Aalto-yliopisto, Sähkötekniikan korkeakoulu
ELEC-D0301 Protopaja
2018

Loppuraportti

Projekti #02
Pyörätietokone

Date: 31.8.2018

Santeri Miettinen
Eetu Reinikainen
Paavo Räihä
Oliver Lagerroos
Kalle Volanto

Information page

Opiskelijat

Santeri Miettinen
Eetu Reinikainen
Paavo Räihä
Oliver Lagerroos
Kalle Volanto

Projektipäällikkö
Santeri Miettinen

Sponsoroiva yritys
Aboense Oy

Aloituspäivä
4.6.2018

Submitted date
31.8.2018

Tiivistelmä

Projektin päämääränä oli rakentaa polkupyörään kiinnitettävä keskustietokone, joka näyttää urheilusuoritukseen liittyviä tietoja, kuten hetkellistä nopeutta, kiihtyvyyttä, sijainnin ja urheilijan sykkeen. Projektissa on yhdistynyt ohjelmointi, anturit, piirilevyn suunnittelu ja valmistaminen, käyttöjärjestelmän hallinta ja kaiken tämän yhteen integrointi. Tavoitteena on ollut tarjota yritykselle potentiaalinen ja toimiva prototyyppi, jota yritys voi tämän jälkeen itse halutessaan muokata ja liittää tähän omaa teknologiaansa.

Mahdollisia vaihtoehtoja laitteen käyttöjärjestelmäksi olivat Arduino, Raspberry Pie ja Android. Laitteen ehdot huomioiden Arduino todettiin liian tehottomaksi ja Raspberry liian yksinkertaiseksi järjestelmäksi. Ryhmässä päädyimme siis yhteisymmärrykseen Androidin suhteen, vaikka sen kehittämisestä ei ryhmän jäsenillä ollut aikaisempaa kokemusta.

Laitteen kehitysalustana on käytetty ARM-pohjaista kehitysalustaa sen tehokkaan prosessorin, helpon saatavuuden, laajan suosion ja Android-yhteensopivuuden takia. Tämä kehitysalusta toimii laitteen pohjana, jonka rinnalle suunniteltiin oma piirilevy. Olennaista laitteen toteutuksessa ovat olleet sen tehokkuus ja tarkkuus, joten kehitysalusta ja prosessori ovat siis olleet keskeisessä roolissa.

Kaiken halutun datan tuottamiseen tarvittiin antureiden lisäksi niille ohjelmat, jotka on toteutettu Android Studiolla. Nämä aliohjelmat yhdistettiin yhdeksi pääohjelmaksi, joka asennettiin laitteelle Androidin mukana. Koodien eri versioiden hallinta tapahtui projektikansion ja Gitin avulla.

Abstract

The goal of the project was to create a visual trainer development platform that is versatile enough to both house the next generation visual interface provided by the company, as well as the sensors needed for it to work as a bike computer. Also, as the project is intended to be used as a working demo by Aboense, it's aimed to be visually pleasing as well.

As we were given quite free hands on the whole platform, we had to start by pondering what to build the system on. A naive option would have been Arduino, but it was quickly stroke off from the list due to its low processing power. Second option would have been a Raspberry Pi, but it didn't make the cut either due to its bulk and has reputation as a common hobbyist platform. So, we ended up with android - perhaps the most ambitious option of them all since none of the group members had ever used it in any of our projects. Also, as we are working to make an actual product, we aimed to make the best product possible, not the one we currently had the know-how of producing.

So, we jotted down what we had to learn, what we had to do, and ordered the development platform suited for our needs. We settled on ARM based platform due to its compatibility with Android and hefty processing power - something needed for pleasing visuals and for the next generation interface provided by the company. Only if it had arrived that is. Waiting for the board we were forced to start the software development before it arrived. Working on the sensors together we were able to get everything we needed to output data, even the Bluetooth heart monitor. After breaking the ice of sensor to screen communication, the next thing was to combine them all together for a one cool looking app which is still a work in progress.

The code still being actively developed, and we are finally at the point where the development is stable. For version management it's a split between a self-hosted project folder and Git.

Sisällysluettelo

1. Johdanto

Projektin tarkoituksena oli kehittää polkupyörään yhteensopiva tietokone, joka tuottaa näytölle dataa, esimerkiksi urheilijan sykkeestä, pyörän nopeudesta ja kiihtyvyydestä sekä mittarin asennosta maanpinnan suhteen. Laitteessa on lisäksi Bluetooth- ja wifi-yhteys, joista Bluetooth-yhteyttä on hyödynnetty laitteen kommunikointiin sykevyön kanssa.

Tavoitteena on ollut luoda toimiva, tehokas ja kompakti piirilevy, joka on myös helposti yrityksen muokattavissa. Laitteen pohjana on käytetty ARM-pohjaista kehitysalustaa, sen suorituskyvyn, yleisyyden ja Android-yhteensopivuuden takia. Projekti on vaatinut paljon uusien asioiden opettelua ja asioiden selvittämistä, kuten piirilevyn suunnittelua ja Android-järjestelmän yhteensovittamista laitteeseen.

Laitteen toimivuutta testattiin käytännössä pyörällä projektin loppussa. Vastaavalla tavalla tuote demonstroitiin yleisölle elokuun demotilaisuudessa. Lopullinen prototyyppi toimii pohjana yrityksen tulevalle tuotteelle, joka yhdistää heidän oman tekniikkansa tavalliseen pyörämittariin, ja tuo uutta teknologiaa jopa huippu-urheiluun.

2. Tavoite

Tärkeimmät tavoitteemme ovat toimiva prototyyppi, sekä aikataulussa ja budjetissa pysyminen.
Tarkoitus on edistää yrityksen tuotekehitystä, ja tarjota demonstraatio tulevan tuotteen toiminnoista.

Muita keskeisiä tavoitteita:

- tarjota yritykselle potentiaalinen tuote-ehdokas
- luoda sovellus, jolla pystytään esittämään useat pyöräilijän tarvitsemat perustiedot
- laitteen helppo tekninen muokattavuus (voidaan esim. lisätä antureita)
- laite on riittävän tehokas (ei tarvitse olla optimaali)
- kestävyys urheilukäytössä (voidaan kehittää myöhemmin)
- laitteen toimintaa pystytään demonstroimaan sen ollessa kiinni pyörässä
- jokainen ryhmän jäsen oppisi jotain uutta

3. Ohjelmointi

Ohjelmat ovat tuotettu Android Studiolla, jossa ohjelmointikielenä käytetään ensisijaisesti Javaa. Vaikka ryhmässä ei ollut ennen projektia kokemusta Javan käyttämisestä ohjelmointikielenä, koimme sen silti olevan fiksuin vaihtoehto ohjelmien tekoon, koska sille löytyy netistä kattavimmat dokumentaatiot. Toteutimme Android Studiolla ohjelmat sykevyölle, gps:lle, sekä kiihtyvyyden, että nopeuden mittaamiselle.

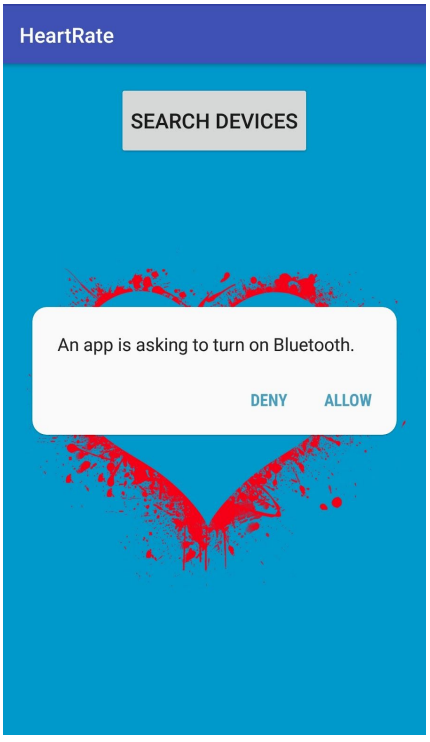
3.1. Sykevyö

3.1.1. Yleistä

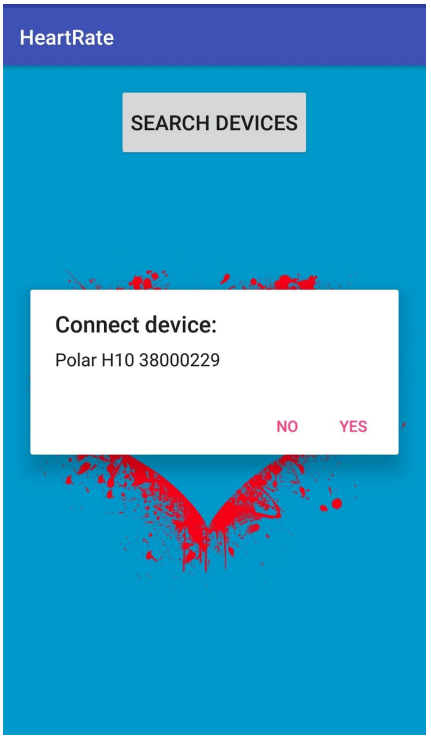
Laitteen yhdistäminen sykevyöhön tapahtuu Bluetooth-yhteyden välityksellä. Yhdistäminen ja tiedon kerääminen sykevyöstä onnistuu kehittämällämme sovelluksella ainakin Polar-merkkisten sykevöiden kanssa, mikä oli yksi yrityksen toiveista.

3.1.2. Sovellus

Sovelluksen avautuessa käyttäjää pyydetään kytkemään laitteen Bluetooth-yhteys päälle (Kuva 1.). Tämä pyyntö toistuu niin pitkään, kunnes Bluetooth hyväksytään. Tämän jälkeen käyttäjän painaessa “Search devices”-painiketta, alkaa laite etsimään Polar-merkkistä sykevyötä. Sellaisen löydyttyä, tulostuu laitteen näytölle ikkuna (Kuva 2.), joka varmistaa käyttäjältä haluaako hän yhdistää laitteensa löydettyyn sykevyöhön. Mikäli käyttäjä kieltäytyy yhdistämisestä, aloittaa laite sykevyön etsinnän uudelleen. Puolestaan käyttäjän hyväksyessä yhdistämisen, yhdistyy laite valittuun sykevyöhön ja alkaa esittää näytöllä reaaliaikaista lukemaa käyttäjän sykkeestä (Kuva 3.).



Kuva 1. Bluetoothin kytkeminen



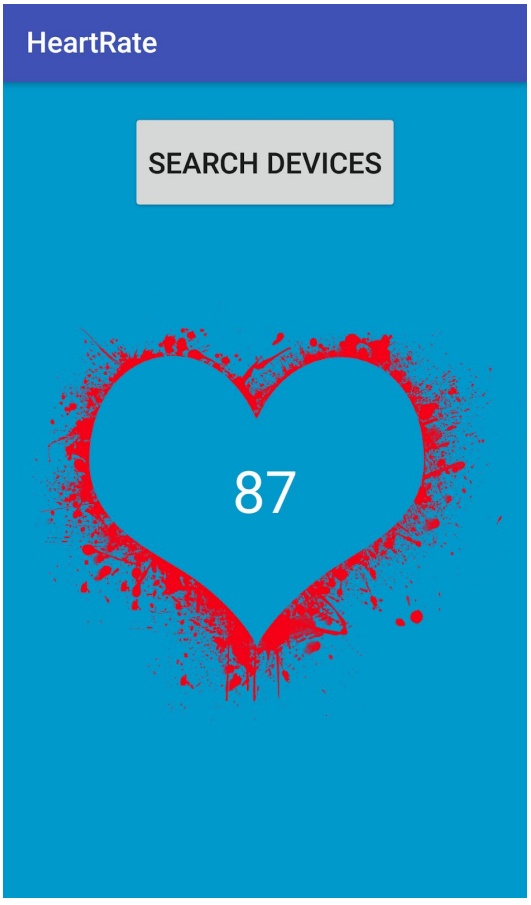
Kuva 2. Laitteen valitseminen

3.1.3. Kehitys

Kyseinen sovellus oli kehitetyistä sovelluksista ensimmäinen, jota aloitettiin tuottamaan. Se oli lopulta yllättävästi myös ensimmäinen, joka saatiin kunnolla toimimaan. Koska ryhmällä ei ollut mitään aiempaa kokemusta Java-ohjelmointikielestä tai Android Studiosta, kului aluksi paljon aikaa perusasioiden opetteluun ja testaukseen. Kun perusasiat saatiin kuntoon, alkoi itse varsinaisen sovelluksen rakentaminen.

Sovelluksen rakentaminen lähti liikkeelle yksinkertaisesti siitä, kuinka Bluetooth-yhteys saadaan kytkettyä päälle, mutta pikkuhiljaa tietoisuus asioista karttui niin, että tietyn sykevyön etsiminen ja parittaminen käytettävän laitteen kanssa onnistui ongelmitta. Varsinaisen laitteiden välisen kommunikaation mahdollistaminen ja toteuttaminen osoittautui kuitenkin suhteellisen haastavaksi. Lopulta tässäkin onnistuttiin hyödyntämällä Polarin kehittäjille tarjoamaa esimerkkikoodia.

Tämän jälkeen sovelluksen kehityksessä keskityttiin lähinnä parantamaan sen ulkoasua ja käyttäjäkokemusta. Esimerkiksi Bluetooth-yhteyden kytkemistä helpotettiin ja ylimääräisiä toimintoja karsittiin.



Kuva 3. Syke

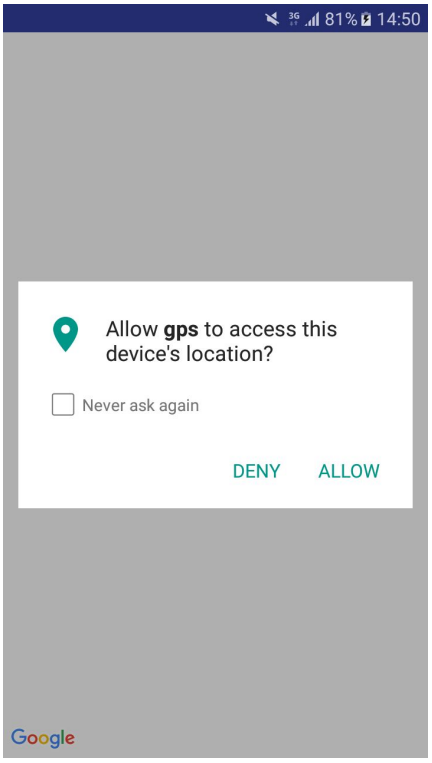
3.2. Kartta

3.2.1. Yleistä

Karttasovelluksen käyttöliittymän pohjana käytetämme Google Maps kirjastoa, johon saadaan käyttöoikeus henkilökohtaisen API-avaimen (Application Programming Interface key) avulla. API-avaimen saa halutessaan haettua Google Mapsin sivuilta.

3.2.2. Sovellus

Käyttäjän avatessa sovelluksen, ensimmäisenä varmistetaan onko sovelluksella oikeudet käyttää laitteen sijaintia. Mikäli oikeuksia ei ole, ilmestyy ruutuun ikkuna (Kuva 4), jossa pyydetään käyttäjää antamaan sovellukselle oikeus laitteen sijaintiin. Tämän jälkeen tarkistetaan vielä onko laitteen paikannus asetettu päälle. Jos näin ei ole, ohjaa sovellus käyttäjän asetuksiin, jossa hän voi kytkeä paikannuksen päälle. Tämän jälkeen sovellus alkaa hakemaan käyttäjän sijaintia ja sen löytäessään, piirtää sovellus kartalle kyseiseen sijaintiin merkin (Kuva 5.). Käyttäjän liikkuessa sijainti päivittyy jatkuvasti ja uusi sijainti päivitetään aina olemaan keskellä laitteen näyttöä. Kartta myös kääntyy vastaamaan käyttäjän kulkemaa suuntaa. Sijainnin haun tiheyttä ja kartan suurennusta, kuten muitakin ohjelman parametreja, voidaan muuttaa haluamamme mukaan.



Kuva 4. Lupa käyttää sijaintia

Kuva 5. Karttakuva

3.2.3. Kehitys

Sovelluksen kehittäminen lähti liikkeelle siitä, että sovellukselle täytyi saada jokin karttapohja, jonka päälle kehitetään varsinaiset sovelluksen ominaisuudet. Tähän päätettiin käyttää Googlen tarjoamaa karttapohjaa, johon saatiin käyttöoikeus jo mainitun API-avaimen avulla.

Tämän jälkeen aloitettiin varsinainen oma osuus sovelluksen kehittämisestä ja ensimmäisenä vuorossa oli merkin lisääminen kartalle, aluksi itse valitsemaamme sijaintiin. Seuraava vaihe oli luonnollisesti hakea käyttäjän oikea sijainti ja lisätä merkki tähän uuteen sijaintiin. Kun laitteen sijainti lopulta saatiin haettua, niin että ohjelma piirsi näytölle aina uuden merkin laitteen uuteen sijaintiin, ilmeni että edelliset merkit eivät poistuneet ikinä näytöltä ilman, että sovellus suljettiin. Tämä ongelma saatiin onneksi kuitenkin helposti ratkaistua. Samoihin aikoihin sovellukseen myös lisättiin, että kartan suurennos pysyy aina vakiona.

Viimeinen suurempi muutos sovelluksen käytettävyyteen oli se, että kartta kääntyy vastaamaan käyttäjän kulkemaa suuntaa. Tämän jälkeen keskityttiin korjaamaan sovelluksessa havaittuja virheitä. Suurin ongelma liittyi siihen, että ensimmäisellä käyttökerralla sovellus kaatui joka kerta, mikä johtui siitä ettei käyttäjä pystynyt hyväksymään sovelluksen paikannuksen käyttöä ennen kuin sovellus sijaintia alkoi etsimään. Tähänkin ongelmaan löydettiin kuitenkin lopulta ratkaisu.

3.2.4. Koodiesimerkki

Tässä näkyy esimerkki kartta sovelluksen koodista kohdasta, joka tarkistaa käyttäjän sijainnin, sen muuttuessa siirtää merkin vastaamaan uutta sijaintia ja lopuksi siirtää uuden merkin ruudun keskelle.

```
private void myListener(){
    listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            Lat = location.getLatitude();
            Lng = location.getLongitude();
            if (mMap != null && Lng != 0.0 && Lat != 0.0) {
                if (merkki != null) {
                    merkki.remove();
                }
                LatLng position = new LatLng(Lat, Lng);
                merkki = mMap.addMarker(new MarkerOptions().position(position).title("Marker"));
                //Numero on zoomauskerroin
                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(position, v: 17));
                updateCameraBearing(mMap, location.getBearing());
                Log.d(TAG, msg: "Location: " + Lat + " " + Lng);
            }
        }
```

Kuva 6.

3.3. *Anturit*

3.3.1. Yleistä

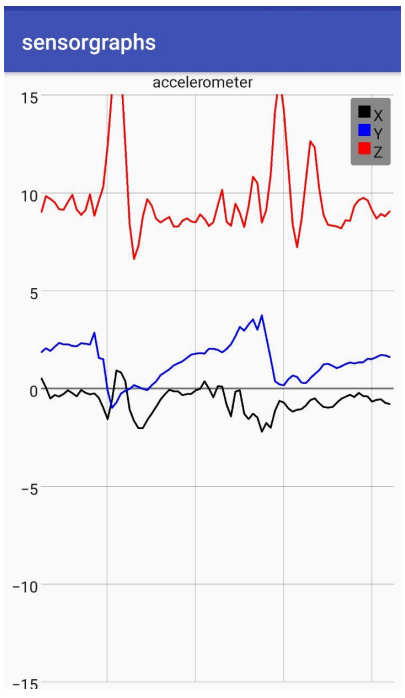
Dragonboard 410c -kehitysalustan päälle liitettävään piirilevyyn on liitetty kiihtyvyysanturi ja gyroskooppi. Lisäksi kehitysalustan GPIO-pinneihin on liitetty magneettikytkin, joka menee kiinni, kun etupyörän puolaan kiinnitetty magneetti kulkee kytkimen ohi. Kun kytkin on auki, GPIO-pinnistä luettu arvo on 1. Jos taas kytkin on kiinni, luettu arvo on 0. GPIO-pinnien käyttö tapahtuu koodissa Androidin komentorivin kautta.

3.3.2. Sovellukset

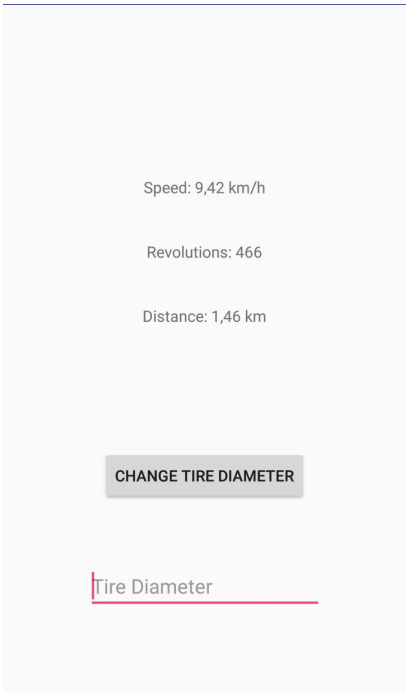
Anturien mittaama data voidaan näyttää käyttäjälle joko yksittäisinä numeroina tai reaaliaikaisena käyränä. Prototyyppiä varten tehtiin kaksi demosovellusta. Molemmat sovellukset ovat toteutettu kokonaan Javalla.

Toinen piirtää reaaliaikaista käyrää kiihtyvyysanturin (tai gyroskoopin) antamasta datasta käyttäen Androidin GraphView-kirjastoa (Lähde 3). Jotta sovellus toimii, kiihtyvyysanturin tulee olla alustettu oikein I2C-väylän kautta sovellukseen ja Android-käyttöjärjestelmään. (Kuva 7)

Toinen sovellus hyödyntää pyörän etuhaarukkaan kiinnitettävää magneettikytkintä, ja mittaa kuljettua matkaa ja nopeutta. Kun etupyörän puolaan kiinnitetty magneetti kulkee kytkimen ohi, sovellus havaitsee sen, ja laskee havaintojen perusteella kuljetun matkan ja nopeuden. Sovelluksessa on mahdollista myös määrittää renkaan koko, mutta oletusarvoisesti koko on asetettu vastaamaan 28″ pyörää. Tämän sovelluksen toiminta on mahdollista toteuttaa pääohjelmassa taustalla, jolloin muita sovelluksia käytettäessä kuljettu matka saadaan mitattua. Prototyypillä kyseinen ei kuitenkaan onnistu liian alhaisen prosessointitehon takia. (Kuva 8)



Kuva 7. Käyrä kiihtyvyysanturin datasta



Kuva 8. Matka/nopeus-sovellus

3.3.3. Kehitys

Kiihtyvyysanturi/gyroskooppi -sovellus

Ensin täytyi selvittää, miten kiihtyvyysanturin (tai gyroskoopin) tuottamaa dataa voidaan lukea. Tätä varten löytyi hyvä dokumentaatio Android Developers -sivuilta (Lähde 5). Seuraavaksi täytyi löytää hyvä keino tehdä saadusta datasta reaaliaikainen kuvaaja. Parhaaksi tavaksi osoittautui GraphView-kirjasto, johon löytyi hyvä dokumentaatio Android Graphview -sivuilta (Lähde 3). Seuraavaksi opettelimme käyttämään kirjastoa mallikoodien (Lähde 6) avulla, jonka jälkeen koodin kirjoittaminen sujui lähes ongelmitta.

Nopeus/matka-sovellus

Aluksi lähdimme tekemään koodia Android Things -kirjaston avulla. Pian kuitenkin selvisi, että käytettävä Android-versio (Android 5.1) ei tue tätä kirjastoa. Löysimme kuitenkin mallikoodin Java-luokasta, joka käyttää Androidin komentoriviä GPIO-pinnien ohjaamiseen (Lähde 6). Koodasimme sen avulla toisen Java-luokan, joka laskee kuljetun matkan ja nopeuden, ja jota itse sovellus voi suorittaa erillisessä prosessissa.

3.3.4. Kehitysmahdollisuuksia

Molempien sovellusten ulkoasua voisi kehittää mieluisamman näköiseksi. Lisäksi havaitsimme, että Java ei osaa lukea tarpeeksi nopeasti laitteen GPIO-pinneiltä saamaansa tietoa, minkä takia läheskään jokaista pyörän pyörähdystä ei pystytä koodissa havaitsemaan. Matka/nopeus-sovelluksen koodi olisi siis todennäköisesti parempi toteuttaa jollain muulla ohjelmointikielellä, kuten C:llä.

3.4. Pääsovellus

3.4.1. Yleistä

Kun muut sovellukset oli saatu suhteellisen hyvään kuntoon, kokoottiin ne yhteen kyseiseen pääsovellukseen. Lähdekoodista haluttiin tehdä mahdollisimman selkeää, joten eri sovellukset järjestettiin eri kansioihin.

3.4.2. Sovellus

Sovelluksessa (Kuva 9.) on painikkeet, jotka ohjaavat muihin kehittämiimme sovelluksiin. Painikkeista ylimmäinen on karttasovelluksen , vasemmanpuolimmainen nopeus- ja matka-sovelluksen, alin sykevyö-sovelluksen ja oikeanpuolimmaisin kiihtyvyys sovelluksen painike. Sovellukseen taustakuvana toimii yrityksen logo. Haluttua painiketta painaessa kyseinen sovellus avautuu erillisenä ikkunana, ja pääikkuna jää taustalle. Käyttäjä voi palata pääikkunaan painamalla Androidin paluu-painiketta.

Kuva 9. Pääsovellus

3.5. Yhteenveto

Sovellus	***Toiminnot***
Sykevyö	● Kytkee tarvittaessa laitteen Bluetooth-yhteyden päälle, käyttäjän tämän hyväksyessä ● Etsii Polar-merkkistä sykevyötä, käyttäjän painaessa etsinnän aloittavaa painiketta ● Löytäessään sopivan laitteen ilmoittaa tästä käyttäjälle ja kysyy häneltä onko kyseinen laite se johon hän haluaa yhdistää ● Käyttäjän kieltäytyessä, aloitetaan laitteen etsiminen uudelleen ● Käyttäjän hyväksyessä, yhdistyy laite sykevyöhön ja alkaa esittämään sykevyöstä saatuja tuloksia henkilön sykkeestä
Gps	● Pyytää tarvittaessa lupaa käyttää laitteen sijaintia ● Ohjaa tarvittaessa käyttäjää kytkemään laitteen paikannuksen päälle ● Aloittaa automaattisesti laitteen paikantamisen ● Jos sovellusta käytetty aiemmin, sijoittaa kartalle aluksi merkin viimeiseen tunnettuun sijaintiin ja sijoittaa merkin ruudun keskelle ● Löytäessään uuden sijainnin, päivittää merkin kartalle uuteen sijaintiin ja sijoittaa uuden merkin ruudun keskelle ● Kääntää kartan vastaamaan käyttäjän kulkemaa suuntaa
Gyroskooppi, Kiihtyvyysanturi	● Lukee kiihtyvyysanturin (tai gyroskoopin) mittaamaa dataa ● Muodostaa reaaliaikaisen käyrän mittausdatasta
Nopeus- ja Matkamittari	● Pyrkii laskemaan pyörän pyörähdysten määrän ja pyörähdyksiin kuluvan ajan avulla pyöräilijän nopeuden ● Laskee pyöräilijän kulkeman matkan ● Matkan ja nopeuden laskeminen tapahtuu taustalla erillisessä prosessissa ● Päivittää tiedot näytölle sekunnin välein

4. Laitteisto

4.1. Dragonboard 410c

Projektin kehitysalustana päädyttiin käyttämään Qualcommin julkaisema Snapdragon 410E prosessoriin perustuva kehitysalustaa, joka on suunniteltu nopeaan ohjelmistojen kehittelyyn ja prototyyppien tekemiseen. Piirissä on itsessään Wi-Fi, Bluetooth ja GPS yhteydet. Tarkemmat tekniset tiedot löytyvät raportin liitteenä olevasta manuaalista.

4.2. Serveri

Loimme jo projektin alkuvaiheessa Ubuntun Apachea käyttäen projektikansion, jota käytimme SSH -yhteydellä yhteisillä käyttäjillä. Se oli nopea tapa lähtökohtaisesti saada omat ideat ja tiedostot jakoon. Serveri itseisännöitynä on kykenevä Adb ja Fastboot palveluihin, eli boardin käsiimme saatua olisimme voineet ohjelmoida ja käyttää sitä etänä. Myös androidin asennus olisi onnistunut etänä Fastboot:tia käyttäen. Näitä tämän serverin ominaisuuksia emme kuitenkaan tulleet hyödyntäneeksi. Tärkein käyttötarkoitus serverillä oli toimia portaalina koodiemme versionhallinnassa ja tiedostojen jakamisessa.

4.3. Mobvoi Ticwatch E

Hankimme myös Android Wear 2.0 pohjaisen älykellon, Mobvoi Ticwatch E:n. Kellon tarkoitus oli toimia varakehitysalustana, jos alkuperäinen Dragonboard 410c ei olisi toiminut halutulla tavalla. Kehittämämme sovellukset eivät kuitenkaan olisi täysin toimineet suoraan Android Wearissa, vaan ne olisi pitänyt tehdä uudelleen tai vähintäänkin muokata laitteelle yhteensopiviksi. Kello oli lopullisen prototyypin viimeinen vaihtoehto, johon ei tarvinnut koskaan siirtyä.

5. Piirilevy

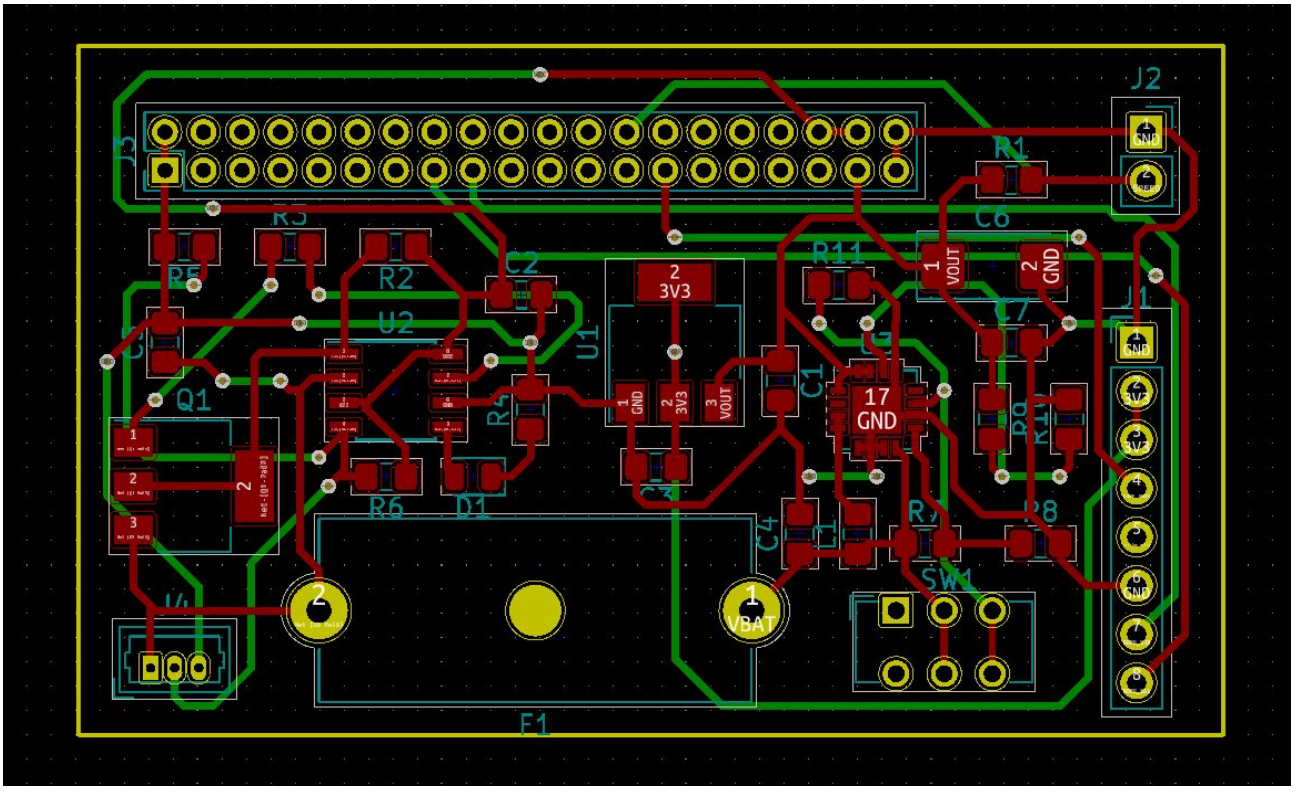
Projektin yhtenä osana oli oman piirilevyn suunnitteleminen. Piirilevyä tarvitaan laitteen virranhallintaan sekä nostamaan litium-akun jännite 5V, jolla laite käynnistyy. Piirilevyyn tulee lisäksi liittimet antureihin ja Dragonboardiin.

5.1. Pohjapiirrustus

Piirilevy koostuu seuraavista komponenteista:

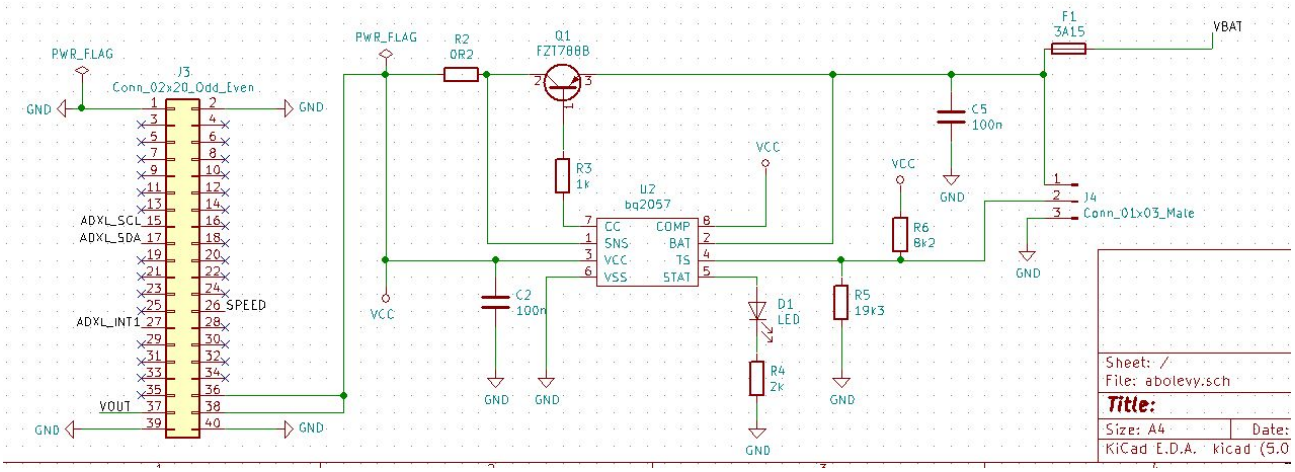
U1 – regulaattori
U2 – virranhallintapiirin ohjain
U3 – regulaattorihakkuri
Q1 – transistori
F1 – sulake
D1 – ledi

SW1 – kytkin
F1 – sulake
R# – resistanssi
C# – kondensaattori
J# – liitin

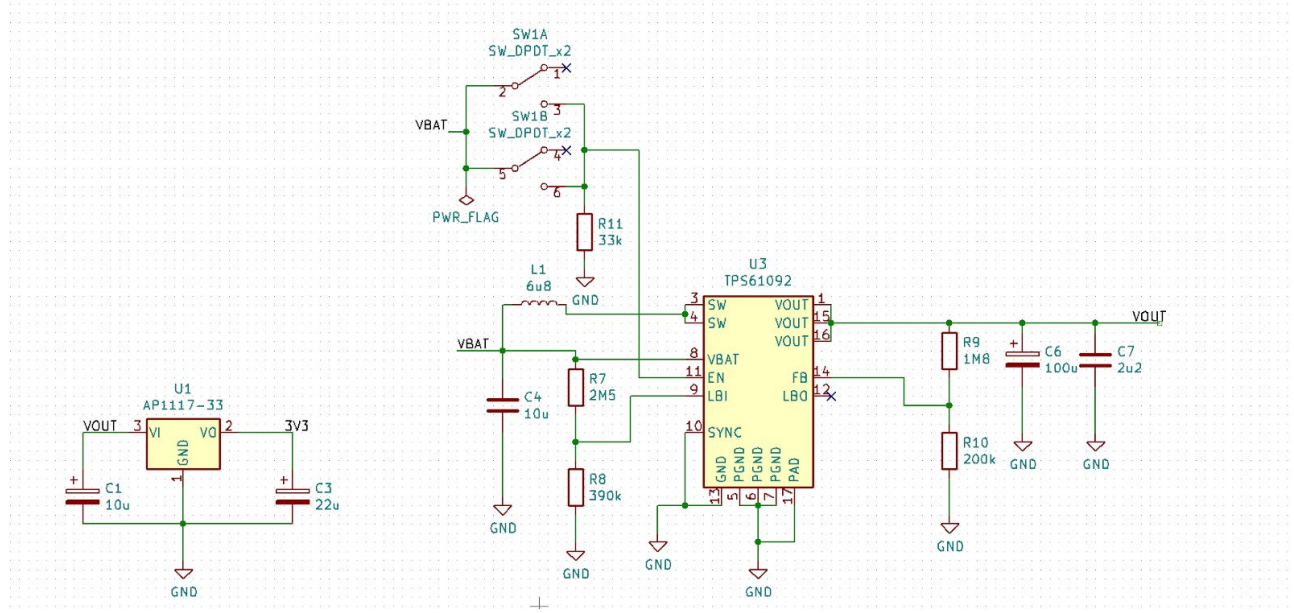


Kuva 10. Piirilevy

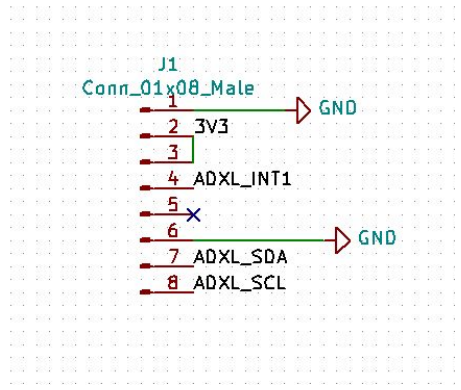
Komponenttien tarkemmat kytkentäkaaviot on esitetty alla:



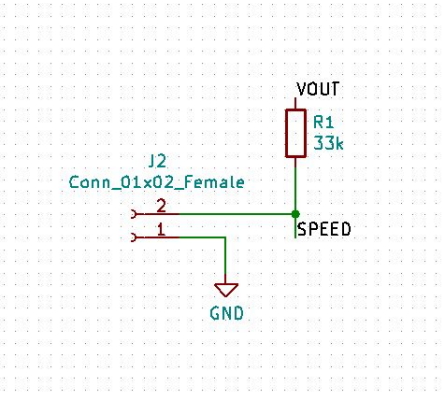
Latauspiiri ja liitäntä Dragonboard 410c:hen



Regulaattori (v.) ja regulaattorihakkuri (o.)



Kiihtyvyysanturin liitäntä



Nopeusanturin liitäntä

5.2. Toiminta

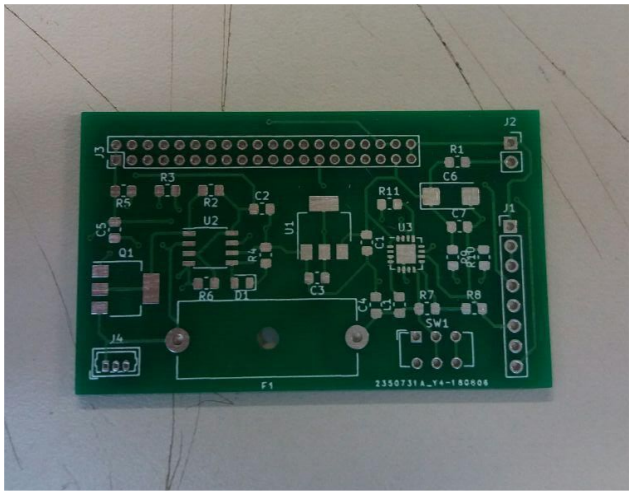
Piirilevy on suunniteltu KiCad nimisellä ohjelmalla,. Regulaattorin (U1) tehtävä on nostaa akun (liitin J4) lähdejännite 5V laitteen käynnistämistä ja toimimista varten. Sen toimintaan liittyy ns. regulaattorihakkuri (U3), joka pilkkoo virtaa ja tuottaa ulostulona tietyn jännitteen, meidän tapauksessamme 0.5A. Valitettavasti laitteemme vaatii käynnistyäkseen vähintään 1.0A suuruisen virran, joten jouduimme muuttamaan kytkentää juottamalla C6:n tilalle rinnakkain suuremman kondensaattorin. Tämän toimiessa epävakaasti päädyimme lopulta muuttamaan koko piirin virransyöttöä. Ongelman voi kuitenkin seuraavassa prototyyppissä helposti ratkaista vaihtamalla U3 komponenttia.

U2 muodostaa yhdessä ympärillä olevien komponenttien kanssa virranhallintapiirin, jonka tarkoitus on huolehtia laitteen ja käyttäjän turvallisuudesta sekä laitteen toimivuudesta: jännite tai virta ei kasva liian suureksi, eikä myöskään ole liian pieni. Lisäksi piiriin on kytketty sulake (F1), joka luonnollisesti palaa virheellisessä virran toiminnassa.

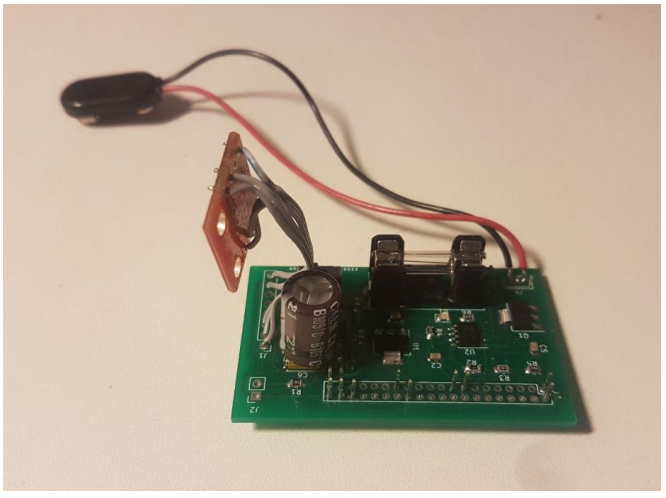
Piirilevyssä on myös liittimet kiihtyvyys- (J1) ja nopeusanturille (J2), akkuliitin (J4) sekä liitin Dragonboardiin (J3), joka on siis väylä kaikelle kommunikaatiolle piirin ja laitteen välillä. Nopeusanturi lukee suunnitelman mukaisesti dataa (jännitteen suuruus) Snapdragon:in GPIO-pinnistä nro 26 (vihreä johdin, oikea yläkulma). Datan lukeminen kyseisestä pinnistä kuitenkin jostain syystä epäonnistui, minkä takia muutimme kytkennän viereiseen pinniin nro 25, josta saimme viime hetkillä nopeusanturin kommunikoimaan pyörän nopeuden näyttävän sovelluksen kanssa.

5.3. Käytännössä

Alapuolella on kuvat, kun piirilevy on saapunut tilauksesta (Kuva 11) ja kun suurin osa komponenteista on juotettu kiinni (Kuva 12).



Kuva 11. Tyhjä piirilevy



Kuva 12. Valmis piirilevy

6. Android

6.1. Yleistä

Meillä oli valittavissa Dragonboardin käyttöjärjestelmäksi Windows IoT, Android 5.1 tai Linux Debian. Kaikissa oli omat hyvät ja huonot puolensa, mutta päädyimme asentamaan Androidin sen helpon sovelluskehityksen, hyvän dokumentoinnin ja todella helpon muokattavuuden takia. Androidin versiot 5.1, 6.0 ja vuoden 2019 aikana julkistettava Android Q, saatiin toimimaan laitteella, mutta kernel ongelmien takia päädyttiin asentamaan laitteelle vanha, mutta parhaiten toimiva Android 5.1. Android versiot 5.1 ja 6.0 löytää asennusohjeiden kanssa Qualcommin developer networkin kautta (lähde 8) ja android Q on osa googlen ylläpitämää Android AOSP projektia.

6.2. Android 5.1

Android 5.1 on Qualcommin itse ylläpitämä Dragonboard:ille tarkoitettu versio. Android 5.1 rakentaminen oli helppoa Qualcommin asennuspaketin ja ohjeiden ansiosta. Kyseisestä android versiosta olisi myös ollut valmiiksi rakennettu versio, mutta päädyimme silti rakentamaan oman, vaikka emme tehneet mitään muutoksia lähdekoodiin.

6.3. Android Q

Android Q:n asentaminen olikin hieman haastavampi homma. Kyseinen versio on vielä kovasti kehityksessä Android AOSP projektin alla. Android Q rakentamisohjeet löytyvät lähteestä 7. Android Q toimi hyvin, vaikkakin laite ei osannut asennuksen jälkeen antaa signaalia Dragonboardista löytyvään USB mux:ille, joka vaihtaa USB tilan pois USB hubista, USB OTG liittimeen. Vika on korjattavissa kerneliä muokkaamalla, mutta ajan puutteen vuoksi kyseistä muokkausta ei keretty tekemään.

7. Projektitoiminta

7.1. Tavoitteiden saavuttaminen

Alle on kuvattu projektimme keskeiset tavoitteet, niiden toteutuminen ja miten jatkossa päästäisiin paremmin kyseisiin tavoitteisiin.

Tavoite	**Toteutuminen**	**Parannus**
toimiva prototyyppi	onnistui, virransyötössä kompromissi	omaan piirilevyyn komponentin muutos
potentiaalinen tuote-ehdokas	välttävä, laite on melko karkea prototyyppi	tehtäisiin seuraavaksi versio 2.
ominaisuudet	kiihtyvyyden esittämistä lukuunottamatta riittävät	kernelin onnistunut muokkaamisen
tekninen muokattavuus	helppoa, voidaan suunnitella uusi piirilevy, joka liitetään laitteeseen	WARP7 parempi kehitysalusta
tehokkuus	hieman odotettua heikompi	Android version vaihtaminen
ulkoasu	protyyppimäinen	suunnitellaan lopullinen kotelo ja huomioidaan urheilukäyttö
demonstrointi	onnistunut kiihtyvyyttä lukuun ottamatta, lisäksi nopeussovellus toimii hitaasti	nopeussovelluksen kirjoittaminen C:llä ja kernelin muokkaaminen
uuden oppiminen	opimme piirilevyn suunnittelua, juottamista, Androidin asentamista, ongelmien ratkaisua, projektinhallintaa	yhteinen läpikäynti toisen vastuualueista
ajankäyttö	tilausongelmien takia työmäärä kertyi projektin loppupuolelle, määrällisesti riittävä	tehokkaampi riskien hallinta

Tyytyväisiä projektissa olemme sykevyö- ja gps-ohjelman toimintaan. Molemmat toimivat sulavasti, ja niissä on yksinkertainen ja käyttäjää miellyttävä ulkoasu. Parannettavaa on nopeusohjelmassa, joka toimii hitaasti, sekä itse suunniteltua piirilevyä voisi luonnollisesti parantaa. Projektiin käytetyn ajan huomioiden pääsimme kohtuullisen lopputulokseen.

7.2. Aikataulu

Projektisuunnitelmassa (Liite 1.) arvioitu aikataulu on pitänyt melko hyvin paikkansa. Odotettua enemmän aikaa vei Android versioiden kokeilu ja asentaminen, sillä ohjelmien kanssa yhteensopivan version etsiminen ja toteuttaminen osoittautui oletettua haastavammaksi. Lisäksi antureiden hallinta Android-pohjaisessa laitteessa on hankalaa, joten myös tämä tuotti odotettua enemmän töitä.

Piirilevyn suunnittelemiseen käytettiin sille varattua vähemmän aikaa. Tilauksen tekemisen kiireellisyyden takia tämä piti kuitenkin hoitaa lyhyemmässä ajassa kuin olimme alun perin suunnitelleet. Varsinaisesti projektin aikana ei ilmennyt työtehtävää, joka olisi tarvinnut suunniteltua vähemmän aikaa. Tulevissa projekteissa on syytä varautua lisääntyvään työmäärään, jota tulee, kun asiat eivät toimi heti niin kuin on suunniteltu.

7.3. Riskianalyysi

Kurssin alussa havainnoimme seuraavat riskit. Nyt taulukkoon on täydennetty myös osioilla, jotka kertovat toteutuiko riski ja miten se voidaan ensi kerralla välttää.

Riski	Vakavuus	Toteutuminen	Kuvailu	Parannus
myöhäinen aloitus	vakava	ei	-	-
tilatut osat ovat viallisia	kohtalainen	ei	-	-
Androidia ei saada toimimaan, tai siihen menee kohtuuttomasti aikaa	vakava	kyllä	uudella Androidilla sovellukset eivät toimineet ja vanhalla näyttö ei tuottanut kuvaa	valita nopeammin Android versio, jolla sovellukset toimivat, näyttö helpompi vaihtaa
aikataulu pettää	vakava	kyllä	WARP7:n toimitus ei onnistunut	pyytää tietoa tilauksen kulusta tilaajalta tai tilata Suomesta
sensorit eivät toimi oikein tai ovat epätarkkoja	kohtalainen	ei	-	-
komponentti rikkoutuu	vakava	kyllä	näyttö meni rikki	varanäyttö tilattiin välittömästi
huono tai vääränlainen kotelointi	lievä	ei	-	-
ryhmän vaikea nähdä säännöllisesti	lievä	ei	-	-
budjetti ylittyy	lievä	ei	-	-
yrityksen toiveet muuttuvat	lievä - vakava	ei	-	-
sekundäärinen piirilevy ei valmistu ajoissa tai sen tekemiseen kuluu liikaa resursseja	kohtalainen -vakava	ei	-	-

Suurimmasta osasta riskejä olimme tietoisia projektin alussa ja pyrimme ennakoimaan näitä muun muassa aloittamalla ohjelmien tekemisen heti projektisuunnitelman valmistumisen jälkeen ja tilaamalla tarvittavat osat mahdollisimman pian. Pyrimme tekemään tasaisesti töitä läpi projektin saadaksemme ajoissa valmista kohtuullisella laadulla.

7.4. Laatu

Tuotteen laadun mittareina voidaan pitää laitteen tehokkuutta, yrityksen tyytyväisyyttä, prototyypin toimivuutta ja ulkoasua, elektronista toimivuutta ja kehittämisen joustavuutta. Laadukasta prototyypissä olivat ohjelmisto ja elektroniikka: laite kommunikoi toimivasti sovellusten kanssa ja kytkennät olivat luotettavia.

Aikataulun kiristyessä kehityksen mukana tulevien ongelmien takia, laatua jouduttiin ryhmän yhteisellä päätöksellä laskemaan. Laatua laski muun muassa tehottomamman Snapdragon 410c:n hankinta ja laitteen melko karu ulkoasu. Projektin alussa suunniteltu laatu ei toteutunut toivotulla tavalla. Laatua on kuitenkin helppo nostaa varsinaista tuotetta ajatellen käyttämällä toista kehitysalustaa, valitsemalla laadukkaat komponentit piirilevyyn ja suunnittelemalla käyttäjäystävällinen design laitteelle.

8. Yhteenveto ja johtopäätökset

Olemme projektin aikana kehittyneet projektinhallinnassa ja ongelmien ratkaisussa, sekä oppineet yritysyhteistyön alkeita. Lisäksi ryhmän jäsenet ovat kukin oppineet käytännön osaamista omasta vastuualueestaan. Vastuualueita ovat olleet ohjelmointi, piirilevyn suunnittelu, Androidin asentaminen, elektroniikka ja integrointi.

Yritys voi hyödyntää sovellusten avointa lähdekoodia ja ottaa mallia piirilevystä, johon anturit ja mahdollisesti akun virranhallinta liitetään. Android-käyttöjärjestelmä on myös kehityskelpoinen alusta, johon yritys voi itse helposti kehittää lisää sovelluksia ja siten lisätä laitteen ominaisuuksia. Prototyyppi ei luonnollisesti ole valmis malli tulevalle tuotteelle, mutta siinä on elementtejä, joita voidaan hyödyntää lopullisen tuotteen suunnittelussa. Yritys saa prototyypin ja sen dokumentaation avulla tietoa myös kriittisistä vaiheista laitteen kehityksessä.

Liitteet

1. *Projektisuunnitelma*
2. *Dragonboard 410c:n manuaali*

Lähteet

1. https://developer.polar.com/wiki/H6,_H7,_H10_and_OH1_Heart_rate_sensors
2. https://developer.android.com/guide/topics/connectivity/bluetooth
3. http://www.android-graphview.org/
4. https://developer.android.com/reference/
5. https://gist.github.com/rosterloh/c4bd02bed8c5e7bd47c5
6. https://github.com/jjoe64/GraphView-Demos/
7. https://www.96boards.org/documentation/consumer/dragonboard/dragonboard410c/guides/aosp.md.html
8. https://developer.qualcomm.com/hardware/dragonboard-410c/