OULUN AMMATTIKORKEAKOULU

Joonas Mattila

KALIBROINTISESSION SUUNNITTELU JA TOTEUTUS BEYOND PULSE -PROJEKTIIN

KALIBROINTISESSION SUUNNITTELU JA TOTEUTUS BEYOND PULSE -PROJEKTIIN

Joonas Mattila
Opinnäytetyö
Kevät 2020
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, laite- ja tuotesuunnittelu

Tekijä: Joonas Mattila
Opinnäytetyön nimi: Kalibrointisession suunnittelu ja toteutus Beyond Pulse -projektiin
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: Kevät 2020 Sivumäärä: 26 + 1

Opinnäytetyön tavoitteena oli suunnitella projektin eri osapuolien kanssa yhteistyössä kalibrointisessio-ominaisuus Beyond Pulsen tuotesarjaan sekä lisätä siitä toteutus Beyond Pulse -tuotesarjan sensorilaitteeseen, joka on kehitetty Movesense-sensorialustan päälle. Kalibrointisessio on käyttäjän näkökulmasta harjoituskerta, jonka aikana käyttäjä tekee sarjan ohjattuja liikeharjoitteita, joiden aikana talletetun datan perusteella palvelimilla olevia algoritmeja pyritään säätämään siten, että ne huomioivat yksittäisen käyttäjän juoksutekniikan laskennoissaan. Tällä tavoin kalibrointisession tehneet käyttäjät saavat jatkossa esimerkiksi jalkapalloharjoituksistaan tarkempaa dataa.

Työ koostui suunnitteluosuudesta sekä toteutusosuudesta. Suunnitteluosuudessa suunniteltiin ja sovittiin datansiirtoon sekä sensorinhallintaan liittyvistä yksityiskohdista ja käytänteistä yhteistyössä mobiilikehittäjien kanssa. Sen lisäksi data-analyytikon avustuksella suunniteltiin kalibrointisessiossa käytettävälle datalle oma uusi tietorakenne, jonka mukaisesti dataa tullaan kalibrointisession aikana säilömään sekä session päätteeksi myös lähettämään. Toteutusvaiheessa edellä mainitun suunnitelman pohjalta ohjelmoitiin sensoriratkaisun nykyiseen käyttöjärjestelmään halutut ja sovitut toiminnallisuudet, jotka myös alustavasti testattiin yhteistyössä data-analyytikon kanssa.

Projektin toimeksiantajia ovat Symbio Finland Oy sekä projektin asiakas Beyond Pulse. Materiaalina tähän opinnäytetyöhön käytettiin SIG:n Bluetooth Low Energy -standardia, Movesensen dokumentaatiota sekä aikaisempaa tietotaitoa, jota Movesensen parissa työskennellessä minulle on karttunut.

Suunnitteluvaihe eteni aikataulussa ja kalibrointisession yksityiskohdat saatiin sovittua ja dokumentoitua laadukkaasti. Toteutusvaihe eteni myös alustavan aikataulun mukaisesti ja täytti kaikki sille asetetut tavoitteet. Kokonaisuudessaan opinnäytetyön tavoitteet saavutettiin aikataulussa ja alustavien testien perusteella kalibrointisessio toimii halutulla tavalla.

Asiasanat: Movesense, sulautetut järjestelmät, Bluetooth, tiedonlouhinta

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Device and Product Development

Author: Joonas Mattila
Title of thesis: Designing and Implementing a Calibration Session Feature to Beyond Pulse Project
Supervisor: Eino Niemi
Term and year when the thesis was submitted: Spring 2020 Number of pages: 26 + 1

The goal for this bachelor's thesis was to design a calibration session feature to Beyond Pulse's product in co-operation with the mobile developers and then develop an implementation of it into the existing Beyond Pulse's sensor solution. From user's perspective a calibration session is a session where the user does a predefined movement routine that produces data that is used to adapt the backend's data processing algorithms to individual's running technique. This way users receive more accurate data from their normal training sessions since the algorithms take their running technique into consideration when analyzing the data.

The bachelor's thesis consisted of planning section and development section. In the planning section the calibration session's features, requirements and data transmission related details where planned and documented in co-operation with the mobile developers and a new data model for the calibration session was designed in co-operation with a data analysist who is also working in the project. In development phase the calibration session was programmed and implemented onto the latest firmware of the Beyond Pulse's Movesense based sensor solution.

SIG's Bluetooth Low Energy -standard, Movesense's documentation and previous knowledge and know-how was used as material for this project.

The planning section was done in time and documented properly in co-operation with the project's mobile developers. The development phase was also done in time and the end result met all the requirements that were given to it. This bachelor's thesis project was a success based on the preliminary testing that was done with it.

Keywords: Bluetooth, Movesense, Embedded development, Data collecting

ALKULAUSE

Haluan kiittää kaikkia kollegoitani sekä erityisesti esimiestäni Matti Lehtistä kaikesta tuesta ja avusta, jota olen saanut heiltä työskennellessäni Symbiolla Beyond Pulse -projektin sekä tämän opinnäytetyön parissa. Lämmin kiitos myös Oulun ammattikorkeakoulun henkilökunnalle ja opettajille, jotka ovat opettaneet ja ohjanneet minua läpi opintojeni ja auttaneet minua kartuttamaan hyvän pohjan, jonka myötä minun oli helppo lähteä työskentelemään oikeisiin projekteihin ja aloittamaan uraani insinöörinä.

I would like to extend my gratitude towards the great people of Beyond Pulse for allowing me to write this bachelor's thesis about their project. I am very honored and grateful that I have been given the opportunity to be a part of this magnificent project and I want to thank Beyond Pulse and Symbio Finland for that. I would also like to thank everyone who I have been in contact with throughout this project.

Oulussa 21.4.2020

Joonas Mattila

SISÄLLYS

TERMISTÖ

BLE: Bluetooth Low Energy. Langaton radioteknologia, joka on yleinen muun muassa erilaisissa pienen virrankulutuksen IoT-laitteissa joustavuutensa ja pienen virrankulutuksensa vuoksi. (1.)

BLE SIG: Bluetooth Special Interest Group. Järjestö, joka ylläpitää ja valvoo Bluetoothin standardeja, vastaa lisensoinneista sekä valvoo Bluetooth-laitevalmistajia. (2.)

EEPROM-muisti: Electronically erasable programmable read-only memory. Muistityyppi, joka ei poista sisältämäänsä dataa virrattomana eli se on niin sanottua non volotile - muistia. (3.)

IoT: Internet of things, esineiden internet. Älykkäitä sulautettuja laitteita, jotka kykenevät tarkkailemaan ympäristöään ja toimimaan sen perusteella älykkäästi. (4.)

Metriikka: Arvo, jonka avulla voidaan seurata ja arvioida jonkin asian suorituskykyä ja kehitystä. Juoksemisessa näitä ovat esimerkiksi askelmäärä, askeltiheys ja juoksunopeus. (5.)

SOC: System on a chip. Piirilevy, jossa on integroituna kaikki tietokoneen tarvitsemat ja sisältämät komponentit, kuten prosessori ja muistit. Esimerkiksi Raspberry Pi ja Movesense-sensori ovat tällaisia. (6.)

Synkronointipaketti: Datapaketti, jonka sisältämällä datalla on mahdollista saada kaksi laitetta toimimaan yhdessä jaetun datan perusteella. Esimerkiksi lämpötilasensorilaitteen jakaessa synkronointidatassaan mittaamaansa lämpötilaa mobiililaitteelle kykenee mobiililaite näyttämään käyttäjälle sensorin mittaaman lämpötilan tämän myötä. (7.)

Tavujärjestys: kuvastaa tietokoneissa järjestystä, jossa tietokone käsittelee yli yhden tavun mittaisia kokonaisuuksia. Esimerkkitavujärjestyksiä ovat Big endian- ja Little endian -järjestys. (8.)

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on suunnitella ja lisätä Beyond Pulse -tuotteen sensorivyössä olevaan sensoriin kalibrointisessio-ominaisuus. Kalibrointisession tarkoituksena on tarjota tuotesarjaan kuuluvalle taustaohjelmistolle enemmän ja monipuolisemmin dataa, jonka perusteella sensori sekä taustaohjelmiston analytiikka saadaan räätälöityä yksilön juoksutyylin ja -tekniikan mukaiseksi. Tämä lisää tarkkuutta kalibrointisession tehneiden käyttäjien tavallisten sessioiden analysointiin ja tulkintaan. Sessiolla tässä kontekstissa tarkoitetaan urheilusuoritusta, jonka aikana sensori mittaa ja varastoi suorituksesta dataa, jonka pohjalta käyttäjä saa session päätteeksi tietoa suorituksestaan. Tavallinen sessio voi olla esimerkiksi itsenäinen juoksulenkki ihan yksikseen tai jalkapalloharjoitus joukkueen kanssa. Kalibrointisessiossa sen sijaan tehdään vain useita ennalta määriteltyjä harjoitteita ohjatusti, joiden aikana talletetun datan perusteella tuotesarjaan kuuluvan verkkokäyttöliittymän taustaohjelmistoissa toimivat datananalysointialgoritmit oppivat tuntemaan ja jatkossa myös huomioimaan eri käyttäjien juoksutekniikat analyyseissään. Tällä tavoin ne kykenevät jatkossa tarjoamaan käyttäjäkohtaisesti tarkempaa dataa käyttäjilleen.

Beyond Pulse -tuotesarja sisältää mobiilisovelluksen, verkkokäyttöliittymän sekä rintavyöhön kiinnitettävän sensorin, joiden yhdistelmää jalkapallojoukkueiden valmentajat sekä pelaajat voivat hyödyntää parantaakseen harjoitustensa laatua sekä saadakseen laadukasta dataa harjoitusten onnistuvuudesta ja rasittavuudesta. Tämän datan avulla valmentajien on helpompi seurata joukkueen kehitystä sekä harjoitustensa toimivuutta ja tätä kautta saada harjoittelustaan enemmän irti. (.9)

Kalibrointisession lisääminen sensoriratkaisuun tulee vaatimaan koodillisia muutoksia sensoriratkaisun ohjelmistoon sekä askeleentunnistusalgoritmi- että Bluetooth Low energy -kommunikaatioväylään liittyvän koodin osalta. Mobiilikehityksestä vastaa toinen yritys, joten BLE-muutoksia suunniteltaessa on oleellista keskustella muutoksista mobiilikehittäjien kanssa parhaan mahdollisen ratkaisun löytämiseksi ja sovitut asiat tulee dokumentoida laadukkaasti ja selkeästi.

Tämänhetkisen sensorin ohjelmistoversio tarjoaa hyvän pohjan kalibrointisession lisäämiseen, sillä sensorissa on jo BLE-kommunikaation hallintakoodia sekä algoritminhallinnointikoodia tavallisen session hallinnointiin. Tätä koodia täydentämällä ja muokkaamalla kalibrointisessioon tarvittavien ominaisuuksien ohjelmointi on helpompaa, sillä kaikkea ei tarvitse tehdä aivan tyhjästä.

2 TEHTÄVÄN KUVAUS JA SIIHEN LIITTYVÄ TEORIA

2.1 Beyond Pulse tuotteena

Beyond Pulse -tuotesarjan tavoitteena on tarjota tuotteiden käyttäjille laadukasta dataa, joka mahdollistaa paremman ja laadukkaamman harjoittelun sekä tukee yksilökehitystä harjoittelun ohella. Tuotesarja koostuu muunnellusta Movesense-sensorista, jota pelaaja pitää harjoitellessaan päällä sykevyön tavoin. Movesense-sensori ajaa sisällään muunneltua ohjelmistoa, joka on kehitetty juuri Beyond Pulsen käyttötarkoituksen mukaiseksi. Sensoriratkaisun tueksi pakettiin kuuluu myös mobiilisovellus, jolla sensoria hallitaan, sekä verkkokäyttöliittymä, joka mahdollistaa harjoittelusessioista kerätyn datan tarkastelemisen ja visualisoinnin. (9.)

Beyond Pulse -mobiilisovelluksesta on kaksi eri versiota: valmentajien versio ja pelaajien versio. Näiden kahden ero on se, että valmentajien sovelluksella on mahdollista hallinnoida koko joukkueen sensoreita, kun taas pelaajasovellus antaa mahdollisuuden hallinnoida vain pelaajan omaa sensoria. Pelaajien versio sallii myös yksilöharjoittelun datan jakamisen valmentajille, mikäli käyttäjä niin haluaa tehdä. Beyond Pulsen tuotteiden pääkäyttäjäkunta on jalkapallojoukkueet eri puolilla maailmaa. (9.)

2.2 Movesense-kehitysympäristö ja sen tarjoamat mahdollisuudet

Movesense on Suunnon kehittämä avoin kehitysalusta. Se on rakennettu Nordic Semiconductorsin nRF52832 -järjestelmäpiirin ympärille, jonka keskiössä on Armin Cortex M4F -prosessori.(10.) Movesense-kehitysalusta mahdollistaa nopeamman sekä tehokkaamman sensoriratkaisun kehittämissyklin, sillä anturointilaitteistoa, kuten esimerkiksi kiihtyvyysanturia tai sykkeenmittausanturia ei itse tarvitse valita ja implementoida sensoriin, vaan anturit ovat jo sisäänrakennettuina tarjolla ja niiden hallinnointiin on myös tehty helppokäyttöisiä ohjelmointirajapintoja, joita hyödyntämällä ohjelmistokehittäminen on nopeaa ja yksinkertaisempaa.(11.) Movesensen ohjelmoimista on myös helpotettu tarjoamalla ohjelmistorajapintoja moniin muihin sensorin ominaisuuksiin ja komponentteihin, kuten esimerkiksi Bluetoothin hallintaan, sensorin sisäisten asetusten hallintaan sekä Eeprom-muistin hallinnointiin. Movesensen pohjakoodissa oleva Whiteboard-niminen järjestelmä-

palveluidentarjoaja hoitaa myös matalamman tason prosessit ja niiden hallinnoinnin, kuten esimerkiksi muistin allokoinnin, prosessien vuorottamisen sekä keskeytyspalveluiden hallinnoinnin, joten kehittäjänä voi keskittyä ylemmän tason ohjelmointiin.

Movesense-sensorissa on laadukas kiihtyvyysanturi, gyroskooppi, magnetometri, sykkeenmittausmahdollisuus sekä 1-Wire-väylä, joka mahdollistaa erilaisten liitännäisten kytkemisen sensoriin sykkeenmittauksen sijaan. Suunto on tunnettu korkeatasoisten urheilukellojen valmistaja sekä suunnittelija, joten yrityksen käyttämät anturit ovat myös todella laadukkaita. Sensori kommunikoi ulkomaailman ja sovellusten kanssa käyttäen Bluetooth Low Energyä, jonka kanssa on mahdollista käyttää joko omaa GATT-palvelua tai Movesensen sisäisen Whiteboard-järjestelmän tarjoamaa palvelua. (12.)

Movesense käyttää sisäiseen sekä vaihtoehtoisesti myös ulkoiseen kommunikointiin Whiteboard-nimistä järjestelmää, joka tarjoaa REST-tyylisiä ohjelmointirajapintoja sekä asynkronisia callback-funktioita, joiden avulla ohjelmisto kykenee lähettämään sekä vastaanottamaan dataa ohjelmointirajapintojen kanssa sisäisesti sekä ulkoisesti. Esimerkiksi jos kehittäjä haluaisi saada ohjelmistossaan kiihtyvyysanturin tarjoamaa dataa, hän voi ohjelmallisesti tilata sitä halutulla taajuudella kiihtyvyysanturi-rajapinnalta ja vastaanottaa tätä tilattua raakadataa callback-funktiossa. Samalla tavoin myös lähettämällä PUT-komennon mukana dataa rajapinnalle voi esimerkiksi kiihtyvyysanturin asetuksia muuttaa ja rajapinnan vastausta voi tarkastella myös omassa callback-funktiossaan, jolloin voi helposti tarkistaa, onko komento mennyt perille ja se on käsitelty oikein. (12.)

2.3 Movesensen kehitys- ja käännöstyökalut

Movesense-alustalle ohjelmistoa kehittäessä on tärkeää tietää, millä kehitystyökaluilla ja käännöstyökaluilla ohjelmiston kehittämistä suositellaan tekemään. Movesensen kehitysympäristöksi suositellaan Microsoftin Visual Studio IDE:tä, sillä Movesense on kehittänyt sille niin kutsutun simulaattori-koontiversion, jota käyttämällä on mahdollista simuloida Movesense-sensorin toimintaa, antureita sekä kaikkia muita ominaisuuksia, paitsi Bluetoothia, jolle simulaattorissa ei toistaiseksi ole tukea.

Simulaattorin avulla Movesense-sensorin ohjelmistonkehittäminen nopeutuu huomattavasti, sillä sen avulla ohjelmiston testaaminen tapahtuu nopeasti ja Visual Studion sisäänrakennetut työkalut

helpottavat myös koodin syntaksin kanssa. Simulaattorille on myös mahdollista syöttää esimerkiksi raakaa kiihtyvyysdataa .csv -tiedostona, jota simulaattori sitten simuloinnin aikana syöttää kiihtyvyysanturin ulostulona. Tätä ominaisuutta hyödyntämällä on mahdollista simuloida samaa käyttötilannetta samalla kiihtyvyysdatalla lukuisia kertoja peräkkäin ohjelmistoa kehittäessä ja kehittäjällä on myös täysi tieto käytettävästä kiihtyvyysdatasta, joten sitä on mahdollista analysoida myös muilla työkaluilla ja tavoilla, mikä myös helpottaa ohjelmiston kehittämistä.

Kun ohjelmisto on saatu siihen pisteeseen, että sitä voisi myös kokeilla itse oikealla Movesense-sensorilla, täytyy ohjelmisto ensin kääntää sensorille. Ohjelmiston kääntäminen vaatii GNU-työkalut Armin sulautetuille laitteille, Ninja-työkalun, Cmake-työkalun, Python 2.7 -kääntäjän sekä muutamia Nordic Semiconductorin tarjoamia työkaluja, joita käännösprosessi hyödyntää. Tarkka lista työkaluista ja niiden versioista sekä ohjeet ohjelmiston kääntämiseen löytyvät Movesensen sivuilta. (13.)

Edellä mainittujen työkalujen avulla Movesense-sensorin ohjelmisto on mahdollista kääntää joko niin kutsutuksi Over the air- paketiksi tai jos saatavilla on Movesense-sensorille tarkoitettu fyysinen ohjelmointityökalu (JIG), on käyttöjärjestelmäohjelmisto mahdollista kääntää myös binäärikäännökseksi ja ohjelmoida sensoriin käyttäen tätä työkalua. OTA-paketti on mahdollista lähettää sensorille Bluetoothin yli käyttämällä Movesensen tarjoamaa Showcase-app -mobiilisovellusta joko Android- tai iOS -laitteella. OTA-paketti tulee siirtää puhelimeen ja Showcase-appia käyttämällä se voidaan lähettää ja asentaa sensorille suhteellisen vaivattomasti. Toisin kuin JIG:llä ohjelmointi, tämä OTA-prosessi kestää useamman minuutin, ja siksi JIG on parempi työkalu käyttöjärjestelmän ohjelmointiin sensorille. (13.)

Movesensen Showcase-apin avulla on mahdollista hallinnoida, testata tai muokata asetuksia Movesense-sensoreissa. Sillä on myös mahdollista tallentaa sensorilta esimerkiksi raakaa kiihtyvyysdataa, jota voi sitten hyödyntää simulaattorissa taikka datan analysoinnissa. Movesensen suosittelema menetelmä ohjelmistokehittämiseen onkin tallettaa tarvittavaa sensoridataa tarkasteltavasta ilmiöstä tai liikkeestä, analysoida sitä ja analyysin lopputulosten ja pääpiirteiden perusteella siirtyä kehittämään ohjelmistoa sensorille. Esimerkiksi jos sensorilla haluttaisiin tarkkailla ja analysoida hiihtäjän sykettä ja hiihtonopeutta, olisi ensimmäiseksi parasta pukea sensori hiihtäjän ylle ja tallettaa vähän aikaa kiihtyvyys- ja sykedataa hiihtäjän hiihdosta. Tämän jälkeen talletettua dataa voisi analysoida ja analyysin perusteella voitaisiin kehittää sensorin ohjelmistoon algoritmi haluttujen asioiden tarkkailuun. On suositeltavaa suorittaa mahdollisimman paljon laskentaa Movesense-

sensorissa, sillä siinä on hyvin laskutehoa ja laskemalla sensorilla haluttuja asioita saadaan lähetettävän datan määrä pienemmäksi ja sitä kautta paristonkestoa sensorissa paremmaksi.

Toinen hyvä työkalu Android-puhelimille on NRF-connect-mobiilisovellus, joka on Nordic Semiconductorsin kehittämä puhelinsovellus. Sen avulla voi tutkia Movesense-sensorin Bluetooth-puolen ominaisuuksia kuten mainostuspakettia sekä erityisesti GATT-palveluiden toimintaa sekä ominaisuuksia tarkemmin kuin monilla muilla sovelluksilla.

2.4 Kalibrointisession tavoitteet sensorikehityksessä

Kalibrointisession tärkeimmät ominaisuudet sekä muutokset liittyvät sensorin BLE-kommunikointiin sekä algoritmilta saadun datan käsittelyyn sensorin sisällä. Perimmäinen ajatus on, että mobiililaitteella kyetään valitsemaan tavallisen session ja kalibrointisession väliltä, ja jotta tämä olisi mahdollista, sensorin täytyy kyetä erottamaan tavallisen session ja kalibrointisession aloituskomennot toisistaan. Beyond Pulsen sensori käyttää BLE GATT -palvelua komentojen vastaanottamiseen ja datan synkronoimiseen mobiililaitteelta, joten kalibrointisession hallinnointi tulee saada sulautettua GATT-palveluun mukaan rikkomatta aikaisempia toiminnallisuuksia. Paras mahdollinen ratkaisu kalibrointisession aloittamisen ja tavallisen session aloittamisen erottamiseen toisistaan on lisätä kalibrointisession aloitukselle ihan oma komentonsa, jonka sensori sitten osaa tulkita oikein komennon vastaanottaessaan.

Kalibrointisessio ja tavallinen sessio eroavat toisistaan siten, että kalibrointisession aikana sensori tallentaa useampaa erilaista metriikkaa algoritmikoodilta tavalliseen sessioon verrattuna ja kalibrointisession datan talletustaajuus on myös korkeampi kuin tavallisessa sessiossa. Tämän takia kalibrointisession datoille päätettiin kehittää oma datapaketti-malli, johon kalibrointimetriikan datat saadaan pakattua tiiviisti, mutta riittävän tarkasti. Datamallia suunnitellessa jouduttiin huomioimaan sisään tulevien metriikoiden halutut tarkkuudet, sekä bittimäärä, johon datamalli haluttiin saada mahtumaan. Parempi tarkkuus esimerkiksi desimaalien osalta kasvattaa huomattavasti datapakettien kokoa, ja ne taas vievät enemmän tilaa rajalliselta EEPROM-muistilta. Isot datapaketit myös vaativat enemmän lähetysaikaa Bluetoothilta, mikä vaikuttaa myös suoraan sensorin pariston elämänkaareen, sillä Bluetooth on yksi suurimmista virrankuluttajista Movesense-sensorissa. Datamallia kehittäessä siis oli tärkeintä löytää kompromissi tarkkuuden ja paketin koon väliltä.

2.4.1 Bluetooth Low Energy kommunikaatiorajapinta

Bluetooth Low Energy on pienivirtainen ja monipuolinen radioteknologia, jota monet IoT-laitteet hyödyntävät kommunikoimiseensa. Movesense-sensorit käyttävät myös BLE:tä tiedonsiirtoonsa ja kommunikointiinsa juurikin näistä syistä. (1.) uusjava

BLE:n monikerroksinen protokollapino sisältää useita protokollia aina fyysisestä kerroksesta istuntokerrokselle asti. Koska Movesense-sensori on rakennettu Nordic Semiconductorsin nRF52832-järjestelmäpiirin ympärille, tulevat kaikki osat protokollapinosta valmiina kuvassa 1 näkyvän Soft devicen mukana sensoreihin. Soft device on Nordic Semiconductorsin kehittämä kokoelma koodia, joka sijoitetaan tiettyyn kohtaan laitteen järjestelmämuistia ja se sisältää kaiken BLE-protokollapinon koodin. Suunnon tarjoama Movesensen ohjelmointiympäristön koodi sekä ohjelmointirajapinnat taas hyödyntävät ja rakentavat ominaisuutensa tämän Soft devicen ympärille ja tarjoavat sitä kautta Movesense-kehittäjille mahdollisuuden vaikuttaa laitteen tiettyihin BLE-ominaisuuksiin. (14.)

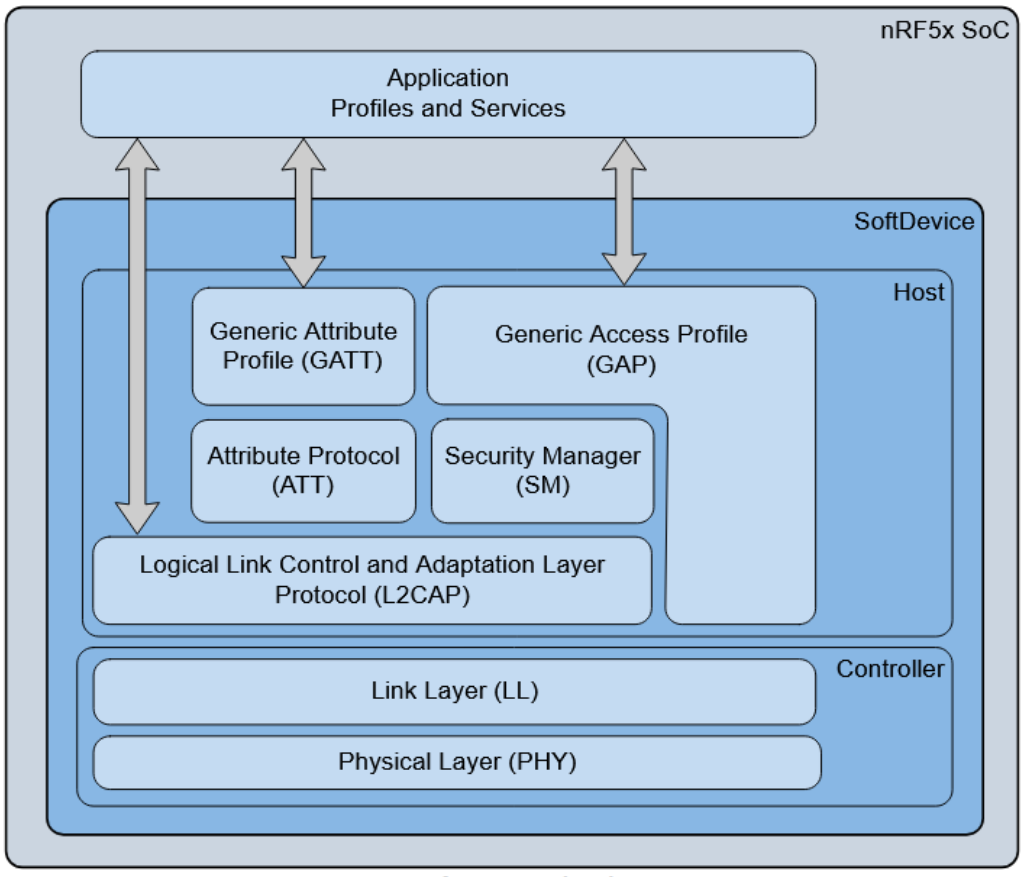
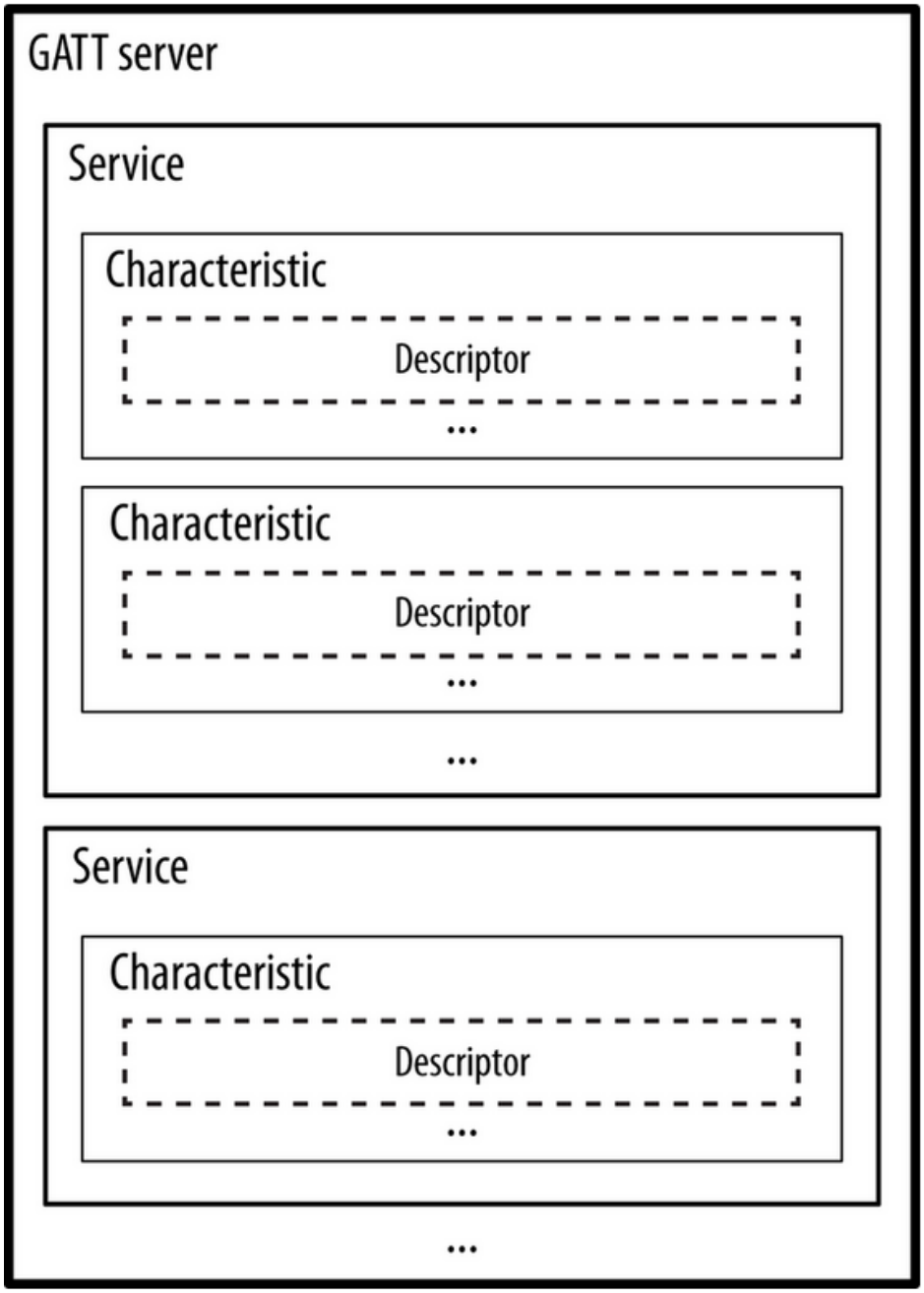


Figure 1. SoftDevice stack architecture

KUVA 1. Soft devicen protokollapinon arkkitehtuuri. (14.)

Movesense-kehittäjälle tästä BLE:n protokollapinosta oleellisimpia protokollia ovat Generic Access Profile eli GAP sekä Generic Attribute Profile eli GATT. GAP vastaa BLE-laitteen mainostamisesta ja löytyvyydestä, yhteyden luomisesta sekä siihen liittyvistä turvallisuusominaisuuksista ja laitteen roolista yhteydessä. Movesense tarjoaa muun muassa mahdollisuuden muokata BLE-mainostuspakettia sekä GAP-profiilissa olevaa laitteen nimeä sekä muita asetuksien vaihteluja GAP-profiilissa BLE-ohjelmointirajapinnan välityksellä. (15.)

GATT taas määrittelee ja vastaa datan siirtämisestä BLE-laitteelta toiselle GATT-palveluiden avulla. GATT-palveluita voi olla useita ja yksittäinen GATT-palvelu koostuu yhdestä tai useammasta characteristicista. Characteristic on BLE:ssä eräänlainen laitteiden välinen datansiirtopiste, jonka kautta GATT-palvelu kykenee lähettämään ja vastaanottamaan dataa eri BLE-laitteiden välillä. Havainnollistavana esimerkkinä GATT-palvelun voi nähdä postilaitoksena ja characteristicin pakettiautomaattina. Characteristic jakautuu declaration-osioon sekä value-osioon. Declaration-osio sisältää muun muassa characteristicin yksilöintiin käytettävän UUID:n eli Universally Unique Identifierin sekä characteristicin ominaisuudet. Characteristicin ominaisuudet määrittelevät voiko characteristicille esimerkiksi kirjoittaa arvoja vai salliiko se vain datan lukemisen. Characteristicin value-osio taas sisältää itse characteristicin datan, jota voidaan declaration-osion määrittelemien ominaisuuksien mukaisesti käsitellä. Characteristicin alla voi olla myös descriptoreita, joiden tarkoitus on tarjota enemmän metadataa characteristicista, sekä mahdollistaa notifikaatioiden ja indikaatioiden aktivoiminen ja lopettaminen clientin eli GATT-palvelun asiakkaan toimesta. Notifikaatiolla tarkoitetaan sitä, että palvelu lähettää ilmoituksen clientille, kun uusi arvo on asetettu characteristicille ja indicaatio on muuten vastaavanlainen, mutta siinä clientin tulee lähettää vahvistus joka kerta, kun se on vastaanottanut dataa GATT-palvelulta. Kuvassa 2 näkyy rakennekuva GATT-palvelimesta, jonka alta löytyy kaksi erilaista GATT-palvelua, joista toisessa on vain yksi characteristic ja toisessa kaksi characteristicia. (16.)



KUVA 2. Esimerkki GATT-palveluiden rakenteesta (16.)

GATT-palveluita, niiden alta löytyviä characteristiceja sekä mahdollisia descriptoreita erotellaan toisistaan UUID:n eli Universally Unique Identifierien avulla. GATT-palvelun tai sen alta löytyvän characteristicin UUID voi olla joko 16-bittinen taikka 128-bittinen. Bluetooth-standardissa olevat standardisoidut GATT-palvelut, characteristicit sekä descriptorit käyttävät standardista löytyviä 16-bittisiä UUID:tä ja Bluetooth SIG myös myy 16-bittisiä UUID:tä niitä haluaville 2500:n euron hintaan.

128-bittiset UUID:t taas eivät ole varattuja, joten 128-bittisen UUID:n GATT-palvelulleen taikka characteristicilleen voi generoida itse. Bluetooth SIG on listannut varatut ja standardisoidut UUID:t omilla nettisivuillaan, joiden kautta niitä voi tarkastella. (17.)

16-bittisten UUID:den lisäksi Bluetooth SIG on määritellyt standardissaan myös sarjan yleisimpiä GATT-palveluita, joiden UUID:t, characteristicit sekä toimintaperiaatteet organisaatio on määritellyt ja standardoinut. Hyvä esimerkki tällaisesta standardisoidusta GATT-palvelusta on sykkeenmittaus-palvelu, johon Movesense:n kehitysalusta tarjoaa hyvän esimerkkikoodin sekä optionaalisen moduulin, joiden avulla palvelun saa helposti toimimaan sensorissa. Standardoitujen palveluiden etu piilee siinä, että monet sovellukset tukevat standardisoituja GATT-palveluita, sillä niiden toiminta on aina standardin mukaista.

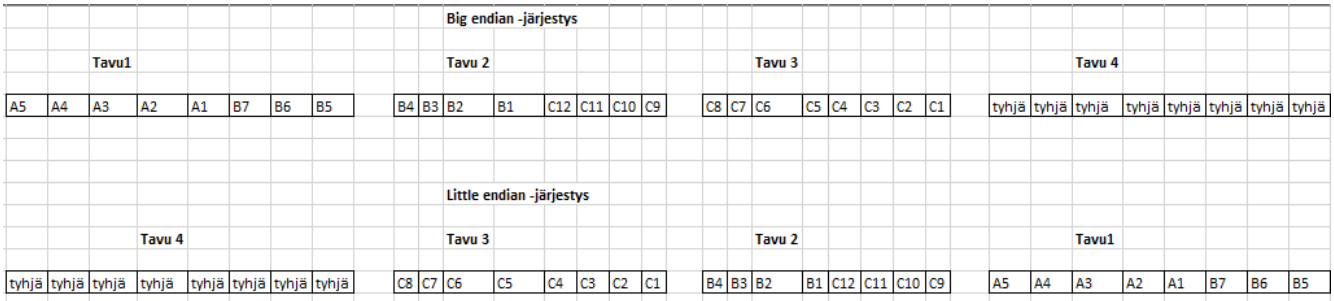
Movesensen kehitysympäristössä itsemääriteltyjen GATT-palveluiden luominen ja kehittäminen tapahtuu Whiteboardin BLE-ohjelmointirajapinnan kautta. Ohjelmointirajapinnalle määritellään GATT-palvelun UUID, characteristicit, sekä niiden alla olevat descriptorit, jonka jälkeen rajapinta hoitaa lopun työn. Rajapinnan kautta saadaan myös kaikki tarvittavat sisäiset osoitteet, joiden avulla characteristiceille voidaan kirjoittaa sekä niiltä voidaan lukea arvoja erinäisten callbackien avulla sensorin sisällä.

2.4.2 Synkronointipaketti ja sen muodostaminen

Synkronointipaketti Beyond Pulsen projektissa on kokoelma datapaketteja, jotka sensori lähettää mobiililaitteelle BLE:n välityksellä, ja se sisältää viimeisimmän session aikana askeleentunnistusalgoritmilta vastaanotetun metriikan perusteella luodun prosessoidun datan. Datapaketteja luodaan tasaiseen tahtiin session aikana ja valmiit datapaketit pakataan EEPROM-muistille odottamaan session loppumista, jonka jälkeen ne voidaan lähettää mobiililaitteelle jatkokäsittelyyn ja sitä kautta myös pilvessä oleville jatkokäsittelyalgoritmeille sekä tietokantaan. Jotta session pituus voidaan maksimoida ilman, että sensorin EEPROM-muisti loppuisi kesken, ja jotta synkronointipaketin siirto BLE:n yli tapahtuisi mahdollisimman tehokkaasti ja paristoa säästäen, täytyy synkronointipaketin data pakata tiiviisti. On myös tärkeää, ettei dataa tiivistetä liikaa, jotta datassa olevien metriikoiden tarkkuus ei kärsi.

Lähetettäessä dataa BLE:n yli tulee tiedostaa ja huomioida se, että data sekä monitavuiset arvot lähetetään Little endian -järjestyksessä eli pienin tavu lähetetään ensin ja isoin tavu viimeisenä. Tämä tarkoittaa sitä, että jos data on sensorilla tavallisesti Big endian -järjestyksessä (isoin tavu ensimmäisenä), tulee data muuttaa Little endian -järjestykseen, jotta se pyörähtää takaisin Big endian -järjestykseen vastaanottopäässä ja helpottaa tällä tavoin datan purkamista synkronointipaketista.

Koska projektin datapaketin sisältö ja bittijärjestys ei ole julkista tietoa, lähestyn synkronointipaketin luomista esimerkin avulla. Jos arvolle A olisi varattu 5 bittiä, arvolle B 7 bittiä ja arvolle C 12 bittiä ja ne haluttaisiin yhdistää kuvassa 3 näkyvään neljästä uint8_t-arvosta koostuvan uint32-arvon sisälle sekä toimittaa vastaanottajalle kuvan ylärivin mukaisessa Big endian -järjestyksessä, täytyisi arvot järjestellä kuvan alarivissä näkyvään Little endian -järjestykseen. Data pyörähtää lähetyksen myötä Big endian -järjestykseen ja vastaanottajan on sen jälkeen mahdollista parsia helposti A:n, B:n ja C:n arvot itselleen Big endian -järjestyksessä olevasta datasta.



KUVA 3. Esimerkkidataa Big endian- ja Little endian -järjestyksessä

Bittitasolla arvojen siirtelyyn ja sijoitteluun tärkeimpiä työkaluja ovat bittioperaattorit, jotka noudattavat Boolen Algebran logiikkaa. Esimerkkidatan tapauksessa hyödyllisimpiä bittioperaattoreita ovat bittisiirtäminen(bitshifting), binääritason AND-operaatiolla bittien parsiminen eli ns. bittimaskittaminen sekä bittitason OR -operaatiot. Näitä työkaluja hyödyntämällä datan tavujärjestystä voidaan muuttaa ja datan bittejä voidaan siirrellä ja jakaa tavujen sisällä.

Jos A:n arvolle on esimerkkikuvan 3 mukaisesti varattu tilaa viiden bitin verran ja A:n arvo halutaan sijoittaa Little endian -järjestyksessä olevaan tavu 1:n, tarvitsee A:n arvo ensin rajata aina viiden bitin kokoiseksi ja sen jälkeen se tulee siirtää bittitasolla oikealle paikalle tavu 1:sen sisään. Arvon A rajaaminen viiteen bittiin tarkoittaa sitä, että sen maksimiarvo voi olla 31 (2 potenssiin 5) riippumatta muuttujan maksimiarvosta, johon A:n arvo on sidottu. Tämä tarkoittaa, että rajaamalla A:n

arvon viiteen bittiin varmistamme, että jos A:n alkuperäinen arvo olisi sidottu esimerkiksi kahdeksan bitin kokoiseen uint8_t muuttujaan, rajoittuu sen maksimiarvo silti 31:een, vaikka uint8_t maksimiarvo voi ollakin 255. Arvon rajaaminen tapahtuu bittimaskaamalla eli tekemällä AND-bittioperaation toista bittiarvoa vasten, jonka tuloksena ainoastaan viisi bittiä voivat päästä läpi operaatiosta. Kuvassa 4 näkyvän Boolean logiikan AND-operaation totuustaulun mukaisesti AND-operaation lopputulos voi olla tosi eli 1 vain jos kumpikin operaation osapari on 1. Joten jos asetamme A:n arvon AND -operaatioon arvoa 31 (0x1F heksadesimaalisena lukuna) vasten, jossa binäärinä ensimmäiset viisi bittiä ovat ykkösiä, voimme rajata a:n arvon viiden bitin kokoiseksi. (18.)

Logical conjunction

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

KUVA 4. AND-operaation totuustaulu (19.)

Asettamalla A:n arvoksi 255:n, joka on maksimiarvo uint8_t muuttujalle ja reilusti yli haluamamme maksimiarvon 31, voimme todistaa kuvassa 5, että A:n arvo rajoittuu 31:ksi bittimaskauksen seurauksena.

A:n arvo	1	1	1	1	1	1	1	1
31(0x1F)	0	0	0	1	1	1	1	1
lopputulos	0	0	0	1	1	1	1	1

KUVA 5. Bittimaskaus, joka rajoittaa A:n arvon korkeintaan viiden bitin suuruiseksi

Kuten lopputuloksesta näkyy, A:n arvo on nyt rajattu viiteen bittiin eli maksimissaan 31:n ja tämä rajaus pitää huolen, että jos A sattuisi virheellisesti saamaan isomman arvon kuin 31, ei ylimääräiset bitit pääsisi läpi bittimaskauksesta ja sitä kautta ylimenevät bitit eivät myöskään pääse sotkemaan muille arvoille varattuja bittialueita.

Nyt kun A:n arvo on bittimaskattu viiteen bittiin, voimme huomata, että nämä viisi bittiä pitäisi siirtää vielä tavun 1 toiseen päähän sillä, jos asettaisimme bittimaskatun A:n arvon vain suoraan tavun 1 arvoksi, näyttäisi se kuvan 6 mukaiselta bittitasolla verrattuna sille tarkoitettuun järjestykseen.

A:n arvo	0	0	0	1	1	1	1	1
Haluttu järjes-tys	a5	a4	a3	a21	a1	b7	b6	b5

Kuva 6. A:n arvo 31 siirrettynä tavuun 1. verrattuna haluttuun bittijärjestykseen

Jotta A:n bitit saadaan niille kuuluvalle paikalle, täytyy niitä siirtää bittitasolla eli ns. bitshiftata vasemmalle kolmen bitin verran. Bittisiirtämisoperaattori siis siirtää bittejä, joko vasemmalle tai oikealle haluttujen bittien verran. (20.) Kun A:n arvoja on bittisiirretty vasemmalle kolmen bitin verran, näyttää lopputulos kuvan 7. mukaiselta.

A:n arvo	1	1	1	1	1	0	0	0
Haluttu järjes-tys	a5	a4	a3	a21	a1	b7	b6	b5

KUVA 7. A:n arvo 31 siirrettynä tavuun 1. bittisiirron jälkeen verrattuna haluttuun bittijärjestykseen

Nyt A:n bitit ovat tavussa 1 niille kuuluvilla paikoillaan ja näitä samoja tekniikoita hyödyntäen myös B:n ja C:n arvot voidaan sijoitella ja rajata niille kuuluviin bittialueisiin eri tavuissa. Liitteestä 1 löytyvä C++ -koodi esittelee, kuinka A:n B:n ja C:n arvot voidaan pilkkoa ja järjestellä esimerkkikuvan 3 mukaiseen järjestykseen.

Kun tavut 1, 2 ja 3 ovat muodostettu edellä mainittuja tekniikoita hyödyntäen, tulisi niistä vielä kasata yksi suuri uint32-datayksikkö, jotta esimerkkidatapaketti olisi valmis säilöttäväksi ja lähetettäväksi. Tavujen sijoittaminen uint32-datayksikköön tapahtuisi bittisiirtämällä tavut niille kuuluville paikoilleen samalla tavoin, kuin arvojen liikuttelukin tapahtui tavujen sisällä. Esimerkiksi siirtämällä tavun kolme uint32-datayksikköön, pitäisi kolmatta tavua siirtää vasemmalle 16 bitin verran, jotta se asettuisi sille varattuun kohtaan datayksikköä. Tärkeä huomio on, että kun tavuja 2 ja 1 lisätään uint32-datayksikköön ja niitä siirrellään oikeille paikoilleen, tulee tavut laittaa datayksikköön OR-

bittioperaattorilla, jotteivat datayksikölle asetetut edelliset data-arvot ylikirjoitu, kun käytetään pelkkää =_operaattoria.

Kuvassa 8 olevasta OR-bittioperaattorin totuustaulusta voi lukea, että OR-bittioperaattori antaa arvoksi ykkösen, jos kumpi tahansa operoitavista biteistä on ykkönen tai jos molemmat ovat ykkösiä. (20.) OR-operaattori on siis täydellinen bittioperaattori, kun tavuja liitetään uint32-datayksikköön. Esimerkkikuvan 3 neljäs tavu on täynnä tyhjää, joten sitä ei tarvitse erikseen luoda taikka bittisiirrellä, vaan sen voi asettaa täyteen nollaa alustamalla uint32-datayksikön nollaksi, ennen kuin siihen alkaa bittisiirtämään ja OR-operoimaan tavuja sisään. Liitteissä oleva C++ -koodi esittää myös, kuinka tavut voidaan järjestellä eri tavujärjestykseen koodillisesti.

Logical disjunction

p	*q*	*p* ∨ *q*
T	T	T
T	F	T
F	T	T
F	F	F

KUVA 8. OR-operaation totuustaulu (19.)

Movesensen Whiteboard-kirjasto tarjoaa Makearray-funktion, jonka avulla on mahdollista luoda Whiteboardin omia datataulukkoja sille parametrina toimitetusta taulukosta. Tätä funktiota tulee käyttää luotuun datapakettiin, sillä EEPROM-muistin hallintakoodi, jota projektissa käytetään, ei hyväksy muita kuin whiteboard-array -tyyppisiä taulukkoja sisääntulokseen. Hyvä puoli tässä funktiossa on myös se, että sitä oikein käyttämällä on mahdollista parsia kuvan 3 esimerkin neljäs tavu pois, sillä asettamalla funktion type-kentän uint8_t:ksi ja asettamalla datan koon 3:n uint8_t:n kokoiseksi, tyhjä neljäs tavu ei kopioidu EEPROM-muistille. Kun dataa luetaan EEPROM-muistilta ja valmistellaan Bluetoothin yli lähetettäväksi, voidaan EEPROM-muistilta lukea dataa aina kolmen uint8_t:n verran ja niistä voidaan muodostaa lähetettävä datapaketti, jossa neljäs tavu on tyhjä. Tällä tavoin voidaan välttää tyhjien tavujen tallentaminen EEPROM-muistille.

3 TYÖOSUUS JA SEN VAIHEET

3.1 Suunnittelu ja esityö

Kalibrointisession esityössä tutkittiin ja suunniteltiin kalibrointisession synkronointipaketin sisältöä, sekä BLE:n hallinnointikomentojen toteutusta. BLE-kommunikaation suunnittelussa mobiilikehittäjien kanssa keskusteltiin siitä, kuinka kalibrointisession aloituskomento erotetaan tavallisen session aloituskomennosta. Vaihtoehtoina oli joko valita kalibrointisessiolle oma lähetettävä komentonsa tai vaihtoehtoisesti lisätä kalibrointisession aloituskomento tavallisen session aloituskomennon lisäparametriksi, jonka avulla session tunnistaisi kalibrointisessioksi. Keskustelujen lopputuloksena päätettiin, että kalibrointisessiolle lisätään oma komentonsa, koska se on selkeämpää toteuttaa niin ja tällä tavoin myös estetään kalibrointisession virheellisiä tunnistuksia, joita parametrin kanssa voisi pahimmassa tilanteessa tulla. Tavallisen session aloituskomennon lisäparametritila haluttiin säästää myös tulevaisuutta varten.

Synkronointipaketin suunnitteluvaiheessa käytiin keskusteluja projektissa työskentelevän data-analyytikon/testaajan kanssa. Synkronointipakettiin valittiin sellaisia arvoja askeleentunnistusalgoritmin tarjoamista metriikoista, jotka olisivat tarpeellisia kalibroimisen kannalta. Arvojen valinnan jälkeen päätettiin arvojen tarkkuuksista ja niille varattavista bittimääristä. Näiden keskusteluiden tavoitteena oli optimoida synkronointipaketti riittävän suureksi, ettei arvojen tarkkuus kärsisi, mutta samalla tavoitteena oli pitää paketti riittävän pienenä, ettei EEPROM-muistin tila loppuisi välittömästi kesken.

Keskusteluissa päädyimme valitsemaan kalibrointisession synkronointipakettiin meneviksi arvoiksi tavallisessa sessiossa käytettävien metriikoiden lisäksi vain muutamia lisämetriikoita pitääksemme synkronointipaketin koon sopivana, mutta riittävän monipuolisena. Tällä tavoin kalibrointisession ulostulosta voidaan saada riittävästi informaatiota kalibrointia varten ilman, että EEPROM-muisti loppuisi nopeaa kesken. Valintaa helpottamaan tehtiin myös testauskäyttöön muunnelma senhetkisestä ohjelmistoversiosta, jonka avulla data-analyytikko pystyi tutkimaan ja testailemaan askeleentunnistusalgoritmin eri ulosanteja ja sitä kautta valitsemaan kalibrointisessioon parhaiten soveltuvia metriikoita.

Kun BLE:n hallintakomennoista sekä synkronointipaketin sisällöstä oli saatu alustavasti sovittua, dokumentoitiin nämä päätökset yhteen dokumenttiin, joka sitten käytiin läpi eri osapuolien kanssa. Dokumentti sisälsi päätökset BLE:n hallintakomennoista, synkronointipaketin sisällöstä, sekä myös näiden päätösten pohjalta suunnitellun tarkan bittijärjestyksen synkronointipaketille. Tarkan bittijärjestyksen avulla mobiilipuolen kehitystiimi kykenee myöhemmin kehittämään oman koodinsa, joka sitten purkaa ja järjestelee synkronointipaketeista saadun datan mobiililaitteessa.

3.2 Kalibrointisession vaiheet ja kehitystyö

Kalibrointisession kehitys päätettiin lopulta jakaa kahteen vaiheeseen sensoriohjelmiston osalta. Ensimmäisessä vaiheessa askeldataa talletettiin kahden sekunnin välein luotaviin keskiarvoitettuihin datapaketteihin, kuten tavallisessakin sessiossa tehdään, mutta kaikki lisämetriikat olivat nyt myös mukana näissä datapaketeissa. Myös kalibrointisession BLE-puolen hallinnointikomentojen tunnistus- sekä logiikkakoodi tehtiin jo tässä vaiheessa. Symbio on kehittänyt tätä projektia varten BLE-kehittäjätyökalun, joka kykenee parsimaan synkronointipaketteja ja lähettämään BLE-komentoja, joten tätä työkalua kehitettiin ja muokattiin tukemaan myös kalibrointisessiota. Tällä tavoin testaamiseen ja kehittämiseen oli helppokäyttöinen työkalu, jolla voitiin simuloida mobiililaitteen ja sensorin yhteistoimintaa. Työkalu kykeni lähettämään kalibrointisession aloituskomennon sekä vastaanottamaan ja parsimaan kalibrointisession datapaketteja.

Kun ohjelmiston ensimmäinen versio oli testausvalmis, sitä testattiin yhteistyössä data-analyytikon kanssa käyttäen edellä mainittua testaustyökalua apuna, sillä mobiilipuoli ei ollut vielä aloittanut kalibrointisessioon liittyvää kehitystänsä muiden prioriteettitöiden vuoksi.

Toisessa vaiheessa kalibrointisessio muokattiin tallettamaan dataa jokaisella askeleella keskiarvoitettujen datapakettien sijaan ja mikäli session ollessa käynnissä sensorinkäyttäjä on paikallaan, luodaan datapaketteja puolen sekunnin välein sen aikaa, kunnes käyttäjä alkaa jälleen liikkumaan. Tällä tavoin estetään sykemittauksen tulosten menettäminen käyttäjän ollessa paikallaan. Koska datan säilömislogiikka muuttui kalibrointisession kohdalla tasavälein keskiarvottamisesta askelkohtaiseen säilömiseen, täytyi sensorille luoda uudenlainen logiikka, jota suorittaa kalibrointisession aikana. Tämä muutos oli suurimmalta osin nopea tehdä, sillä pohjimmiltaanhan sensorin ei tarvitse

kuin tunnistaa askelalgoritmin ulosannista, milloin pelaaja lähtee liikkeelle tai pysähtyy, ja sen perusteella toimia toivotusti. Askeleentunnistusalgoritmissa kuitenkin on muutamia ei-toivottuja ominaisuuksia, jotka voivat vaikeuttaa pysähtymisen tunnistamista, joten niitä varten piti luoda muutamia erikoisehtoja ohjelmaan, jotta sensori osaisi toimia oikein myös näiden ominaisuuksien ilmentyessä.

Kun kalibrointisession toinen vaihe oli valmis, testattiin sitä data-analyytikon kanssa käymällä läpi sarjoja erilaisia testejä sekä toimistoympäristössä, että oikeassa käyttöympäristössä. Testien tarkoituksena oli nähdä, kuinka kalibrointisession toteutus reagoisi erilaisissa erikoistapauksissa sekä oikeassa käytössä. Näiden testien perusteella kartoitimme mahdollisia virheitä koodista, joita korjattiin sitä mukaa, kun niitä löydettiin. Näiden muutamien virheenkorjaustestausten jälkeen kalibrointisessio oli valmis. Mobiilipuolen prioriteettitöiden ja muiden hidasteiden vuoksi kalibrointisession toimivuutta mobiililaitteen kanssa ei päästy testaamaan tämän opinnäytetyön aikaan, mutta se tullaan tekemään heti kun mahdollista.

4 POHDINTA

Sensorille kehitetyn kalibrointisessio-ominaisuuden tavoitteena oli saada sensori aloittamaan kalibrointisessio BLE-komennosta. Kalibrointisession aikana sensorin tavoitteena oli kerätä kalibrointisession uuden tietorakenteen mukaisiin datapaketteihin dataa askeleentunnistusalgoritmilta ja säilöä tämä data sensorin EEPROM-muistiin. Kalibrointisession päätteeksi sensorin oli tavoitteena kyetä lähettämään nämä datapaketit synkronointipakettina mobiililaitteelle.

Kalibrointisession kehityksessä asetetut tavoitteet saavutettiin ja toteutus saatiin kehitettyä vastaamaan sille asetettuja tavoitteita. Käyttäjätestauksien jälkeen ja pitemmän ajan kuluessa kalibrointisession toimivuus ja kalibrointiominaisuuden tuoma lisäarvo selvenee paremmin, sillä tällä hetkellä kalibrointisessio on vasta sisäisessä testauksessa ja odottaa mobiilipuolen tukea. Kun kalibrointisessio otetaan käyttöön kaikissa olemassa olevissa tuotteissa, on helpompi arvioida sen kokonaisvaltaista onnistumista, mutta alustavasti tilanne näyttää hyvältä.

Kalibrointisession kehittäminen ja toteuttaminen oli mielenkiintoista tehdä, sillä sen parissa pääsi työskentelemään monen eri osa-alueen ja koodikerroksen parissa aina bittitasolta arkkitehtuuritasolle asti. Olen työskennellyt Beyond-Pulse -projektin parissa jo vuoden verran ja kalibrointisession kaltaisen uuden ja ison ominaisuuden suunnittelu sekä kehittäminen oli oikein antoisaa ja virkistävää. Kehittäminen sujui myös hyvin nopeasti ja tehokkaasti aikaisemman tietotaidon ansiosta, sillä vaikka Movesense-ympäristö on monipuolinen ja hyvin tukea tarjoava, on sen parissa työskentelemisessä alkuvaiheessa jyrkkä oppimiskynnys.

Koska mobiilikehityksen puolella kesti melko pitkään aloittaa kalibrointisessioon tulevien mobiiliominaisuuksien kehitys, päätettiin mobiilikehityksessä hypätä käyttämään suoraan viimeistä versiota eli kakkosvaiheen kalibrointisession sensoriversiota, joten ensimmäinen vaihe jäi kokonaan käyttämättä. Näin jälkiviisaana tämän ensimmäisen vaiheen olisi voinut jättää kokonaan tekemättä, sillä näin lopulta se paljastui vain turhaksi työksi. Tämänkaltaisia asioita voi olla vaikea arvioida etukäteen, sillä projektien prioriteetit ja aikataulut muuttuvat monesti hyvinkin pikaisesti ketterässä ohjelmistokehityksessä.

LÄHTEET

1. Learn about Bluetooth. Radio Versions. 2020. Bluetooth SIG. Saatavilla: https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/radio-versions/. Hakupäivä 8.3.2020
2. Bluetooth Special Interest Group. Wikipedia. 2020. Saatavilla: https://en.wikipedia.org/wiki/Bluetooth_Special_Interest_Group . Hakupäivä 10.3.2020
3. EEPROM. 2020. Wikipedia. Saatavilla: https://fi.wikipedia.org/wiki/EEPROM . Hakupäivä 29.3.2020
4. Esineiden internet. 2020. Wikipedia. Saatavilla: https://fi.wikipedia.org/wiki/Esineiden_internet. Hakupäivä 8.3.2020
5. Young, Julie. 2020. Metrics. Investopedia. Saatavilla: https://www.investopedia.com/terms/m/metrics.asp . Hakupäivä 29.3.2020
6. System on a chip. 2020. Wikipedia. Saatavilla: https://en.wikipedia.org/wiki/System_on_a_chip. Hakupäivä 8.3.2020
7. Data synchronization. 2019. Wikipedia. Saatavilla: https://en.wikipedia.org/wiki/Data_synchronization . Hakupäivä: 29.3.2020
8. Tavujärjestys. 2020. Wikipedia. Saatavilla: https://fi.wikipedia.org/wiki/Tavuj%C3%A4rjestys. Hakupäivä 8.3.2020
9. What is Beyond Pulse? 2020. Beyond Pulse. Saatavilla: https://beyondpulse.com/en/. Hakupäivä 8.3.2020
10. Suunto open development environment offers rapid prototyping of Bluetooth LE motion sensing solutions.2017. Nordic Semiconductor. Saatavilla: https://www.nordicsemi.com/News/2017/08/Movesense-platform-from-Suunto. Hakupäivä 8.3.2020
11. Opportunities. 2020, Movesense. Saatavilla: https://www.movesense.com/opportunities/. Hakupäivä 8.3.2020
12. Movesense system overview. 2020. Movesense. Saatavilla http://www.movesense.com/docs/system/system_overview/. Hakupäivä 8.3.2020
13. Getting started. 2020. Movesense. Saatavilla: http://www.movesense.com/docs/esw/getting_started/ .Hakupäivä 8.3.2020
14. Bluetooth Low Energy protocol stack. 2019. Nordic Semiconductor. Saatavilla: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fsds_s140%2FSDS%2Fs1xx%2Fble_protocol_stack%2Fble_protocol_stack.html. Hakupäivä 8.3.2020

15. Generic Access Profile (GAP). 2016. Texas Instruments. Saatavilla: http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_40_00_45/docs/blestack/ble_user_guide/html/ble-stack-3.x/gap.html. Hakupäivä 8.3.2020
16. Townsend, Kevin - Cufi, Carles -Akiba - Davidson, Robert. 2020. Getting started with Bluetooth Low Energy, chapter 4. GATT (Services and Characteristics). O'Reilly. Saatavilla: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html. Hakupäivä 1.3.2020
17. 16 Bit UUIDs for Members. 2020. Bluetooth SIG. Saatavilla: https://www.bluetooth.com/specifications/assigned-numbers/16-bit-uuids-for-members/. Hakupäivä 10.3.2020
18. Mask (computing). 2020. Wikipedia. Saatavilla: https://en.wikipedia.org/wiki/Mask_(computing) Hakupäivä: 6.3.2020
19. Truth table. 2020, Wikipedia. Saatavilla:https://en.wikipedia.org/wiki/Truth_table#Truth_table_for_all_binary_logical_operators. Hakupäivä 6.3.2020
20. Bittioperaatio. 2020. Wikipedia. Saatavilla: https://fi.wikipedia.org/wiki/Bittioperaatio. Hakupäivä: 6.3.2020

```cpp
#include <iostream>
#include <bitset>
using namespace std;

int main()
{
    /* esimerkin a, b ja c arvot. Näitä muuttamalla voit testailla koodin toimintaa. uint8_t maksimiarvo = 255 ja
    uint16_t maksimiarvo = 65535*/
    uint8_t aValue = 31;
    uint8_t bValue = 127;
    uint16_t cValue = 4095;

    /* alustetaan esimerkin tavut 1 - 4 sekä lähetettävää ja vastaanotettavaa datapakettia kuvastavat muuttujat */
    uint8_t byte1 = 0;
    uint8_t byte2 = 0;
    uint8_t byte3 = 0;
    uint8_t byte4 = 0;
    uint32_t dataPackage = 0;
    uint32_t rvcDataPackage = 0;

    /* Bittimaskit esimerkin bittimaskit */
    uint8_t aBitMask = 0x1F;    // 0001 1111
    uint8_t bBitMask1 = 0x70;   // 0111 0000
    uint8_t bBitMask2 = 0x0F;   // 0000 1111
    uint16_t cBitmask1 = 0xF00; // 1111 0000 0000
    uint16_t cBitMask2 = 0x0FF; // 0000 1111 1111


    /* Bitmaskataan ja bittisiirretään arvojen bitit niille kuuluville paikoille*/
    byte1 |= (aValue & aBitMask) << 3;
    byte1 |= (bValue & bBitMask1) >> 4;

    byte2 |= (bValue & bBitMask2) << 4;
    byte2 |= (cValue & cBitmask1) >> 8;

    byte3 |= cValue & cBitMask2;

    /* siirretään lähetettävät tavut little endian -järjestykseen*/
    dataPackage = byte1;
    dataPackage |= byte2 << 8;
    dataPackage |= byte3 << 16;
    dataPackage |= byte4 << 24;

    /* siiretään tavut vastaanotettuun datapakettiin big endian -järjestykseen*/
    rvcDataPackage = byte1 << 24;
    rvcDataPackage |= byte2 << 16;
    rvcDataPackage |= byte3 << 8;
    rvcDataPackage |= byte4;

    /*Luodaan muutujien arvoista bittipresentaatio tulostamisen helpottamiseksi*/
    bitset<8> byte1InBinary(byte1);
    bitset<8> byte2InBinary(byte2);
    bitset<8> byte3InBinary(byte3);
    bitset<8> byte4InBinary(byte4);
    bitset<32> dataPackageInBinary(dataPackage);
    bitset<32> rvcDataPackageInBinary(rvcDataPackage);

    cout << "byte1 in binary : " << byte1InBinary << endl;
    cout << "byte2 in binary : " << byte2InBinary << endl;
    cout << "byte3 in binary : " << byte3InBinary << endl;
    cout << "byte4 in binary : " << byte4InBinary << endl;
    cout << "rvcDataPackage in binary (Big endian) : " << rvcDataPackageInBinary << endl;
    cout << "DataPackage in binary (little endian) : " << dataPackageInBinary << endl;
}
```