Jonne Pätzold

Kiihtyvyysanturin kehittäminen

Opinnäytetyö
Tieto- ja viestintätekniikan koulutus

2019

Kaakkois-Suomen
ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Jonne Pätzold	Insinööri (AMK)	Toukokuu 2019
Opinnäytetyön nimi Kiihtyvyysanturin kehittäminen		36 sivua 1 liitesivua
Toimeksiantaja Gleap Health Technologies Oy		
Ohjaaja Niina Mässeli		

Tiivistelmä

Tässä opinnäytetyössä käydään läpi kiihdytyksiä mittaavaan elektroniikkalaitteen kehitystä alusta alkaen. Kehitysvaiheissa rakennettiin kaksi erilaista laitetta. Ensimmäinen laite on kehitetty Arduino-kehitysympäristöön Adafruit Flora -alustalle ja sen tarkoitus on antaa suuntaa siitä, miten komponenteista kasattu oma laite kannattaa suunnitella. Kehitystyötä varten kirjoitettiin myös erillinen BTDebug-ohjelma Android-laitteille, jolla voitiin testata, miten yhteyden luominen, tietojen vastaanottaminen sekä käskyjen lähettäminen langattomasti onnistuu laitteen ja kännykän välillä.

Opinnäytteessä käydään läpi myös oman laiteajurin rakentaminen C-kielellä. Laiteajuri tehdään NPX:n valmistamalle FXLS8471Q-kiihtyvyysanturille, koska erillistä valmista kirjastoa laitteelle ei ollut saatavissa. Ajuriin tehdään funktiot anturin alustukselle, lukemiselle, kirjoittamiselle ja kiihtyvyysdatan muuttamiselle 14-bittisestä 16-bittiseksi. Laitteistoajuria käytetään laiteohjelmassa anturin kontrollointiin ja asetusten määrittelemiseen.

Työkaluina opinnäytetyössä käytetään erilaisia mittareita, kuten yleismittaria ja oskilloskooppia. Laitteen kasaamisessa käytetään juotinta, juotospastaa, juotostelinettä sekä juotostinaa. Pinnat, joille ei haluta kontaktia, peitetään lämmönkestävällä teipillä ja signaaliväylinä käytetään paksun sähköjohdon sisältä löytyviä ohuita metallisäikeitä. Ohjelmakehityksessä Arduino-ympäristöön käytetään Arduino IDE:ä ja ARM-ympäristössä Segger Embedded Studiota. Toisen laitteen piirilevy suunnitellaan KiCAD EDA -ohjelmistolla, joka on avoimeen lähdekoodin perustuva monen alustan CAD -ohjelmisto. Tämä ohjelmisto on suunniteltu piirilevyjen tekemiseen ja sisältää piirikaavion ja piirilevyn suunnitteluun tarvittavat työkalut. Laitetta ohjelmoidaan erillisellä laiteohjelmoijalla, joka on mukana nRF52 -kehitystyökalussa.

Asiasanat

ohjelmointi, elektroniikka, kehitys

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Jonne Pätzold	Barchelor of Engineering	May 2019

Thesis title

Development of an accelerometer

36 pages
1 pages of appendices

Commissioned by

Gleap Health Technologies Oy

Supervisor

Niina Mässeli

Abstract

This thesis examined the development of the electronic device for measuring accelerations from the beginning. Two different devices were built in the development stages. The first device was developed for the Arduino development platform on the Adafruit Flora and was designed to guide on how to design your component assembled device. For development, a separate BTDebug program for Android devices was also written to test how connectivity, receiving data, and sending commands function wirelessly between the device and the mobile phone.

The thesis also examines how to build your own device driver in C-language. The device driver was made for an NPX FXLS8471Q accelerometer, because a separate ready-made library for the device was not available. Functions were made in the driver for initializing, reading, writing, and converting acceleration data from 14-bits to 16-bits. The hardware driver was used in the firmware to control the sensor and define the settings.

The tools used in the thesis were various meters, such as a multimeter and an oscilloscope. A soldering iron, solder paste, third-hand tool and solder were used to assemble the device. Surfaces that do not require contact were covered with heat-resistant tape and thin metal strands found inside the thick electrical conductor were used as signal paths. Arduino IDE was used on development in Arduino environment and on the ARM environment, Segger Embedded Studio was used. The circuit board of the other device was designed by KiCAD EDA software, which is an open source multi-platform CAD software. This software was designed to make circuit boards and includes tools for circuit diagram and circuit board design. The device was programmed with a separate firmware programmer included in the nRF52 development tool.

Keywords

programming, electronics, development

SISÄLLYS

1 JOHDANTO 7

2 KEHITYSTYÖKALUT 8

2.1 KiCad EDA 9

2.2 Arduino ja Arduino IDE 9

2.3 nRF5 SDK ja Segger Embedded Studio 10

2.4 Development Kit 11

2.5 Softdevice 11

2.6 BTDebug 12

2.7 Oskilloskooppi 12

3 KOMMUNIKOINTIVÄYLÄT 13

4 PROTOTYYPPIEN OSAT 14

4.1 Adafruit Flora 14

4.2 Bluetooth SoC 15

4.3 Kiihtyvyysanturit 16

4.4 MicroSD-kortti 17

4.5 nRF52832 SoC 18

4.6 Latauspiiri 19

5 ENSIMMÄINEN PROTOTYYPPI 20

6 LAITTEISTOLÄHEINEN OHJELMOINTI 21

6.1 Keskeytykset 22

6.2 Bluetooth Low Energy 22

6.3 Laiteohjelma 23

6.4 Laiteohjelman tärkeimmät funktiot 25

6.5 FXLS8471Q-laiteajuri 27

7 TOINEN PROTOTYYPPI 32

7.1 Juottaminen 33

8 JOHTOPÄÄTÖS JA JATKOKEHITYS 33

LÄHTEET 35

KUVALUETTELO 36

LIITTEET

Liite 1. Toisen prototyypin kytkentäkaavio

LYHENTEET JA SANASTO

CAD	Computer-Aided Design, tietokoneavusteinen suunnittelu
DEBUG	Virheenetsintä, joka mahdollistaa prosessorin tarkkailun käsky kerrallaan
DK	Development Kit, Kehitystyökalu.
EDA	Electronic Design Automation, elektroninen suunnitteluautomaatio
FAT	File Allocation Table, tiedostojärjestelmä
IDE	Integrated Development Environment, ohjelmistoympäristö
IO	Input / Output, Sisääntulo / Ulostulo
LE	Low Energy, alhainen energiankulutus
LiPo	Lithium polymer, litiumpolymeeri
LSB	Least Significant Bit. Vähiten merkitsevä bitti
MISO	Master In, Slave Out. Isännän sisääntulo, rengin ulostulo
MOSI	Master Out, Slave In. Isännän ulostulo, rengin sisäätulo
MSB	Most Significant Bit. Eniten merkitsevä bitti
PCB	Printed Circuit Board. Piirilevy
RAM	Random-Access Memory. Keskusmuisti
ROM	Read Only Memory. Lukumuisti
SCK	Serial Clock. Sarjallinen kello
SCL	Serial Clock Line. Sarjallinen kellolinja
SD	Secure Digital
SDA	Serial Data Line. Sarjatietolinja
SDK	Software Development Kit. Ohjelmointirajapinta
SPI	Serial Peripheral Interface. Sarjaliitäntä
SoC	System on a Chip. Järjestelmäpiiri
SS	Slave Select. Rengin valinta
TWI	Two wire interface. Kahden johdon liitäntä
USB	Universal Serial Bus. Yleinen sarjaliikenneportti

1 JOHDANTO

Opinnäytetyössä tutustutaan kiihtyvyysanturin kehitykseen. Työ tehdään Gleap Health Technology Oy:lle, joka on perustettu vuonna 2016 ja sijaitsee Haminassa. Yrityksen tarkoitus on kehittää mobiilisovellus, joka auttaa ihmisiä terveyden seurannassa. Mobiiliapplikaatioon on tarkoitus lisätä tuki myös erilliselle laitteelle, jota tässä opinnäytetyössä kehitetään. Työn tarkoitus on lisätä yrityksen tietotaitoa siitä, mitä kaikkea elektroniikkalaitteen kehittäminen vaatii. Tätä tietoa tarvitaan, kun laitetta aloitetaan rakentamaan elektroniikkavalmistajan kanssa. Tieto helpottaa yritysten välistä kommunikaatiota ja antaa osviittaa siihen, mitä laitteelta voidaan vaatia.

Opinnäytetyö on laaja-alainen ja siihen kuuluu elektroniikan sekä ohjelmiston rakentamista. Ensimmäinen prototyyppi tehdään Adafruit Flora -kehitysalustalle. Kehittämisessä käytetään Arduino IDE -kehitysympäristöä. Flora vaatii myös erillisiä moduuleita, jotka lisäävät siihen tarvittavan toiminnallisuuden. Ensimmäisen version tarkoitus on antaa kokonaisvaltainen kuva siitä, mitä kaikkea tarvitaan toimivan laitteen rakentamiseen. Tämän jälkeen toinen prototyyppi kasataan. Se kasataan erillisistä komponenteista reikälevylle ja ohjelmoinnissa käytetään Seggerin Embedded Studiota. Toisen prototyypin ohjelmointi osoittautui vaikeaksi, koska nRF5 SDK on rakennettu modulaarisesti ja asetustiedostossa on yli 12 tuhatta riviä asetuksia. Suurin osa ajasta menikin SDK:n toiminnan selvittämiseen. Toisen prototyypin rakentaminen aloitettiin tekemällä KiCAD-ohjelmalla kytkentäkaavio, johon kaikki tarvittavat komponentit lisättiin ja tämän kytkentäkaavion perusteella komponentit juotettiin reikälevylle.

Projektissa käytettiin monia eri työkaluja. Arduino IDE:llä kirjoitettiin ensimmäisen prototyypin ohjelmakoodi. IDE:ssä on kaikki tarpeellinen ohjelman kehittämiseen. Adafruitin www-sivuilta löytyi suoraan kaikki tarpeelliset kirjastot tähän kehitysympäristöön, jolloin moduulien ohjelmointi oli vaivatonta. Segger Embedded Studiolla kirjoitettiin toisen prototyypin ohjelmakoodi ja vaikeusaste kasvoi reilusti Arduino-ympäristöstä. Toisen prototyypin ohjelmoimiseen käytettiin erillistä laiteohjelmoijaa. Laiteohjelmoija on mukana nRF5-kehitysalus-

tassa ja sillä voitiin ladata, sekä debugata erillisiä prosessoreja. Kehittämisessä käytettiin yleismittaria ja oskilloskooppia, jolla voitiin tutkia prototyypin signaalien oikeellisuutta.

Laitetta lähtiin kehittämään näiden määritelmien mukaan. Laitteen tulee olla mahdollisimman pieni ja vähävirtainen. Latausvirran laite saa microUSB-liittimen kautta. Laite mittaa kiihtyvyyksiä ja tallentaa mittausdatan MicroSD-kortille. Laitteen tulee kommunikoida mobiiliapplikaation kanssa Bluetooth-yhteydellä siten, että sieltä saadaan kaikki mobiiliapplikaation tarvitsemat tiedot.

Vaikkei kehitystyössä saatukaan laitetta siihen kuntoon, että sarjatuotanto olisi ollut mahdollista, antoi se hyvin laajan kuvan kaikista asioista, joita elektroniikkalaitteen kehittämiseen tarvitaan. Kumpikin prototyyppi saatiin toimimaan siten, että niiden koekäyttö oli mahdollista.

2 KEHITYSTYÖKALUT

Kehitystyökaluja on paljon erilaisiin tehtäviin ja alustoille. Projektissa käytettiin kuitenkin mahdollisimman paljon kaikille saatavissa olevia työkaluja. Koska kehityksessä on tarkoitus kerryttää tietoa, pyrittiin projektissa käyttämään suosittuja ohjelmistoja. Suosittuja ohjelmia käyttää moni, jolloin myös työkalujen ohjeita on paljon saatavissa. Iloinen uutinen oli se, että Segger Embedded Studio on ilmainen, jos sillä ohjelmoidaan Nordic Semiconductorin piirejä. Tämä työkalu oli suhteellisen uusi markkinoilla, joten siinä oli pieniä virheitä. Alussa se kaatui satunnaisesti, kun välilehtiä suljettiin. Tämä kuitenkin korjattiin yhdessä versiopäivityksessä.

Jotta työn edistyminen on nopeaa ja vaivatonta, on hyvä käyttää oikeanlaisia työkaluja. Tarvittavia työkaluja ovat muun muassa laiteohjelmoija, kehitysympäristö, kääntäjä ja oikeat ohjelmakirjastot. Laiteohjelmoijaa tarvitaan binääritiedostojen siirtämiseen laitteeseen, koska nRF52832-piirissä ei ole erillistä tukea tälle ominaisuudelle. Yleismittarilla varmistettiin, että laitteet saivat tarvitsemansa määrän jännitettä ja että signaali ei katkennut missään välissä matkaa. Oskilloskoopilla voitiin tutkia tarkemmin, minkä näköistä signaalia väylissä kulki.

Laitteiden kokoamisessa käytettiin juotinta, juotostinaa, juotostelinettä, pinsettejä sekä lämpöä kestävää teippiä. Teippiä käytettiin eristämään kontaktipintoja, joita ei tarvittu. Ensimmäisen prototyypin kasaamisessa ei juotinta paljon tarvittu, ainoastaan johtojen liitosten varmistamiseen kontaktipinnoille. Toisen prototyypin kasauksessa, kolvi ja pinsetit olivat välttämättömiä.

2.1 KiCad EDA

KiCadin julkaisi vuonna 1992 sen alkuperäinen kehittäjä, Jean-Pierre Charras, mutta nykyjään sitä kehitetään erillisen KiCad-kehitystiimin kanssa. KiCad on avoimen lähdekoodin ohjelmistopaketti, joka toimii Windows-, Linux- ja IOS-ympäristössä. Ohjelmalla voidaan suunnitella kytkentäkaavio ja piirilevy. Ohjelmaan löytyy paljon valmiita komponentteja suurten komponenttivalmistajien sivuilta, jolloin niitä ei tarvitse ohjelmassa tehdä erikseen. Koska ohjelmaa on kehitetty pitkän aikaa, niin se ei ole käyttäjäystävällisin, mutta siitä löytyy kaikki tarpeellinen. Myös erilaisia tutoriaaleja sen käyttämiseen löytyy paljon. (About KiCad s.a.)

Vaikka ohjelman graafinen ulkoasu ei vastannut tätä päivää, niin sillä oli helppoa tehdä piirustukset (liite 1) valmiiksi. Monet piirilevyvalmistajat tukevat KiCadilla tehtyjä piirilevymalleja, jolloin valmis piiri olisi voitu suoraan tilata KiCadin tiedostoilla. KiCad eroaa Autodesk Eagle -ohjelmistosta myös siten, että valmiin piirilevyn 3D-mallinnukseen ei tarvita erillistä mallinnusohjelmaa. Eaglessa mallinnukseen tarvitaan Autodeskin Fusion 360 -ohjelmisto.

2.2 Arduino ja Arduino IDE

Arduino on avoin kehitysalusta, joka perustuu helposti käytettäviin laitteisiin ja ohjelmiin. Arduinon kehitti Ivrea Interaction Design Institute. Se kehitettiin työkaluksi, jolla voidaan nopeasti tehdä prototyyppejä, ilman syvempää tietoa elektroniikasta tai ohjelmoinnista. Vuosien saatossa Arduino on toiminut monien projektien alustana ja sitä käyttää niin ammattilaiset kuin aloittelijat. Arduino ja Arduino-yhteensopivat laitteet ovat yleensä halpoja ja täten ovat kaikkien saatavilla. (What is Arduino s.a.)

Ensimmäisen prototyypin ympäristönä toimii Arduino-yhteensopiva Flora, joten luontevin tapa ohjelmoida laitetta on Arduinon oma IDE. Siinä ei ole paljon

moderneista ohjelmointiympäristöistä tuttuja ominaisuuksia. Kehitysympäristössä voidaan helposti ladata laitteisto-ohjelmia, ajureita sekä kirjastoja. Kehitysympäristöön löytyy monia esimerkkejä, joita katsomalla oppii nopeasti asioita.

2.3 nRF5 SDK ja Segger Embedded Studio

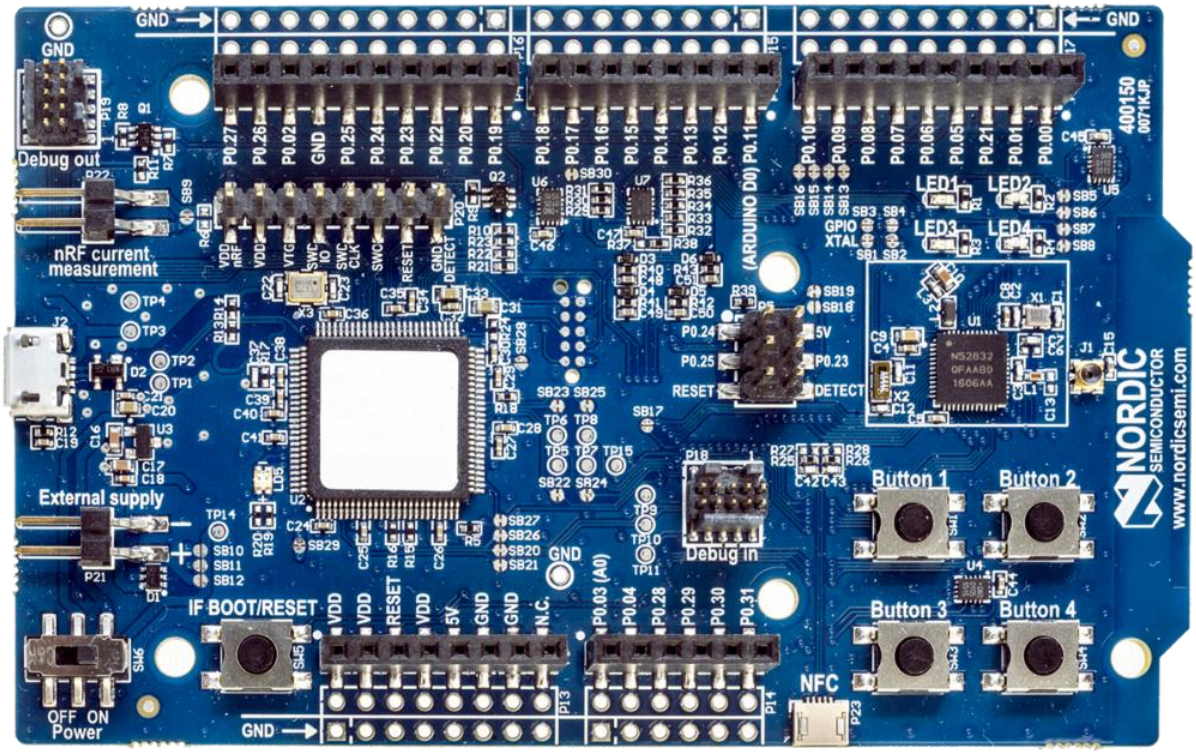
SDK on kirjasto, missä on testattua, valmiiksi kirjoitettua ohjelmointikoodia helpottamaan kehitystä. Kirjasto pitää sisällään paljon esimerkkikoodia, jota tutkimalla saa käsityksen siitä, miten laiteohjelmat toimivat. Kirjastosta löytyy melkein kaikki laitteen ohjelmiston kehittämisessä tarvittavat ominaisuudet. Ainoastaan kiihtyvyysanturin laitteistoajuri puuttui. Koska erilliset väyläohjaimet löytyivät kirjastosta, puuttuvan laitteistoajurin pystyi kirjoittamaan ilman suurempaa vaivaa.

Segger Embedded Studio on täydellinen kehitysympäristö sulautettujen järjestelmien hallintaan, rakentamiseen, testaamiseen ja käyttöönottoon. Se sisältää C/C++-kääntäjän ja debuggerin. Embedded Studio tukee myös suoraan laiteohjelmoijaa, jolloin ohjelma on helppo ladata laitteeseen. (Segger Embedded Studio 2019.)

Toisen prototyypin kehityksessä käytetään Segger Embedded Studiota, jota nRF5 SDK tukee versiosta 14.1.0 alkaen. Kehitysohjelmisto on myös ilmainen Nordic Semiconductorin piireille. IDE:n käyttäminen ohjelmoinnissa helpottaa kehittämistä, koska siinä on automatisoitu tarpeellisia asetuksia. Myös erillinen tuki virheenetsinnälle helpottaa kehitystyötä. Koska nRF5 SDK tukee suoraan tätä kehitysympäristöä, niin esimerkkien tutkiminen ja kääntäminen sujuu helposti. Yksi esimerkki voi vaatia monia eri kirjastoja ja niiden sisällyttämistä projektiin. Tämä tekee ohjelman kehittämisestä vaikeaa. Kun ohjelmistoprojektin avaa suoraan kehitysympäristöllä, niin kaikki tarvittavat kirjastot ovat mukana. Kehitysympäristö sisältää myös erillisen prosessorin käynnistyskoodin, jolloin sitä ei tarvitse erikseen kirjoittaa. Käynnistyskoodilla tarkoitetaan ohjelmaa, joka asettaa pino-osoittimen ja alustaa laitteistokohtaiset asetukset. Tähän kuuluu tietolohkojen muistipaikkojen, sekä prosessorin nopeuden asettaminen.

2.4 Development Kit

Koska ohjelmaa ei pysty ajamaan tietokoneessa, niin tarvitaan erillinen laiteohjelmoija. Tällä ohjelmoijalla binääridata saadaan ladattua ARM M4F-prosessorin käyttämään Flash-muistiin. Kehitysalusta pitää sisällään Segger J-Link OB -laiteohjelmoijan. nRF52 DK (Kuva 1) on erillinen kehitysalusta, jolla voidaan tutkia prosessorissa toimivan ohjelman suoritusta käsky kerrallaan. Se on täysin yhteensopiva Segger Embedded Studion kanssa.



Kuva 1. nRF52 DK

2.5 Softdevice

Softdevice on laiteohjelmisto, jonka Nordic Semiconductor on kehittänyt. Tämä ohjelma on suljettu, eikä sen lähdekoodeja ole saatavilla. Softdevice hoitaa kaiken langattomaan liikenteeseen tarvittavat toiminnot keskeytyksillä ja toimii täysin itsenäisesti. NRF5 SDK -kirjastossa on softdevicen asetuksen asentamiseen kaikki tarpeellinen ja sitä kautta voidaan kontrolloida laitteen mainostamaa nimeä, mainostamisen aikaa, sekä muita hyödyllisiä muuttujia. Softdevice asennetaan laitteeseen ennen varsinaista ohjelmaa.

2.6 BTDebug

Jotta laitteen testaaminen olisi helpompaa, on hyvä kirjoittaa testiohjelmistot, jotka ovat varta vasten suunniteltu järjestelmän testaamiseen. Tähän tarkoitukseen BTDebug luotiin. Testiohjelma on tehty Android-järjestelmään ja se on kirjoitettu Javalla. Sillä on helppo testata yhteyttä, sekä syöttää ja vastaanottaa dataa sellaisessa muodossa, jollaista haluaa käyttää ohjelmistossa. Ohjelman voi myös jatkojalostaa automaattisesti toimivaksi testausohjelmaksi, jolloin se voi väsymättä kerätä tietoja laitteen toiminnasta.

2.7 Oskilloskooppi

Yritykselle hankittiin Rigol DS1054Z -oskilloskooppi. Tässä mallissa oli valmiiksi kaikki tarpeellinen pienelektroniikan kehittämiseen. Laitteen tärkeimmät ominaisuudet ovat:

- 50 MHz kaistanleveys
- 4 kanavaa
- 1 GSa/s reaaliaikainen näytteenottotaajuus
- 24 miljoonaa näytepistettä tallennetussa aaltomuodossa
- 7 tuuman 800x480 WVGA-näyttö
- sarjaväylädekooderi
- nauhoitus ja toisto
- edistyneet liipaisin funktiot.

Oskilloskoopilla voidaan tutkia tarkemmin, miten tieto kulkee signaalissa. Lähinnä oskilloskoopilla tarkasteltiin väylässä liikkuvan datan oikeellisuutta. Siihen tarkoitukseen sarjaväylän dekooderi oli täydellinen, koska sillä näki koko sarjaväylän liikenteen arvoineen. Oskilloskoopilla ratkaistiin yksi vaikea ongelma. Ongelma ilmeni silloin, kun kiihtyvyysanturi ja MicroSD-kortti alustettiin samaan aikaan. Tämä johtui siitä, että molemmissa laitteissa oli erillinen tapahtumaketju, jolla laite saadaan valitsemaan oikeanlainen väylä alustuksessa. Ja kun toinen alustus oli kesken, niin silloin toisen laitteen alustuksen aikaikkuna meni ohi, jolloin laitteeseen jäi väärä väylä. Tämä taas tarkoitti sitä, että laite ei suostunut kommunikoimaan.

3 KOMMUNIKOINTIVÄYLÄT

Ilman kommunikointiväyliä, laitteiden välinen yhteys on mahdotonta. Siksi on erittäin tärkeää tietää, millä eri tavoilla laitteet voivat kommunikoida keskenään. Yleisesti elektroniikassa käytetään I²C- tai SPI-väylää ja nämä väylät löytyvät melkein kaikista laitteista.

Molempien laitteiden prosessorit tukevat suoraan TWI- ja SPI-väyliä. SD-kortti sekä kiihtyvyysanturi tukevat molemmat SPI-väylää, joten projektissa käytetään kyseistä väylää. SPI eroaa TWI-väylästä siten, että SPI-väylä tarvitsee kommunikointiin neljä signaalia kuin taas TWI-väylä ainoastaan kaksi. SPI-väylä voi vastaanottaa ja lähettää samaan aikaan, kun taas TWI-väylä voi kerralla, joko lähettää tai vastaanottaa tietoa.

Väylien toiminnallisuuden testaaminen on hankalaa ja vaatii oskilloskooppia tai erillistä sarjaväylänlukija laitetta. Projektissa väylää testailtiin ohjelmallisesti, sekä lähetettävä data tarkastettiin oskilloskoopilla, jotta väylän toimintavarmuus pystyttiin varmistamaan. Kiihtyvyysanturissa oli erillinen WHOAMI-rekisteri, josta lukemalla laite lähetti teknillisissä ohjeista löytyvän vastauksen, joka tässä tapauksessa oli 0x6A. Kun laitteen vastausta verrataan ohjekirjassa annettuun arvoon, voitiin päätellä, toimiiko laite oikein.

Two wire interface eli TWI-väylä on melkein sama väylä kuin I²C, mutta pienillä eroavaisuuksilla. TWI-väylä on Atmelin ja muiden valmistajien vastine I²C-väylälle, jolla voidaan välttää I²C-väylän tavaramerkkirikkomukset. nRF52832 käyttää TWI-väylää I²C-väylän sijasta, mutta on yhteensopiva I²C-väylän kanssa. Nimensä mukaan TWI-väylä käyttää kahta signaalia lähettämiseen sekä vastaanottamiseen. Nämä signaalit ovat nimeltään SDA sekä SCL. TWI-väylään voidaan liittää monia renkejä, ja erillisellä osoitteella valitaan minkä rengin kanssa kommunikoidaan.

Standardi SPI-väylälle on kehittänyt Motorola. Laitteet kommunikoivat isäntä/renki menetelmällä. Väylä käyttää kommunikointiin neljää signaalia, joilla lähetetään ja vastaanotetaan dataa, valitaan oikea laite sekä annetaan väyläkohtainen kellotaajuus. Signaalit ovat nimeltään MOSI, MISO, SS ja SCK. Väylään voidaan kytkeä monia laitteita, jolloin SS-signaalilla valitaan

laite, joka vastaanottaa datan. Tähänkin väylään voidaan liittää monia laitteita, mutta jokaisessa laitteessa pitää olla oma SS-signaali, jolla voidaan valita käskytettävä laite.

4 PROTOTYYPPIEN OSAT

Ensimmäisen prototyypin osat on ostettu Adafruit Industrylta, ja jokainen osa on erillinen moduuli. Näitä moduuleja yhdistelemällä prototyyppi rakentui nopeasti valmiiksi ja testattavaksi. Prototyypistä puuttui tässä vaiheessa lipoakun lataaminen, koska Flora ei tätä tukenut ja erillinen moduuli olisi tuonut liikaa kokoa laitteelle, jolloin testaaminen olisi kärsinyt. Akun lataaminen ratkaistiin erillisellä lataustelakalla. Prototyyppi oli myös liian iso ja kömpelö käytettäväksi muuten, kuin laitteiston testaamiseen. Prototyyppiä yritettiin aluksi kasata erilliselle koekytkentälevylle. Kaikki muut komponentit saatiin toimimaan alustalla, mutta Bluetooth moduuli ei suostunut toimimaan ilman täydellistä kontaktia Floran väyliin. Kun laitteen kontaktipinnoille juotettiin johdot kiinni, saatiin Bluetoothkin toimimaan.

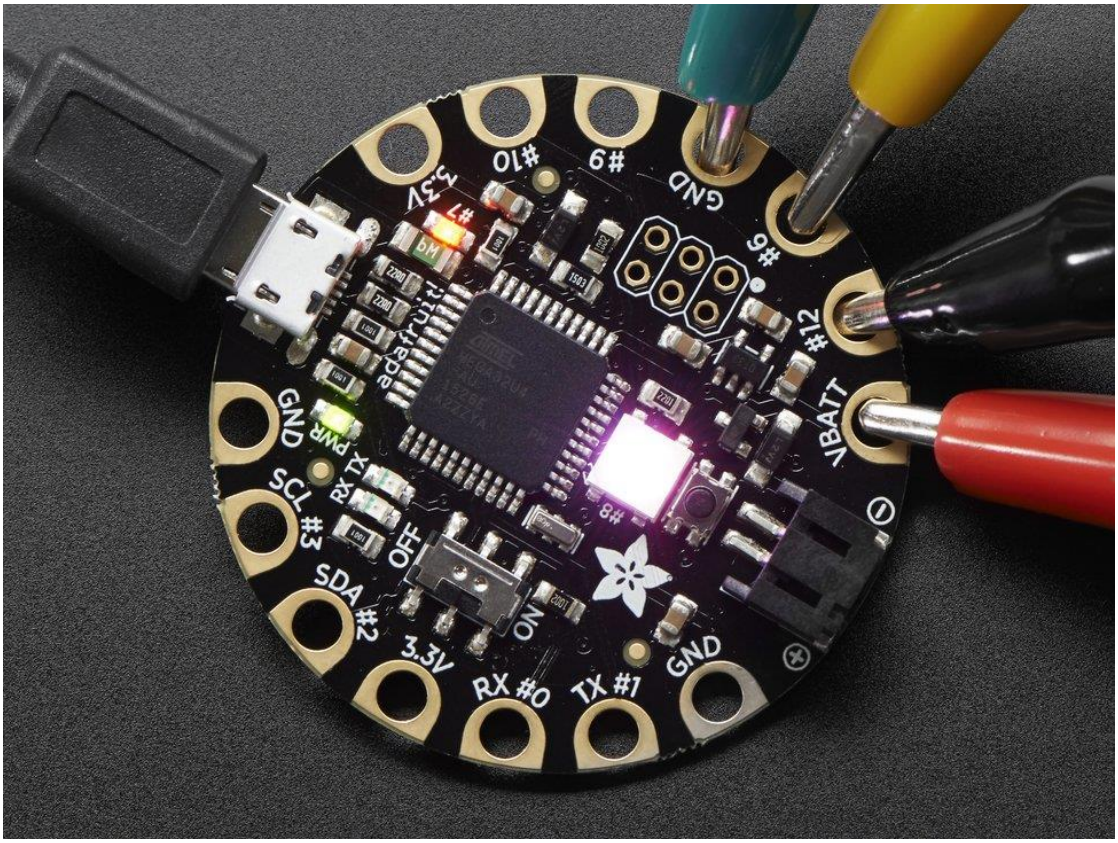
Toinen prototyyppi kasattiin pelkistä irrallisista komponenteista ja oikeanlaisten komponenttien etsiminen oli aikaa vievää. Komponentit pyrittiin valita Floran ja sen moduulien komponenttilistoista, mutta kaikkea samoja komponentteja ei ollut saatavilla. Irtokomponentit olivat halpoja, joten vääränlaisten komponenttien tilaamisessa ei ollut suurta rahallista riskiä.

4.1 Adafruit Flora

Flora on kehitetty päälle puettavien elektroniikkalaitteiden alustaksi (Kuva 2). Se on suunniteltu siten, että laite voidaan ommella kiinni vaatteisiin. Laite on halkaisijaltaan 1,75 tuumaa eli noin 44,5 millimetriä ja sillä on painoa 4,4 grammaa. Laitteessa on 14 kohtaa, josta se voidaan ommella kiinni ja kytkeä piireihin. Lisäksi siinä on kuusi juotettavaa paikkaa SPI-väylälle. (Description s.a.).

Prosessorina Florassa on Atmega32u4, joka perustuu 8 bittiseen AVR RISC -arkkitehtuuriin. Prosessori toimii 16 MHz kellotaajuudella ja siinä on 32 kilotavua ohjelmoitavaa muistia. Flora osaa ohjelmoida prosessoria suoraan USB-yhteyden kautta, joten se ei tarvitse erillistä ohjelmoijaa. Arduino-ympäristö

hoitaa kaiken lähellä rautaa olevan alustamisen ja ylläpidon, joten ohjelmointi on helppoa. Alustaa voidaan käyttää myös erillisellä akulla tai patterilla, jolloin sitä ei tarvitse erikseen kytkeä tietokoneeseen USB-johdolla. Florasta puuttuu kokonaan latauspiiri, joten akut tulee ladata ulkoisella laitteella.

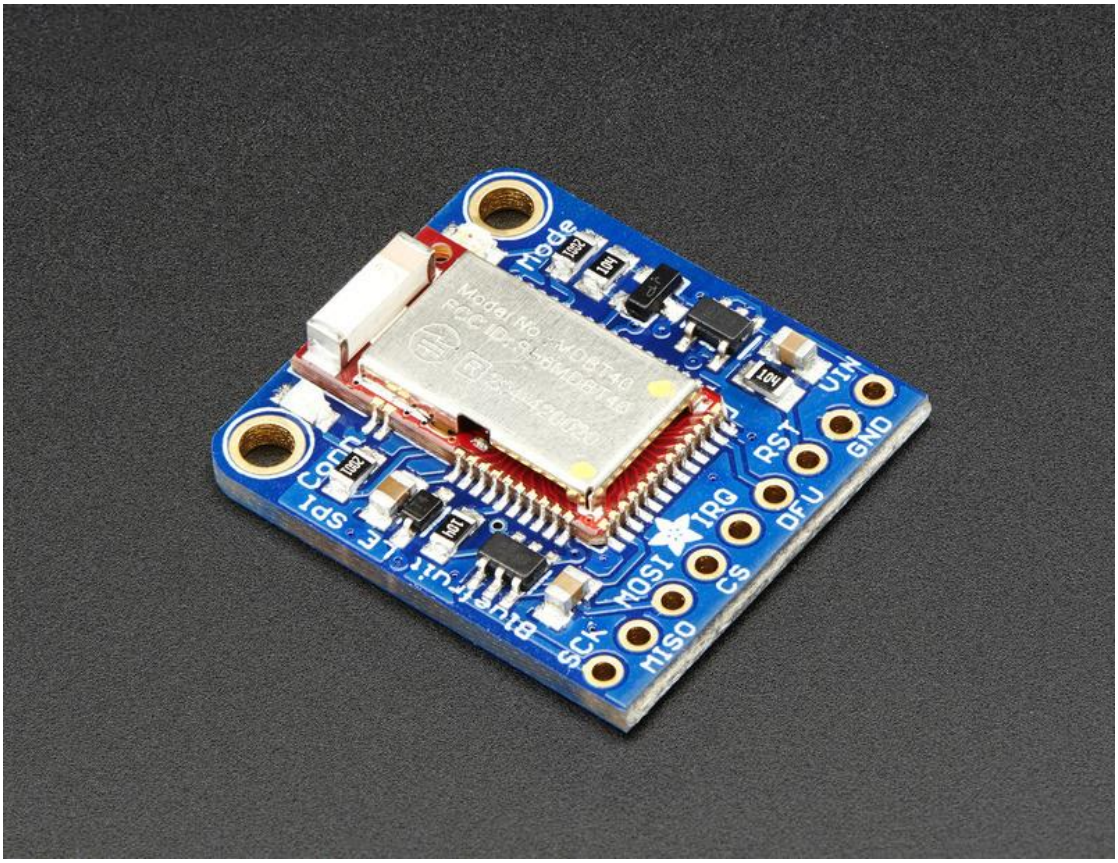


Kuva 2. Adafruit Flora

4.2 Bluetooth SoC

Florassa ei itsessään ole Bluetooth yhteyteen tarvittavia komponentteja, joten siihen tarvittiin erillinen moduuli yhteyden muodostamiseen. Floraan löytyy monia erilaisia Bluetooth-moduuleja, mutta koska projektiin haluttiin eritoten vähävirtaista ratkaisua, niin siihen valittiin Bluefruit LE SPI Friend -moduuli (Kuva 3). Tämä moduuli kommunikoi Floran kanssa SPI-yhteydellä, mikä tarkoittaa sitä, että se tarvitsee ainakin neljä vapaata signaalia pelkästään kommunikoimiseen. Tämän lisäksi moduuli tarvitsee vielä käyttöjännitteen ja maan, jotta virtapiiri muodostuisi ja laite toimisi. Vikatilanteiden varalta on myös hyvä kytkeä nollaussignaali moduuliin, jolloin moduuli voidaan käynnistää uudestaan tarvittaessa.

Bluetooth-moduulissa on nRF51822 SoC, jonka seuraaja nRF52832 SoC toimii toisen prototyypin prosessorina, jolloin laitteeseen ei tarvitse erillistä prosessoria ajamaan ohjelmaa, vaan ohjelman saa toimimaan samanaikaisesti Bluetoothin kanssa.



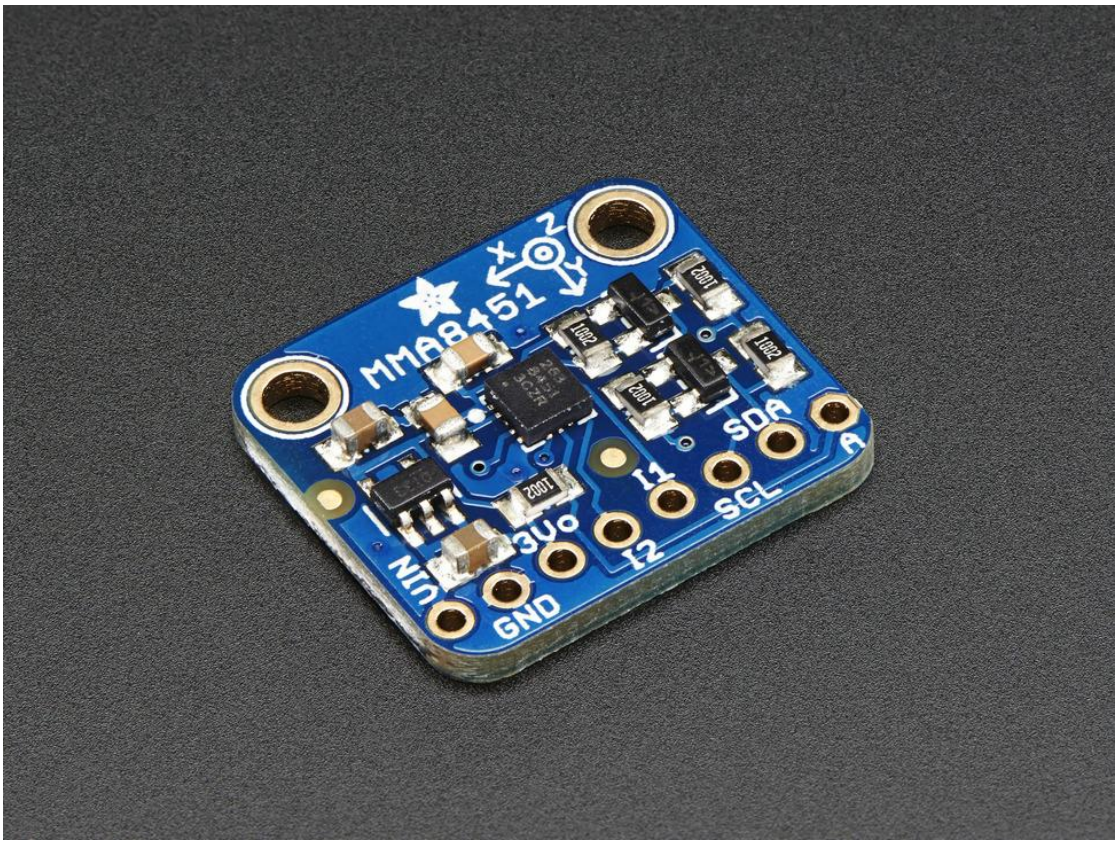
Kuva 3. Bluetooth LE SPI Friend

4.3 Kiihtyvyysanturit

Koska laite mittaa kiihtyvyyksiä, niin kiihtyvyysanturi on tärkeimpiä komponentteja laitteessa. Projektiin tarvittiin tarpeeksi tarkka laite, jotta mittaustulokset olisivat mahdollisimman yhdenmukaiset tutkimustulosten kanssa. Samalla pitää ottaa huomioon virrankulutus, sekä anturin tukemat väylät ja käyttöjännitteet, jolloin ei tarvita erillisiä piirejä muokkaamaan signaaleja kaikille komponenteille sopiviksi.

Ensimmäiseen prototyyppiin asennettiin Adafruit Triple-Axis -kiihtyvyysanturimoduuli (Kuva 4). Moduulissa on Freesaclen MMA8451-kiihtyvyysanturi. Kiih-

tyvyysanturi antaa 14-bittistä dataa ja toimii ±2g - ±8g alueella. Piiri kommunikoi I²C-väylällä. Adafruit on tehnyt moduuliin myös täydellisen ohjelmistokirjaston, joka sisältää esimerkkikoodit laitteen ohjelmoimiseen.



Kuva 4. Adafruit Triple-Axis Accelerometer

Toisen prototyypin kiihtyvyysanturiksi valittiin NPX:n valmistama FXLS8471Q. Laite valittiin prototyyppiin käyttöjännitteen, väylän, magnituditunnistuksen sekä tarkkuuden takia. FXLS8471Q ominaisuuksia ovat:

- käyttöjännite 1.95 V - 3.6 V
- valittava tarkkuusalue ±2 g, ±4 g ja ±8 g
- tukee tapahtumia
    - havaitsee vapaapudotuksen tai liikkeen
    - huomaa kiihtyvyyden magnitudimuutoksen
    - asentotunnistin
    - napautustunnistus
- tukee SPI-sekä I²C-väylää.

4.4 MicroSD-kortti

Standardi SD-kortille kehitettiin vuonna 1999 Sandiskin, Matsushita Electricin ja Toshiban toimesta. Tarkoituksena oli luoda parempi versio MultiMediaCards (MMC) standardista. SD-kortti valittiin projektiin siksi, että se on nopea, kestää

uudelleenkirjoittamista ja voidaan tarpeen vaatiessa vaihtaa ilman laitteen uudelleen juottamista.

SD-kortti tukee kahta väylää, erillistä SD-väylää sekä SPI-väylää. Koska SD-väylää käytetään oletuksena, tulee SD-kortti käskyttää käyttämään SPI-väylää. Ensin SD-kortille kerrotaan väylänopeus, lähettämälle sille kymmenen 0xFF-tavua peräkkäin. Tämän jälkeen tehdään nämä toimenpiteet:

- Ajetaan MISO-signaali ylös.
- Ajetaan CS-signaali alas.
- Lähetetään yksi 0xFF-tavu.
- Lähetetään seuraava 6 tavuinen komento keskeytyksettä.
    - Ensimmäinen tavu 0x40.
    - Seuraavat 4 tavua 0x00.
    - Viimeisenä CRC-tavu 0x95.
- Lähetetään tarvittava määrä 0xFF-tavua, jotta saadaan vastaus.
- Vastauksen pitäisi olla 0x01, tai muuten kortissa on jotakin vikaa.

NRF5-ohjelmistorajapinnassa on valmiit kirjastot SD-kortin käyttämiseen. Tämä helpottaa kehitystä, koska erillistä laiteajuria tai tiedostojärjestelmää ei tarvitse kirjoittaa laiteohjelmaan. Ohjelmistorajapinta käyttää FAT32-tiedostojärjestelmää.

4.5 nRF52832 SoC

Nordic Semiconductor on tällä hetkellä johtava vähävirtaisten Bluetooth SoC:ien kehittäjä (Investor Relations, 2019). Tästä syystä projektiin valittiin heidän kehittämä nRF52832. Koska nRF52832 ei sisällä valmista Bluetooth-antennia, niin laitteeseen valittiin valmis moduuli. Tällä tavoin antennin suunnittelua ja mittausta ei tarvinnut tehdä, vaan se jätettiin kolmannelle osapuolelle. Sopivan hintainen Raytac MDBT42Q-512KV2 löytyi Nordic Semiconductorin suosittelemista kolmannen osapuolten moduuleista. Moduulista löytyi kaksi eri versiota, joista toinen on PCB-levylle tehdyllä antennilla ja toinen erillisellä antennisirulla. Laitteeseen valittiin erillinen antennisiru, jolla pyritään maksimoimaan Bluetooth-signaalin laatu ja kantama.

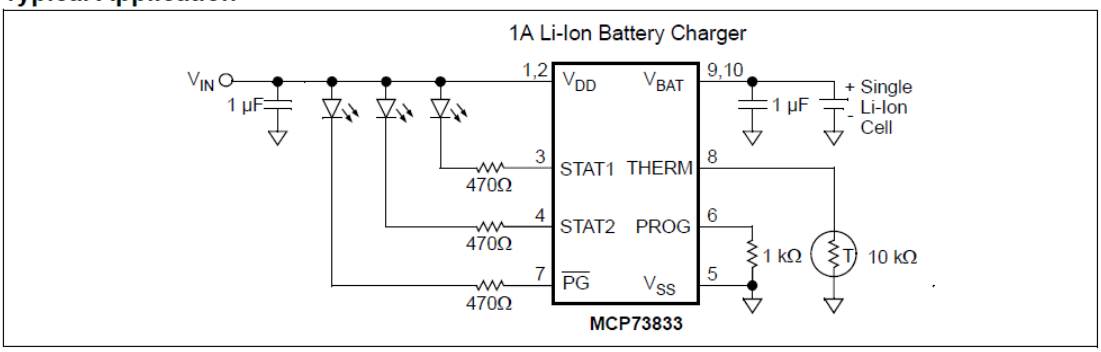
nRF52832piirissä on ARM Cortex-M4F -prosessori, joka toimii 64 MHz:n kellotaajuudella. Uudelleen kirjoitettavaa Flash-muistia on 512 kilotavua ja RAM-muistia 64 kilotavua. Tämä eroaa paljon nykyisistä tietokone prosessoreista, joissa on mahdollisesti yli 4:n GHz:n suoritinnopeus sekä monia gigatavuja

käyttömuistia. Siksi laitteen ohjelmoimisessa on tärkeää ottaa huomioon vähäiset resurssit. Tämän projektin tarkoituksiin, resursseja oli runsaasti. (nRF52832 s.a.)

4.6 Latauspiiri

Latauspiiri huolehtii, ettei LiPo-akkuja ladata liian kovalla virralla tai liikaa. LiPo-akut ovat todella herkkiä syttymään ylilatauksesta, joten on tärkeää, että erillinen piiri pitää huolen siitä, että lataus loppuu, kun akku on täynnä. Prototyyppiin latauspiiriksi valittiin Microchip Technologyn MCP73833, joka on lineaarinen LiPo-latauspiiri. Piiri pystyy syöttämään maksimissaan yhden ampeerin virtaa kennolle (MCP73833 2019). Projektissa virta on kumminkin asetettu sataan milliampeeriin, koska latausvirta tulee USB-linjasta. Tällöin ei ole myöskään ylikuumenemisen riskiä.

Typical Application



Kuva 5. MCP73833:n tyypillinen applikaatio

Tyypillistä applikaatiota (Kuva 5) muokattiin projektiin siten, että ledit jätettiin kokonaan pois piiristä. Ledeille tulevat signaalit sen sijaan menevät prosessorille. Tällöin voidaan ohjelmallisesti lukea ja kertoa latauksen tila (Kuva 6).

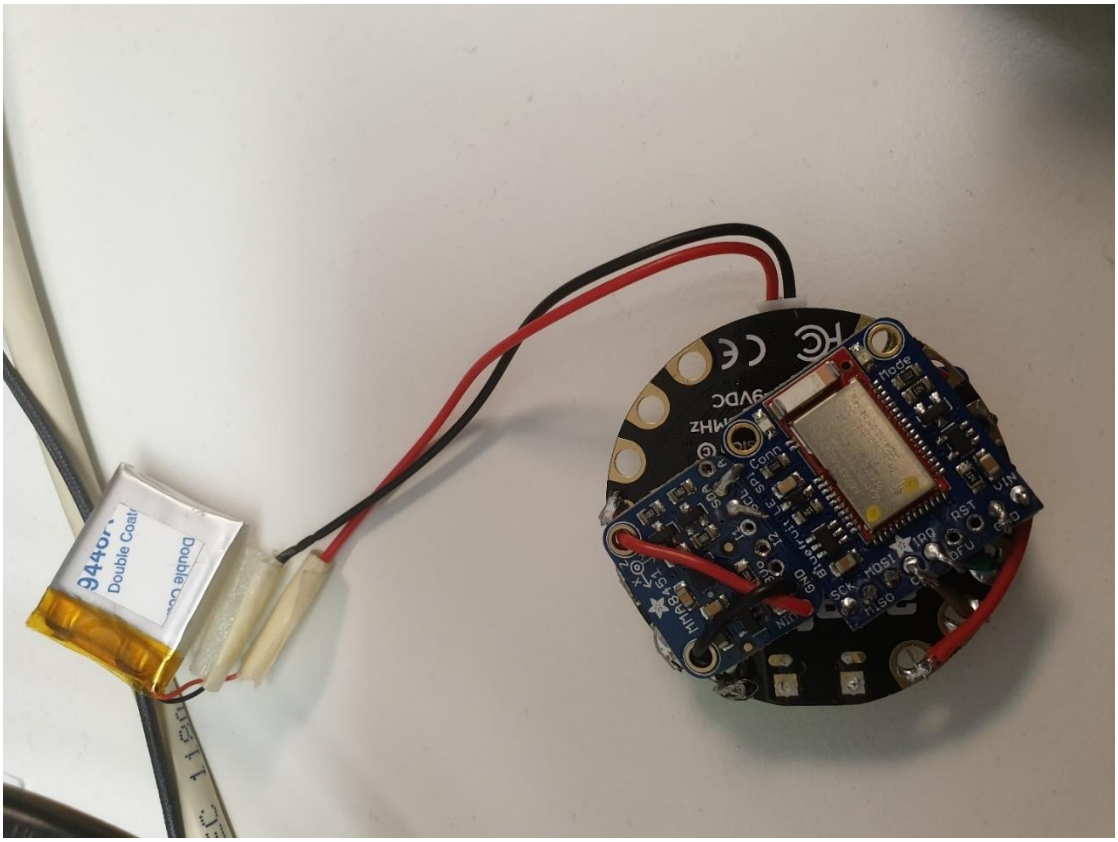
Charge Cycle State	STAT1	STAT2	PG
Shutdown	Hi-Z	Hi-Z	Hi-Z
Standby	Hi-Z	Hi-Z	L
Charge in Progress	L	Hi-Z	L
Charge Complete (EOC)	Hi-Z	L	L
Temperature Fault	Hi-Z	Hi-Z	L
Timer Fault	Hi-Z	Hi-Z	L
System Test Mode	L	L	L

Kuva 6. Latauksen tilat

5 ENSIMMÄINEN PROTOTYYPPI

Koska tieto elektroniikasta ja sen ohjelmoinnista oli lähtökohtaisesti vähäiset, niin paras tapa lähestyä ongelmaa on tutkia, miten muut ovat ratkaisseet laitekehityksen ongelmia. Tähän tarkoitukseen ensimmäinen prototyyppi rakennetaan Andafruit Flora -kehitysympäristöön, jolloin suuri hyppäys vieläkin laitteistoläheisempään ohjelmointiin helpottuisi mahdollisimman paljon. Ongelmaksi tässä muodostuu lopuksi se, että laitteessa on turhia komponentteja ja näin ollen se on paljon suurempi kuin mitä valmis laite voi olla. Tämä taas on ongelmallista siksi, että laite tulee olla mahdollisimman pieni. Pieni koko on välttämätöntä, koska laite on tarkoitus sijoittaa vyötärölle ja sitä kannetaan liikuntasuoritteiden aikana. Laite ei saisi haitata liikkeiden tekemistä. Myös virrankulutus on suurempi johtuen ylimääräisistä komponenteista.

Prototyyppi (Kuva 7) rakennetaan Adafruit Flora -alustalle (Kuva 2). Flora on pyöreän muotoinen kehitysalusta, joka on kehitetty pieniin päälle puettaviin applikaatioihin. Sen ohjelmointi on helppoa, koska siinä käytetään Arduino-kehitysympäristöä, joka on elektroniikka harrastajien suosiossa ja ympäristöön löytyy valmiita kirjastoja moniin erilaisiin tarkoituksiin. Adafruitilla on hyvin dokumentoidut tiedot myytävistä laitteista ja moduuleista. Heillä on myös aktiiviset keskustelualueet, jossa voi kysyä apua erilaisiin ongelmiin.



Kuva 7. Ensimmäinen prototyyppi kasattuna

6 LAITTEISTOLÄHEINEN OHJELMOINTI

Suljetun järjestelmän ohjelmointi eroaa käyttöjärjestelmissä, kuten Windows- tai Linux-ympäristössä ohjelmoimisesta. Nykyiset käyttöjärjestelmät osaavat sammuttaa ohjelman, joka käyttäytyy odottamattomasti ilman koko järjestelmän kaatumista. Tämä ominaisuus puuttuu suljetuista järjestelmistä, missä ei ole tarpeeksi resursseja ylläpitämään erillistä käyttöjärjestelmää. Siksi on erittäin tärkeää käsitellä kaikki virheet siten, että niistä jää tarpeeksi tietoa järjestelmään, jotta tiedettäisiin missä meni vikaan. Virheet eivät myöskään saa jumiuttaa laitetta pysyvästi, palautumattoman vakavan virheen sattuessa on laite käynnistettävä uudelleen.

Laitteistoa ohjelmoidaan C-kielellä, joka ei ole turvallisin ohjelmointikieli. Turvattomuus johtuu siitä, että C-kielessä voidaan muistia osoittaa vapaasti. Tästä johtuen väärään muistipaikkaan osoittavat osoittimet tai väärän kokoiset puskurit voivat aiheuttaa ylivuotoja, datan katoamista tai laitteen kaatumisen. Myös ajon aikainen muistin varaaminen voi aiheuttaa ongelmia. Tästä syystä on hyvä varata tarvittava muisti koodissa, jolloin kääntäjä osaa varoittaa ylimenevästä muistista.

6.1 Keskeytykset

Keskeytys keskeyttää ohjelman normaalin suorituksen ja hyppää keskeytykselle määriteltyyn keskeytyskäsittelijään ja palaa sen suoritettua takaisin paikkaan mistä oli lähtenyt. Keskeytyksillä voidaan automatisoida erilaisia tapahtumia, jotka pitää käsitellä heti tapahtumahetkellä. Laitteen ohjelmassa käytetään keskeytystä kiihtyvyysanturin ylitettyä haluttu rajaarvo ja tällöin voidaan asettaa tallennuslippu todeksi, jolloin ohjelman pääsilmukassa voidaan käsitellä tapahtuma ja tallentaa se MicroSD-kortille. Jos keskeytyksiä ei käytettäisiin, joutuisi ohjelma tarkkailemaan jatkuvasti sitä, onko kiihtyvyysanturissa uutta tietoa ja reagoida siihen. Tämä tekisi ohjelman tekemisestä paljon vaikeampaa sekä kuluttaisi enemmän resursseja. (Kothari ym. 2012, 183.)

Koska myös Softdevice toimii keskeytyksillä, on erittäin tärkeää, ettei keskeytyksissä tehdä mitään liian raskasta. Ongelma raskaissa keskeytyksissä on se, että jos ne vievät liikaa aikaa suorittaa, niin voi tulla päällekkäisiä keskeytyksiä, jolloin pahimmassa tapauksessa ohjelma kaatuu. Myös liian pitkät viiveet voivat haitata keskeytyksien toimimista, joten on parempi pilkkoa pitkät viiveet pienemmiksi silmukoiden avulla.

6.2 Bluetooth Low Energy

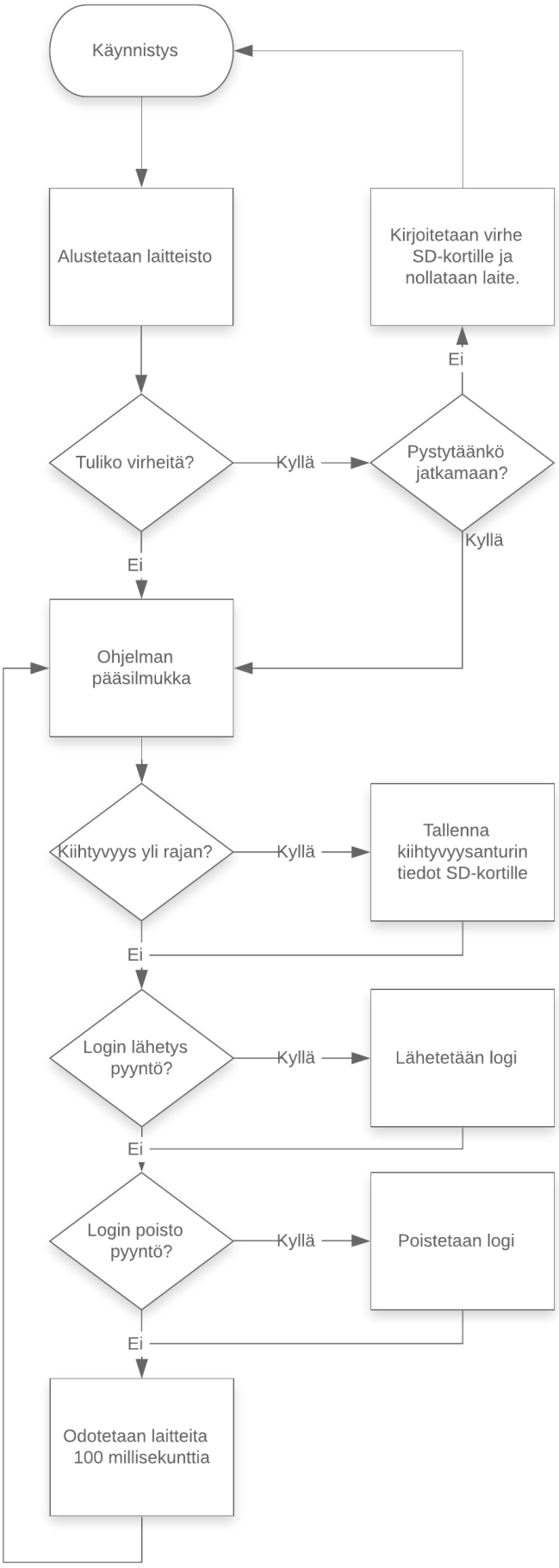
Bluetooth on maailmanlaajuinen standardi lyhyen kantaman, vähäisen virrankulutuksen, edullisen sekä pienimuotoisen langattoman teknologian käyttöön. Bluetoothin avulla laitteet voivat kommunikoida keskenään radiolinkkien kautta ja se on kehitetty korvaamaan sarjaliikennekaapelit eri laitteiden välillä. Nykyään sitä käytetään melkein kaikissa elektroniikka laitteissa. Käyttökohteita Bluetooth sovelluksille ovat muun muassa tiedostojen siirto laitteelta toiselle, musiikin kuuntelu langattomasti ja tiedonvaihto laitteiden välillä. Bluetooth Low Energy on lisätty Bluetooth-standardiin versiosta 4.0 alkaen. Nimensä mukaan Bluetooth Low Energy on suunniteltu siten, että se kuluttaa mahdollisimman vähäisen määrän virtaa. (Gupta 2013, 6.)

Toinen prototyyppi käyttää Nordic Semiconductorin NUS-palvelua, joka on UART-silta Bluetoothin yli. Tällöin laitteen kanssa voidaan kommunikoida, kuten tavallisellakin UART-väylällä.

6.3 Laiteohjelma

Ohjelma on rakennettu Nordic Semiconductorin tekemän mallin päälle. Tässä mallissa on kaikki perustoimenpiteet tehty valmiiksi, jolloin kehittäjä voi keskittyä ohjelmoimaan itse laitetta. Kirjastojen sisällyttäminen tähän projektiin osoittautui vaikeaksi. Missään ohjeissa ei mainittu tiedostoja, jotka erillinen kirjasto käyttää. Projektiin luotiin oma apuohjelma, jolla pystyi esimerkkikoodista purkamaan tarvittavien tiedostojen nimet ja sitten tiedostot lisättiin projektiin. Toinen ongelmallinen kohta oli asetustiedosto, jolla erillisten kirjastojen asetukset määriteltiin. Tässä asetustiedostossa oli yli 12 tuhatta riviä asetuksia, joista pitäisi löytää kaikki tarpeellinen kirjaston ominaisuuksien aktivoimiseksi. Myös tähän tarkoitukseen rakennettiin oma ohjelma etsimään eroavaisuuksia esimerkkikoodin ja oman projektin asetustiedostojen välillä.

NRF5 SDK on rakennettu moduulisesti siten, ettei mitään turhaa tarvitsisi kääntää mukaan ohjelmaan. Tämä on oikea tapa pienentää ohjelman kokoa, mutta samalla se luo jyrkän oppimiskäyrän ohjelmistorajapinnan oppimiseen. Varsinainen ohjelmakehitys alkoi, kuin kaikki tarvittavat tiedostot oli saatu käännettyä projektiin mukaan. Ohjelman rakenne on yksinkertainen (Kuva 8), koska keskeytykset hoitavat suurimman osan toiminnoista.



Kuva 8. Ohjelman vuokaavio yksinkertaistettuna

6.4 Laiteohjelman tärkeimmät funktiot

Laitteeseen tallennetut tiedot lähetetään Bluetoothilla mobiilipäätteelle käyttämällä sendLog-funktiota (Kuva 9). Jokaisen lähetettävän puskurin ensimmäinen alkio on V-kirjain, josta vastaanottava laite pystyy päättelemään paketin sisällön. Silmukka alkaa riviltä 300, jossa ensin luetaan rivi lokitiedostosta. Tämän jälkeen, jos dataa saatiin luettua, kopioidaan luettu data lähetyspuskuriin alkaen alkiosta 1. Rivillä 305 lisätään V-tavun vievä tila luettuihin, jotta koko paketti lähtisi oikein. Rivillä 311 lähetään tietopaketti ja rivillä 316 odotetaan, että lähetys on valmis ennen kuin jatketaan ohjelmaa.

```
285  void sendLog()
286  {
287    static uint8_t readBuf[20];
288    FRESULT ff_result;
289    uint16_t bytesRead;
290    f_close(&file);
291    uint32_t time;
292    ret_code_t err_code = 0;
293
294
295    ff_result = f_open(&file, FILE_NAME, FA_READ);
296    readBuf[0] = 'V';
297
298    do
299    {
300      ff_result = f_read(&file, &readBuf[1], 12, (UINT *)&bytesRead);
301      if ( bytesRead > 0 )
302      {
303
304        memcpy((void *)&time, (void *)&readBuf[1], 4);
305        bytesRead++;
306
307        transferActive = true;
308  #ifdef DEBUG
309        extractDataFromBuffer(&data, &readBuf[6]);
310  #endif
311        err_code = ble_nus_data_send(&m_nus,
312                                     readBuf,
313                                     &bytesRead,
314                                     m_conn_handle);
315
316        while(transferActive && deviceConnected) __WFE();
317
318  #ifdef DEBUG
319        printf("Sending: %ld [ ", time);
320        for ( unsigned int c = 0; c < 7; c++ )
321        {
322          printf("%02X ", readBuf[5 + c]);
323        }
324        printf("] (%d) -> %d ", readBuf[12], err_code);
325        printf("[ %d %d %d ]\n", data.x, data.y, data.z);
326      }
327  #endif
328
329  //   bytesRead = 0; // NOTE: For testing one log line / call.
330
331    } while( !ff_result && bytesRead && deviceConnected );
332
333    f_close(&file);
334    ff_result = f_open(&file, FILE_NAME, FA_READ | FA_WRITE | FA_OPEN_APPEND);
335    printf("Done!\n");
336    sendLogRequest = false;
337
338  }
```

Kuva 9. sendLog-funktio

Kun laite vastaanottaa tietoa, niin kutsutaan nus_data_handler-käsittelijäfunktiota (Kuva 10). Tässä funktiossa tarkistetaan ja käsitellään tuleva data. Laite käsittelee pelkät komennot, joiden ensimmäinen tavu vastaa ascii-merkistön kysymysmerkkiä. Tämän jälkeen katsotaan mikä toinen tavu on, ja jos se vastaa komennoille asetettuja tavuja, käsitellään loput paketista komennon argumentteina. Komentomerkkeinä käytetään näitä:

- P lähettää tallennetut tiedot päätelaitteelle
- T asettaa laitteeseen uuden ajan
- D tyhjentää lokitiedoston

```
348  static void nus_data_handler(ble_nus_evt_t * p_evt)
349  {
350      if (p_evt->type == BLE_NUS_EVT_RX_DATA)
351      {
352          uint32_t err_code;
353          const uint8_t *character = p_evt->params.rx_data.p_data;
354          if ( *character++ == '?' )
355          {
356            printf("Command character\n");
357            NRF_LOG_INFO("Got command character > %c", *character);
358            if (*character == 'P')// [P]ush Log
359            {
360                sendLogRequest = true;
361            }
362
363            if (*character == 'T' )
364            {
365              printf("New time!\n");
366              uint32_t value = *(const uint32_t *)(character + 1);
367              myTimerSet(value);
368              printf("New Time set: %ld -> %s\n", value, ctime((time_t *)(&my_timer_seconds)));
369
370            }
371
372            if(*character == 'D')
373            {
374              deleteLogRequest = true;
375            }
376          }
377      }
378      else if ( p_evt->type == BLE_NUS_EVT_TX_RDY )
379      {
380        transferActive = false;
381      }
382  }
```

Kuva 10. nus_data_handler-funktio

Tiedot tallennetaan levylle, kun kiihtyvyysanturi antaa keskeytyksen, joka ilmoittaa sen, että liikkeen magnitudi on yli asetetun rajan. Tällöin savaAccelerometerDataToDisk-funktio (Kuva 11) ottaa arvon kiihtyvyysanturilta ja kirjoittaa sen levylle. Tallennus tehdään siten, että ensin kirjoitetaan tiedostoon aikaleima, jolloin data on tallennettu ja sitten kirjoitetaan kiihtyvyysanturin tiedot.

Viimeiseksi merkiksi tallennetaan rivinvaihto, jolloin pystytään hyppäämään korruptoituneen kohdan yli.

```
1071  static void saveAccelerometerDataToDisk()
1072  {
1073    uint8_t bytes_written;
1074      FRESULT ff_result;
1075      printf("Writing to file!\r\n");
1076
1077    // Read acceleration data to struct
1078    readAccData(&m_twi, 0x1E, accValues);
1079
1080    // Write buffer to file
1081    writeBufferToFile((char *)(&my_timer_seconds), sizeof(my_timer_seconds));
1082    writeBufferToFile(accValues, 7);
1083    accValues[0] = '\n';
1084    writeBufferToFile(accValues, 1);
1085
1086    f_sync(&file);
1087
1088    // Clear interupt
1089    clearInterrupt(&m_twi, 0x1E);
1090    shouldSaveToFile = false;
1091  }
```

Kuva 11. saveAccelerometerDataToDisk-funktio

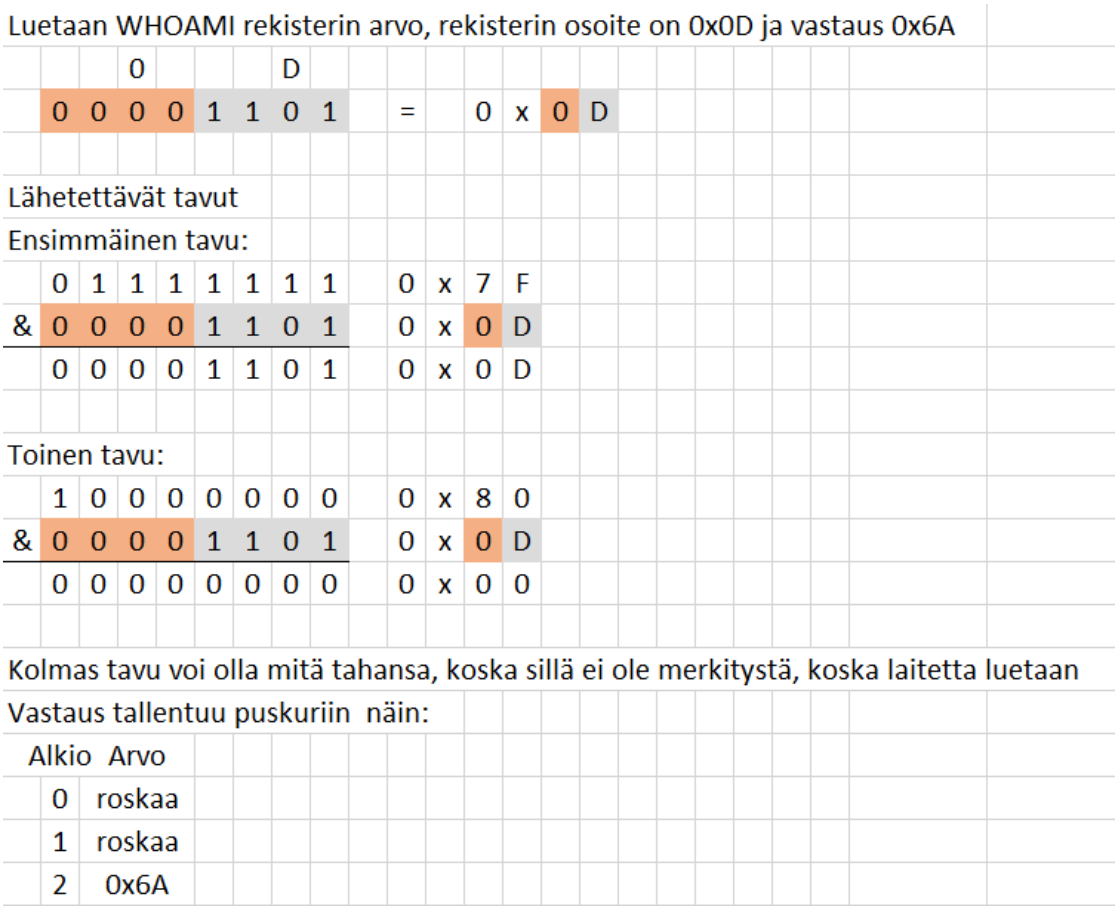
6.5 FXLS8471Q-laiteajuri

Projektissa käytetään NPX:n valmistamaa FXLS8471Q-kiihtyvyysanturia. Anturille ei ollut erillistä kirjastoa nRF5 SDK:ssa, joten ajuri ohjelmoitiin alusta asti. Ajurin rakentaminen aloitetaan selvittämällä kaikki tarpeelliset tiedot laitteen teknisestä ohjekirjasta. Ohjekirjan piireille saa yleensä ladattua suoraan valmistajan sivuilta. Ensimmäiseksi on hyvä kirjata ylös kaikki rekisterit ja niiden muistipaikat, jotta ohjelmassa olisi helpompi käsitellä rekistereitä nimellä, eikä tarvitse muistaa jokaisen rekisterin osoitetta erikseen. Rekistereiden nimet ja osoitteet tallennetaan erilliseen tiedostoon (Kuva 12), joka sitten sisällytetään mukaan ajurin lähdekoodiin.

```
//FXLS8471Q Registers

#define STATUS_REG              0x00    // STATUS Register

#define OUT_X_MSB_REG           0x01    // [7:0] are 8 MSBs of the 14-bit X-axis sample
#define OUT_X_LSB_REG           0x02    // [7:2] are the 6 LSB of 14-bit X-axis sample
#define OUT_Y_MSB_REG           0x03    // [7:0] are 8 MSBs of the 14-bit Y-axis sample
#define OUT_Y_LSB_REG           0x04    // [7:2] are the 6 LSB of 14-bit Y-axis sample
#define OUT_Z_MSB_REG           0x05    // [7:0] are 8 MSBs of the 14-bit Z-axis sample
#define OUT_Z_LSB_REG           0x06    // [7:2] are the 6 LSB of 14-bit Z-axis sample

#define F_SETUP_REG             0x09      // F_SETUP FIFO Setup Register
#define TRIG_CFG_REG            0x0A      // TRIG_CFG Map of FIFO data capture events
#define SYSMOD_REG              0x0B      // SYSMOD System Mode Register
#define INT_SOURCE_REG          0x0C      // INT_SOURCE System Interrupt Status Register
#define WHO_AM_I_REG            0x0D      // WHO_AM_I Device ID Register
#define XYZ_DATA_CFG_REG        0x0E      // XYZ_DATA_CFG Sensor Data Configuration Register
#define HP_FILTER_CUTOFF_REG    0x0F      // HP_FILTER_CUTOFF High Pass Filter Register

#define PL_STATUS_REG           0x10      // PL_STATUS Portrait/Landscape Status Register
#define PL_CFG_REG              0x11      // PL_CFG Portrait/Landscape Configuration Register
#define PL_COUNT_REG            0x12      // PL_COUNT Portrait/Landscape Debounce Register
#define PL_BF_ZCOMP_REG         0x13      // PL_BF_ZCOMP Back/Front and Z Compensation Register
#define P_L_THS_REG             0x14      // P_L_THS Portrait to Landscape Threshold Register

#define FF_MT_CFG_REG           0x15      // FF_MT_CFG Freefall and Motion Configuration Register
#define FF_MT_SRC_REG           0x16      // FF_MT_SRC Freefall and Motion Source Register
#define FT_MT_THS_REG           0x17      // FF_MT_THS Freefall and Motion Threshold Register
#define FF_MT_COUNT_REG         0x18      // FF_MT_COUNT Freefall Motion Count Register

#define TRANSIENT_CFG_REG       0x1D      // TRANSIENT_CFG Transient Configuration Register
#define TRANSIENT_SRC_REG       0x1E      // TRANSIENT_SRC Transient Source Register
#define TRANSIENT_THS_REG       0x1F      // TRANSIENT_THS Transient Threshold Register
#define TRANSIENT_COUNT_REG     0x20      // TRANSIENT_COUNT Transient Debounce Counter Register

#define PULSE_CFG_REG           0x21      // PULSE_CFG Pulse Configuration Register
#define PULSE_SRC_REG           0x22      // PULSE_SRC Pulse Source Register
#define PULSE_THSX_REG          0x23      // PULSE_THS XYZ Pulse Threshold Registers
#define PULSE_THSY_REG          0x24
#define PULSE_THSZ_REG          0x25
#define PULSE_TMLT_REG          0x26      // PULSE_TMLT Pulse Time Window Register
#define PULSE_LTCY_REG          0x27      // PULSE_LTCY Pulse Latency Timer Register
#define PULSE_WIND_REG          0x28      // PULSE_WIND Second Pulse Time Window Register

#define ASLP_COUNT_REG          0x29      // ASLP_COUNT Auto Sleep Inactivity Timer Register

#define CTRL_REG1               0x2A      // CTRL_REG1 System Control 1 Register
#define CTRL_REG2               0x2B      // CTRL_REG2 System Control 2 Register
#define CTRL_REG3               0x2C      // CTRL_REG3 Interrupt Control Register
#define CTRL_REG4               0x2D      // CTRL_REG4 Interrupt Enable Register
#define CTRL_REG5               0x2E      // CTRL_REG5 Interrupt Configuration Register

#define OFF_X_REG               0x2F      // XYZ Offset Correction Registers
#define OFF_Y_REG               0x30
#define OFF_Z_REG               0x31

#define A_VECM_CFG_REG          0x5F    // Acceleration vector-magnitude configuration register
#define A_VECM_THS_MSB_REG      0x60    // Acceleration vector-magnitude threshold MSB
#define A_VECM_THS_LSB_REG      0x61    // Acceleration vector-magnitude threshold LSB
#define A_VECM_CNT_REG          0x62    // Acceleration vector-magnitude debounce count

#define A_VECM_INITX_MSB_REG    0x63    // Acceleration vector-magnitude X-axis reference value MSB
#define A_VECM_INITX_LSB_REG    0x64    // Acceleration vector-magnitude X-axis reference value LSB
#define A_VECM_INITY_MSB_REG    0x65    // Acceleration vector-magnitude Y-axis reference value MSB
#define A_VECM_INITY_LSB_REG    0x66    // Acceleration vector-magnitude Y-axis reference value LSB
#define A_VECM_INITZ_MSB_REG    0x67    // Acceleration vector-magnitude Z-axis reference value MSB
#define A_VECM_INITZ_LSB_REG    0x68    // Acceleration vector-magnitude Z-axis reference value LSB

#define A_FFMT_THSX_MSB_REG   0x73    // X-axis FFMT threshold MSB
#define A_FFMT_THSX_LSB_REG   0x74    // X-axis FFMT threshold LSB
#define A_FFMT_THSY_MSB_REG   0x75    // Y-axis FFMT threshold MSB
#define A_FFMT_THSY_LSB_REG   0x76    // Y-axis FFMT threshold LSB
#define A_FFMT_THSZ_MSB_REG   0x77    // Z-axis FFMT threshold MSB
#define A_FFMT_THSZ_LSB_REG   0x78    // Z-axis FFMT threshold LSB
```

Kuva 12. Kiihtyvyysanturin rekisterit

Seuraavaksi rakennetaan funktiot, joilla voidaan lähettää ja vastaanottaa tietoa anturilta. Teknisestä ohjekirjasta tulkiten anturille lähetetään 3 tavua, joista ensimmäisen tavun eniten merkitsevään bittiin asetetaan luku/kirjoitusarvo. Kun bitti on 0, niin laitteeseen kirjoitetaan ja sen ollessa 1 laitetta luetaan. Seuraavat 7 bittiä ovat rekisterin osoitteen 7 vähiten merkitsevää bittiä ja seuraavan tavun eniten merkitsevä bitti on 8-bittisen rekisterin eniten merkitsevä bitti. Muut bitit tässä tavussa eivät tee mitään. Kolmas tavu kirjoituksessa kertoo arvon, joka halutaan rekisteriin asetettavan. Jos laitetta luetaan, niin laitteeseen lähetetään niin monta tavua tietoa, kuin luettavia tavuja on. Lähetettävien tavujen sisällöllä ei ole merkitystä, niiden tarkoitus on ainoastaan kellottaa takaisin tulevat tavut. Esimerkissä (Kuva 13) näytetään, miten tavut dekoodataan lukemista varten.



Kuva 13. WHOAMI-rekisterin lukeminen

```
102  uint8_t accReadByte(uint8_t RegisterAddress)
103  {
104    const uint8_t byteOffset = 2;
105
106    tx_buf[0] = 0x7F & RegisterAddress;
107    tx_buf[1] = 0x80 & RegisterAddress;
108    tx_buf[2] = 0x00;
109
110    APP_ERROR_CHECK(nrf_drv_spi_transfer(&spi, tx_buf, 2, rx_buf, 1+byteOffset));
111    return rx_buf[byteOffset];
112  }
```

Kuva 14. Lukufunktio

```
114  void accReadMultipleBytes(int8_t RegisterAddress, uint8_t *buffer, uint8_t bytesToRead)
115  {
116    const uint8_t byteOffset = 2;
117
118    tx_buf[0] = 0x7F & RegisterAddress;
119    tx_buf[1] = 0x80 & RegisterAddress;
120    tx_buf[2] = 0x00;
121
122    APP_ERROR_CHECK(nrf_drv_spi_transfer(&spi, tx_buf, 2, rx_buf, bytesToRead+byteOffset));
123
124    memcpy(buffer, rx_buf+byteOffset, bytesToRead);
125  }
```

Kuva 15. Monen tavun lukemiseen tehty funktio

Lukufunktiossa (Kuva 14) asetetaan rivillä 106, tavun kaikki muut bitit ykkösiksi paitsi eniten merkitsevä bitti, jolloin binäärisen JA-operaation jälkeen tavuun tallentuu ainoastaan osoitteen 7 vähiten merkitsevää bittiä. Seuraavalla rivillä toisen tavun merkitsevin bitti asetetaan ykköseksi ja muut nolliksi, tällöin binäärisen JA-operaation jälkeen rekisteri osoitteesta tallentuu ainoastaan merkitsevin bitti. Muut bitit tässä tavussa ovat merkityksettömiä, koska osoite koodataan ensimmäisen tavun 7:stä vähiten merkitsevästä bitistä ja seuraavan tavun merkitsevimmästä bitistä. Koska funktio lukee vain yhden tavun annetusta rekisteristä, niin kolmas tavu lähetetään anturille. Tällöin samaan aikaan kun kolmas tavu lähetetään anturille, saadaan tieto laitteen rekisterin arvosta. Koska anturilta saadaan vastaus jokaisesta kirjoitetusta bitistä, niin rekisterissä oleva arvo on vasta vastaanoton kolmannessa alkiossa, siksi on tärkeää palauttaa vain kolmas alkio puskurista. Monen perättäisen rekisterin lukeminen on myös mahdollista (Kuva 15), tällöin lähetetään kahden ensimmäisen tavun jälkeen niin monta tavua kuin halutaan tietoa luettavan. Tämä on mahdollista siksi, että anturi osaa jokaisen lukukerran jälkeen kasvattaa rekisterin osoitetta yhdellä.

```
93   void accWriteByte(uint8_t RegisterAddress, uint8_t data)
94   {
95     tx_buf[0] = 0x80 | RegisterAddress;
96     tx_buf[1] = 0x80 & RegisterAddress;
97     tx_buf[2] = data;
98
99     APP_ERROR_CHECK(nrf_drv_spi_transfer(&spi, tx_buf, 3, NULL, 0));
100  }
```

Kuva 16. Kirjoitusfunktio

Kirjoitusfunktion (Kuva 16) rivillä 95 käytetään binääristä TAI-operaatiota, koska merkitsevin bitti halutaan pitää ykkösenä. Seuraavalla rivi säilyy muuttumattomana lukufunktiosta. Rivillä 97 kirjoitetaan lähetyspuskurin kolmanteen alkioon arvo, joka halutaan kirjoittaa anturin rekisterin osoitteeseen.

```
127  void accReadValues(ACCDATA *data)
128  {
129    uint8_t buf[6];
130    accReadMultipleBytes(OUT_X_MSB_REG, buf, 6);
131
132    data->x = (((buf[0] << 8) | buf[1]) >> 2);
133    data->y = (((buf[2] << 8) | buf[3]) >> 2);
134    data->z = (((buf[4] << 8) | buf[5]) >> 2);
135
136    // Changing 14 bit values to 16 bits
137    data->x = (int16_t)(data->x << 2) / 4;
138    data->y = (int16_t)(data->y << 2) / 4;
139    data->z = (int16_t)(data->z << 2) / 4;
140  }
```

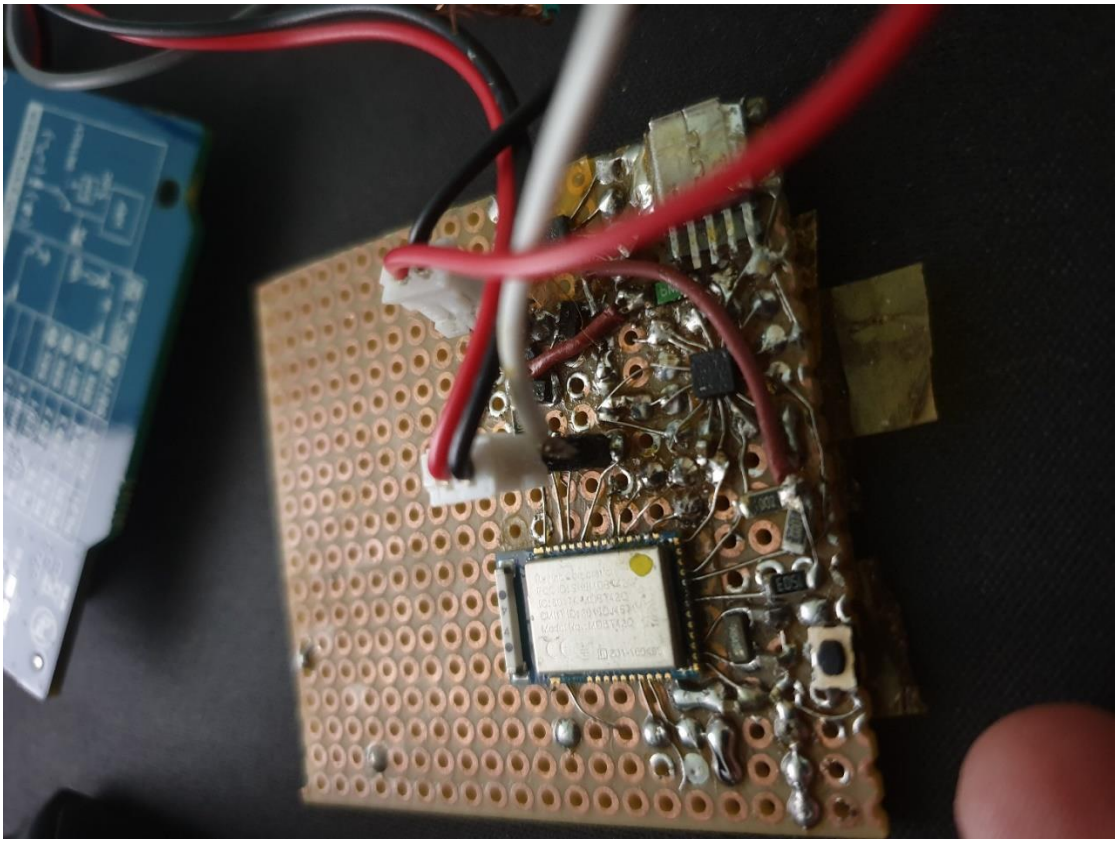
Kuva 17. Kiihtyvyystietojen lukufunktio

Kiihtyvyysanturitiedon lukemisessa pitää muistaa, että luetut arvot ovat 14-bittisiä, joten ne tulee muuntaa 16-bittisiksi. Tämä muuntaminen tehdään accReadValues-funktiossa (Kuva 17). Rivillä 130 haetaan OUT_X_MSB-, OUT_X_LSB-, OUT_Y_MSB-, OUT_Y_LSB-, OUT_Z_MSB- ja OUT_Z_LSB-rekisterin tiedot puskuriin. Riveillä 132 - 134 yhdistetään kahdesta 8-bittisestä arvosta 16-bittinen luku, mutta luku on vielä 14-bittinen kahden komplementti, jolloin negatiiviset arvot eivät suoraan toimi. Siksi rivillä 137 siirretään bittejä kaksi vasemmalle, jolloin saadaan etumerkistä vastaava bitti merkitsevimmäksi. Tämän jälkeen luku jaetaan neljällä, jolloin magnitudi säilyy samana, mutta etumerkki pysyy paikallaan. Sama toistetaan y- ja z-akseleille riveillä 138 ja 139.

7 TOINEN PROTOTYYPPI

Toinen prototyyppi kasattiin alusta asti, omista komponenteista yhdelle reikälevylle. Komponentit valittiin osittain samoiksi mitä, ensimmäisessäkin prototyypeissä käytettiin. Kaikkia samoja komponentteja ei ollut suoraan saatavilla, jolloin projektiin valittiin mahdollisimman samoilla ominaisuuksilla olevia komponentteja.

Laitteen suunnittelu aloitettiin KiCad EDA -ohjelmalla, joka on monen alustan avoimeen lähdekoodiin perustuva cad-ohjelmisto. Ohjelma on tehty elektroniikka laitteiden suunnitteluun, joten siitä löytyy erillinen työkalu piirien tekemiseen. Laitteen piirustukset oli nopea tehdä tällä ohjelmalla eikä ohjelman oppimiseen tarvinnut käyttää paljon aikaa. Myös komponenttien malleja oli paljon saatavilla, joten ei ollut tarvetta rakentaa omia. Vaikka ohjelmassa oli mahdollisuus rakentaa suoraan piirilevy, niin omatekoisten piirilevyjen luonti työympäristössä olisi ollut liian työlästä. Ohjelmalla sai aikaan hyvän piirustuksen, joka auttoi reikälevylle tehtävän prototyypin kasaamisessa.



Kuva 18. Toinen prototyyppi kasattuna

7.1 Juottaminen

Komponenttien liittäminen toisiinsa vaati juottamista ja koska komponentit ovat pieniä ja reikälevyssä on suhteessa suuret kontaktipinnat, niin komponenttien ja reikälevyn väliin laitettiin kuuma kestävää teippiä. Juottaminen oli projektin aikaa vievin osa johtuen komponenttien pienuudesta. Hyppylankoina projektissa käytettiin paksumman sähköjohdon sisältämiä yksittäisiä metallisäikeitä. Näiden ohuiden lankojen juottaminen pintaliitoskomponenttien kontaktipinnoille tuotti alussa paljon vaikeuksia. Juotostekniikan parannuttua työ helpottui merkittävästi.

Juottamisessa työvälineinä käytetään juotinta, pinsettejä, juotostelinettä, juotospastaa sekä juotostinaa. Juottaminen tulee tehdä hyvin tuuletetussa ympäristössä, etteivät haitalliset höyryt kulkeudu keuhkoihin. Komponenttien lämmönkestävyys on myös otettava huomioon. Liian kuumaksi lämmitetty komponentti voi lopettaa toimimisen. Myöskään liian kylmällä juottimella ei kannata yrittää liittää komponentteja piiriin. Tällöin kylmäjuotoksen mahdollisuus on suuri, jolloin piiri ei välttämättä toimi oikein. On myös mahdollista, että kylmäjuotos hajoaa laitetta käsitellessä, jolloin kontakti häviää kokonaan.

Tapahtumia käyttämällä laite voidaan asettaa tekemään IO-keskeytys sellaisen tapahduttua, joten ohjelmiston ei tarvitse jatkuvasti lukea anturia, vaan voi tehdä tämän vain silloin, kuin tietty tapahtuma on aktiivinen. Tämän tyyppinen ratkaisu säästää virtaa.

8 JOHTOPÄÄTÖS JA JATKOKEHITYS

Projekti oli hyvin työläs, koska se sisälsi paljon uutta asiaa. Projektia hidasti myös se, ettei yrityksessä ei ollut valmiiksi mitään elektroniikan kehittämiseen tarvittavia työkaluja, vaan kolvit ja mittauslaitteet tuli tilata ennen projektin aloittamista. Osa työkaluista tilattiin projektin edetessä, jolloin oli mahdollisimman hyvä kuva siitä, mitä työkaluja projektin toteutus vaatii. Projekti eteni suhteellisen nopeassa aikataulussa ja antoi paljon tietoa siitä, miten elektroniikka laitteiden kehittäminen kannattaa toteuttaa. Projekti onnistui hyvin, vaikka läh-

tökohdat tämän kaltaisen projektin työstämiseen olivat heikot. Projektiin käytetty työmäärä oli valtava, mutta aihealueen kiinnostus auttoi vähentämään tehtyjen tuntien raskautta.

Projektin toteuttamiseen vaadittu tieto haettiin pääsäännöllisesti internetsivuilta ja Youtube-videoista. Myös keskustelualustat kuten Stack Overflow ja Nordic Semiconductorin foorumit antoivat tietoa ongelmien ratkaisussa.

Kiihtyvyysantureille on monia eri käyttötapoja. Niitä käytetään nykyjään melkein kaikessa elektroniikassa. Erillistä kiihtyvyysanturia voidaan käyttää peliohjaimena, joka voidaan kiinnittää jalkaan ja ohjelmoida erilaisia liikkeitä, kuten esimerkiksi tekemään pelissä tarvittavia ohjainkomentoja. Tämänlainen toiminnallisuus auttaisi esimerkiksi kätensä vammauttaneita pelaajia, joiden on vaikea pelata normaalilla peliohjaimella. Jalkaa nostaessa nopeasti ylös, voitaisiin toteuttaa esimerkiksi pelissä hahmon hyppääminen. Mahdollisuuksia on monia ja nykyiset VR-laitteet käyttävät kiihtyvyysantureita pään asennon tunnistamiseen.

LÄHTEET

About KiCad. s.a. KiCad. WWW-dokumentti. Saatavissa: http://kicad-pcb.org/about/kicad/ [viitattu 09.05.2019]

Description, s.a. Adafruit. WWW-dokumentti. Saatavissa: https://www.adafruit.com/product/659 [viitattu 27.05.2019]

Gupta, N. 2013. Inside Bluetooth Low Energy. Boston: Artech House.

Investor Relations. 2019. Nordic Semiconductor. WWW-dokumentti. Saatavissa: https://www.nordicsemi.com/?sc_itemid=%7B3843917A-7CD6-4863-BA25-D6B74E70B386%7D [viitattu 04.05.2019]

Kothari, D., Shriram, K., Sundaram, R. & Murali, N. 2012. Embedded Systems. New Delhi: New Age International.

MCP73833. 2019. Microchip. WWW-dokumentti. Saatavissa: https://www.microchip.com/wwwproducts/en/MCP73833 [viitattu: 09.05.2019]

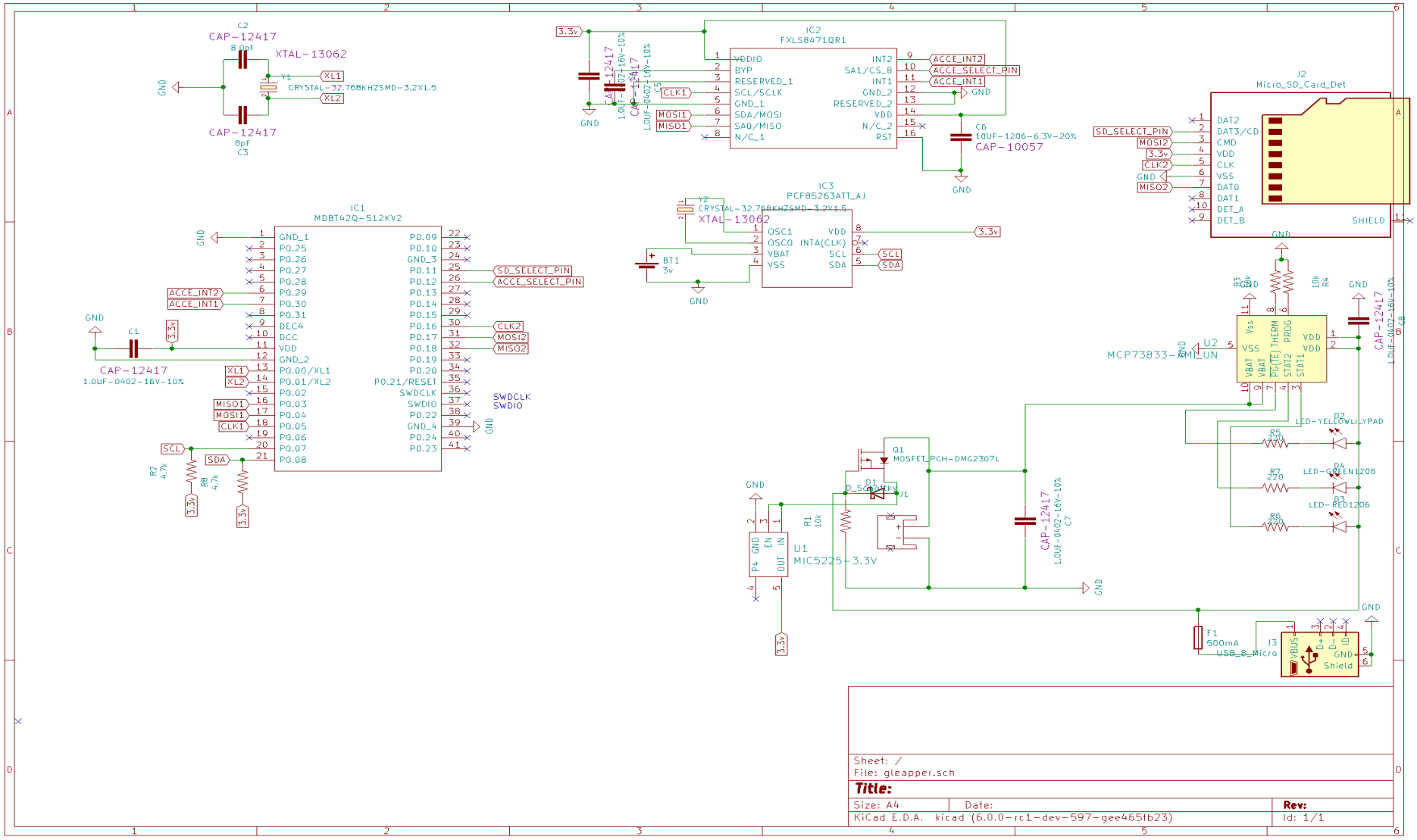
nRF52832. s.a. Nordic Semiconductor. Tuotetiedot. Saatavissa: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52832 [viitattu 09.05.2019]

Segger Embedded Studio. 2019. Segger. WWW-dokumentti. Saatavissa: https://www.segger.com/products/development-tools/embedded-studio/ [viitattu 04.05.2019]

What is Arduino. s.a. Arduino. WWW-dokumentti. Saatavissa: https://www.arduino.cc/en/Guide/Introduction# [viitattu: 09.05.2019]

KUVALUETTELO

Kuva 1. nRF52 DK 11
Kuva 2. Adafruit Flora 15
Kuva 3. Bluetooth LE SPI Friend 16
Kuva 4. Adafruit Triple-Axis Accelerometer 17
Kuva 5. MCP73833:n tyypillinen applikaatio 19
Kuva 6. Latauksen tilat. 19
Kuva 7. Ensimmäinen prototyyppi kasattuna 21
Kuva 8. Ohjelman vuokaavio yksinkertaistettuna 24
Kuva 9. sendLog funktio 25
Kuva 10. nus_data_handler funktio 26
Kuva 11. saveAccelerometerDataToDisk funktio 27
Kuva 12. Kiihtyvyysanturin rekisterit 28
Kuva 13. WHOAMI-rekisterin lukeminen 29
Kuva 14. Lukufunktio 30
Kuva 15. Monen tavun lukemiseen tehty funktio 30
Kuva 16. Kirjoitusfunktio 31
Kuva 17. Kiihtyvyystietojen lukufunktio 31
Kuva 18. Toinen prototyyppi kasattuna 32


C2
CAP-12417
8.0pF
XTAL-13062
Y1
CRYSTAL-32.768KHZSMD-3.2X1.5
XL1
XL2
GND
CAP-12417
8pF
C3
IC2
FXLS8471QR1
VDDIO
BYP
RESERVED_1
SCL/SCLK
GND_1
SDA/MOSI
SAO/MISO
N/C_1
INT2
SA1/CS_B
INT1
GND_2
RESERVED_2
VDD
N/C_2
RST
ACCE_INT2
ACCE_SELECT_PIN
ACCE_INT1
CLK1
MOSI1
MISO1
C6
1.0UF-1206-6.3V-20%
CAP-10057
C4
C5
CAP-12417
1.0UF-0402-16V-10%
3.3v
J2
Micro_SD_Card_Det
DAT2
DAT3/CD
CMD
VDD
CLK
VSS
DAT0
DAT1
DET_A
DET_B
SHIELD
SD_SELECT_PIN
MOSI2
CLK2
MISO2
IC1
MDBT42Q-512KV2
GND_1
P0.25
P0.26
P0.27
P0.28
P0.29
P0.30
P0.31
DEC4
DCC
VDD
GND_2
P0.00/XL1
P0.01/XL2
P0.02
P0.03
P0.04
P0.05
P0.06
P0.07
P0.08
P0.09
P0.10
GND_3
P0.11
P0.12
P0.13
P0.14
P0.15
P0.16
P0.17
P0.18
P0.19
P0.20
P0.21/RESET
SWDCLK
SWDIO
P0.22
GND_4
P0.24
P0.23
SWDCLK
SWDIO
CL
CAP-12417
1.0UF-0402-16V-10%
IC3
PCF85263ATT_AJ
OSC1
OSCO
VBAT
VSS
VDD
INTA(CLK)
SCL
SDA
Y2
CRYSTAL-32.768KHZSMD-3.2X1.5
XTAL-13062
BT1
3v
U2
MCP73833-AMI_UN
Vss
VBAT
VBAT
PG(TE)
THERM
PROG
STAT2
STAT1
VDD
VDD
VSS
R3
220
R4
10k
C8
CAP-12417
1.0UF-0402-16V-10%
D2
LED-YELLOWLILYPAD
D4
LED-GREEN1206
D3
LED-RED1206
R5
220
R7
220
R6
220
Q1
MOSFET_PCH-DMG2307L
D1
D_SCHOTTKY
J1
R1
10k
U1
MIC5225-3.3V
GND
EN
IN
OUT
C7
CAP-12417
1.0UF-0402-16V-10%
F1
500mA
J3
USB_B_Micro
VBUS
D+
D-
ID
GND
Shield
R2
4.7k
R8
4.7k
SCL
SDA
Sheet: /
File: gleapper.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (6.0.0-rc1-dev-597-gee4651b23)
Id: 1/1