S., Siddiqui, F. & Baig, Z. 2019. 25 Years of Bluetooth Technology. Future Internet 11 (9), 194.