Tampereen ammattikorkeakoulu

# Arctic tracker

## Rekikoiran aktiivisuusjärjestelmä

Ari Savolainen

OPINNÄYTETYÖ
Toukokuu 2021

Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikka
Tietoliikennetekniikka ja tietoverkot

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikka
Tietoliikennetekniikka ja tietoverkot

SAVOLAINEN, ARI:
Arctic tracker
Rekikoiran aktiivisuusjärjestelmä

Opinnäytetyö 65 sivua, joista liitteitä 8 sivua
Toukokuu 2021

---

Opinnäytetyön päätavoitteena oli luoda arkkitehtuurisuunnitelma rekikoirille suunnatusta aktiivisuusjärjestelmästä ja toteuttaa aiheeseen liittyen ensimmäinen prototyyppi alustariippumattomasta mobiiliohjelmasta, joka voidaan liittää arktisiin olosuhteisiin suunniteltuun sulautettuun järjestelmään. Suunniteltu aktiivisuusjärjestelmä seuraa rekikoirien työsuoritteiden aikaista rasitusta ja hyvinvointia.

Työssä selvitettiin mahdollisuudet luoda rekikoirille suunnattu aktiivisuusjärjestelmä rekikoirien fyysisten ominaisuuksien sekä työskentelyolosuhteissa vallitsevien ääriolosuhteiden perusteella. Tukena selvitystyössä käytettiin kokeneiden lajiharrastajien kokemuksia ja näkemyksiä, joiden perusteella kehitettiin aktiivisuusjärjestelmältä vaadittavat turvallisuusprotokollat ja ominaisuudet. Lisäksi määriteltiin, millaiset jatkokehitystoimet olisivat tarpeellisia. Työssä käytettiin sulautetun järjestelmän prototyyppimallina Suunto Oy:n kehittämää MoveSense-sensoriteknologiaa, mobiiliohjelmiston koodaamiseen Dart -ohjelmointikieltä ja ohjelmointikielelle kehitettyä Flutter -ohjelmistopakettia. Mobiiliohjelmiston analysointiin ja testaamiseen käytettiin Dart DevTools-testausympäristöä sekä OnePlus 6T Android- ja iPhone 6 iOS -mobiililaitetta.

Työn tuloksia ja kokonaisuutta tarkastellessa voidaan todeta, että kokonaisuudessaan rekikoirille suunnatun aktiivisuusjärjestelmän kehittäminen on mahdollista, mutta vaatii huomattavan määrän jatkokehitystä, työtunteja ja resursseja suunnitteluun, tutkimukseen ja kehittämiseen. Mobiiliohjelmiston testitulokset osoittivat, että ohjelmistoratkaisut ja suunnittelutyö ovat avain hyvään lopputulokseen, kun ollaan luomassa mobiilisovellusta, joka mahdollistaa laajan käyttöympäristön sekä on tehokas ja toimintavarma.

Työn jatkokehityksen kannalta on tärkeää, että ajallisia ja taloudellisia resursseja saadaan lisättyä, jolloin yksilöllisen mittausjärjestelmän valmistaminen on mahdollista ilman kompromissiratkaisuja.

---

Asiasanat: aktiivisuusjärjestelmä, rekikoirat, dart, movesense

## ABSTRACT

Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering
Telecommunications and Networks

SAVOLAINEN, ARI:
Arctic Tracker
Activity System for Sled Dogs

Bachelor's thesis 65 pages, appendices 8 pages
May 2021

---

The main objective of this thesis was to create a complete prototype plan of the activity system for sled dogs and to implement the first prototype of a platform-independent mobile program for the end user. The work was carried out entirely as individual work and was designed to monitor the workload and well-being of sled dogs during their work. The aim of the work was to design a complete activity system plan and an energy efficient prototype for a mobile software that can be integrated into an embedded system individually designed for Arctic extreme conditions.

In this engineering work, the absolute safety protocols, characteristics, and further development opportunities required of an activity system were drawn up drawn up using physical characteristics of sled dogs and the extreme working conditions of sledges, as well as the experiences of sled dog breeders. The prototype of the embedded system was used as a model of MoveSense sensor technology developed by Suunto Oy and the Dart Dev Tools testing environment as well as the Android, iOS device for the analysis and testing of mobile software.

Based on the results of the work, it can be stated that the development of an activity system aimed for sled dogs is possible. Although it requires a considerable amount of further working hours and resources for planning, research, and development. Test results of the mobile software showed that the software solutions and design work are the key to creating a mobile application that enables a wide range of operating environments, is efficient and reliable.

For further development of the work, it is important to enable the required resources for production of an individual measurement system without ready-made compromises for the practical solutions.

Key words: activity system, sled dogs, dart, movesense

**SISÄLLYS**

## LYHENTEET JA TERMIT

| | |
|---|---|
| $I_0$ | Kokonaissäteilyn intensiteetti, W/m$^2$ |
| $I$ | Valon kulkema matka kudoksessa, nm |
| $M$ | Planeetan massa, kg |
| $\alpha$ | Absorptiokerroin tietyllä aallonpituudella, mol$^{-1}$cm$^{-1}$dm$^3$ |
| ECG | Elektrokardiografia |
| ICG | Impedanssikardiografia |
| MEMS | Micro-Electro-Mechanical-Systems |
| PCG | Fonokardiografia |
| PPG | Photoplethysmografia |
| PQRST | Sydänkäyrän kompleksimerkintä |
| RISC | Suoritinarkkitehtuurien suunnittelufilosofia |
| RR | Peräkkäisten sydämenlyöntien välinen aika |
| RX | Vastaanottosignaali, Hz |
| TX | Lähetyssignaali, Hz |

# 1 JOHDANTO

Työ toteutettiin opinnäytetyön tekijän rakkaudesta rekikoiriin ja niiden hyvinvointiin. Tämän opinnäytetyön tavoitteena oli luoda aktiivisuusjärjestelmän arkkitehtuurisuunnitelma ja mobiilisovellusprototyyppi MEMS (Micro-Electro-Mechanical-Systems) -sensoriteknologian tuottaman datan käsittelyyn. Tämä sensoriteknologia mahdollistaa työssä käytettävien rekikoirien paikannuksen sekä niiden aktiivisuuden, kuntotason ja hyvinvoinnin monitoroimisen reaaliaikaisesti. Seurantaa voidaan tehdä työtehtävien ja harjoitusten aikana sekä niiden jälkeen.

Opinnäytetyössä keskityttiin pääsääntöisesti aktiivisuusjärjestelmän ohjelmistokehitykseen, sillä projekti kokonaisuudessaan on useamman henkilövuoden mittainen. Projektin sensoriteknologiana hyödynnettiin Suunto Oy:n ohjelmistokehittäjille tarjoamaa valmista MoveSense-sensorijärjestelmää. Sensoriteknologia ja sen valinta rajatiin opinnäytetyön ulkopuolelle.

Kaikkien rekikoirien kohdalla henkinen ja varsinkin fyysinen rasitustaso työtehtävissä ja kilpailuissa on korkea. Näin ollen koirille, aivan kuten huippu-urheilijoillekin, on suunniteltava jokaiselle kisakaudelle oma harjoitusohjelmansa. Tämä perustuu aloituskuntotasoon, jota lähdetään kehittämään ruokavaliolla, harjoituksien pituudella sekä lepoajoilla, jotta päästään tavoiteltuun kuntotasoon. Suunniteltu aktiivisuusjärjestelmä seuraa rekikoirien työsuoritteiden aikaista rasitusta ja hyvinvointia.

Rekikoirien kuntotasoa tai aktiivisuuden rekisteröimistä ei ole aikaisemmin tehty hyvinvointiteknologiaa hyödyntäen, vaan mittaamista on suoritettu vain visuaalisesti elinkeinokseen ammattia harjoittavien tai aktiiviharrastajien toimesta.

# 2 REKIKOIRAT

Luvussa kaksi perehdytään kattavasti rekikoiriin, niiden historiaan, toimintaympäristöön, sekä fysiologiaan. Rekikoira ei ole mikään tavallinen lemmikkikoira, se on jalostettu työkäyttöön (kuva 1). Opinnäytetyöntekijän harrastepohjan ja kokemusten perusteella laaja-alainen ymmärrys rekikoirista, niiden fysiologiasta ja toimintaympäristöstä on tärkeää, jotta lukija ymmärtää paremmin tässä opinnäytetyössä käsiteltävää aihetta. Opinnäytetyön tekijä pystyi harrastekokemuksensa ansiosta vastaamaan haasteisiin, joita on huomioitava eläinten hyvinvointiteknologiaa suunniteltaessa ja sitä kehitettäessä talviolosuhteisiin.

KUVA 1. Rekikoirat työssä (Bruce 2019).

## 2.1 Määritelmä

Rekikoiriksi yleisemmin mielletään pohjoisen koirarotuja, mutta perinteisesti rekikoirina on käytetty kunkin maantieteellisen alueen omaa koirakantaa. Rekikoirien alkuperäinen käyttötarkoitus on edelleen toimia vetoeläimenä arktisilla alueilla, minne ajoneuvoilla ei ole mahdollista kulkea. Rekikoirista tunnetuin rotu on Siperianhusky, mutta alla on listaus yleisimmistä rekikoirina käytetyistä roduista (Cary, Bob. 2009, 7–21; Leonhard Seppala 2013).

- Samojedinkoira
- Alaskanmalamuutti
- Siperianhusky
- Grönlanninkoira
- Kanadaneskimokoira
- Chinook
- Koilissiperianrekilaika
- Alaskanhusky
- Skandinaviankoira
- Eurasier

## 2.2 Historia lyhyesti

Yksi koiran varhaisimmista käyttötarkoituksista on ollut sen käyttö vetoeläimenä. Tarkasti ei kuitenkaan tiedetä, milloin koiria on alettu ensimmäisen kerran käyttää työskentelemään vetotarkoituksessa. Tämän opinnäytetyön kirjoittamisen aikaan on ollut todistettavasti tiedossa, että eskimot käyttivät rekikoiria jo 1500 vuotta sitten. Toisaalta on myös todisteita siitä, että Euraasian alkuasukkaiden keskuudessa rekikoirien käyttö alkoi jo yli 4000 vuotta sitten. Tuolloin Keski-Aasialaisia kansoja siirtyi kansoittamaan nykyisen Siperian ja muiden arktisten seutujen alueita ja heidän mukanaan kulkeutui pohjoiseen sakaalityyppisiä koiria. Nämä sakaalityyppiset koirat risteytyivät paikallisten susien kanssa. Näistä jälkeläisistä kehittyi ajan saatossa koiria, joita nykyisin kutsutaan alkukantaisiksi roduiksi (Northern Breeds). Näistä ensimmäisistä pohjoisen alkukantaisista koiraesi-isistä kehittyi useita eri luonteenpiirteitä omaavia arktisien koirien sukuhaaroja. (Siperianhusky Valjakkoelämää, 2013, 9–13).

Varsinkin arktisilla alueilla on satoja vuosia käytetty kulkuvälineenä juuri koiravaljakkoa. Metsästäjät, hylkeenpyytäjät, kalastajat, posti, poliisi ja jopa rajavartiolaitos sekä armeija ovat tukeutuneet tähän luotettavaan kulkuvälineeseen, jolla pääsi perille vaikeissakin olosuhteissa (Rekikoirien historia, 2021).

Yksi maailman tunnetuimmista koiravaljakko-ohjastaja legendoista on itse asiassa suomensukuinen Leonhard Seppala (1877–1967), hänen äitinsä oli suomalainen ja isä norjalainen. Seppalan legenda syntyi 1925, kun hänen koiravaljakkonsa näytteli erittäin tärkeää osaa ns. ”Seerumi ajossa”. Nomen kaupungissa Alaskassa oli puhjennut vakava kurkkumätäepidemia vuonna 1925, jonka johdosta kaupunkiin tarvittiin seerumia hillitsemään taudin leviämistä ja pelastamaan erityisesti pieniä lapsia, joille sairaus oli kohtalokas ilman lääkitystä. Tuohon aikaan eivät lentokoneet pystyneet lentämään Nomeen kovan talvikauden aikana, eikä muita kuljetuskeinoja ollut kuin koiravaljakkojen viestiketju. Seppala ja hänen siperianhuskyvaljakkonsa (kuva 2) ajoivat erittäin rankoissa talviolosuhteissa ehdottomasti pisimmän ja vaikeakulkuisimman osuuden koko viestiketjusta (Siperianhusky Valjakkoelämää, 2013, 14–15; Leonhard Seppala 2013).

KUVA 2. Leonhard Seppala ja kuusi hänen rekikoiristaan vasemmalta oikealle, Togo, Karinsky, Jafet, Pete, Ilsa ja Fritz vuonna 1925 (Copyright © Carrie McLain Museum / AlaskaStock).

Rekikoirien henkistä ja fyysistä ihannekuvaa jalostuksen näkökulmasta on muokannut metsästykseen koiralta vaadittavat ominaisuudet. Riistan, joka useimmiten koostui hirvistä ja peuroista, kiinni saamiseksi oli valjakon jokaisen koiran syytä olla hyvin ohjautuva, stressitilanteissa toimintakykynsä säilyttävä, nopea ja kestävä. Rekikoirilla metsästystä on suoritettu vielä pitkään 1900-luvullakin perinteisiä aseita keihästä ja jousta käyttäen. Metsästyssaalis oli usein kuljetettava pitkiä matkoja leiriin tai kylään, ja kun riista päätti siirtyä kauemmaksi, oli metsästäjien muutettava leirinsä riistan mukana (Cary, Bob. 2009, 7–21; Siperianhusky Valjakkoelämää, 2013, 9–10; Rekikoirien historia, 2021).

Nykyisin rekikoiria käytetään pääasiassa harrastus- ja kilpailutoiminnassa erinäisissä valjakkourheilulajeissa ja elämysmatkailussa valjakkosafareita järjestettäessä. Vielä kuitenkin löytyy kansoja kuten eskimot ja tšuktšit, jotka ylläpitävät rekikoiria puhtaasti niiden alkuperäistä tehtävää varten (Cary, Bob. 2009, 7–21; Siperianhusky Valjakkoelämää, 2013, 9–10; Rekikoirien historia, 2021).

## 2.3 Välineet

Tärkeimmät työvälineet rekikoiralle ovat ehdottomasti valjaat, tossut ja takit. Seuraavana tärkeysjärjestyksessä tulevat vasta kaikki muut valjakkourheilussa vaadittavat välineet kuten liinat, ankkurit, reet tai vedettävät kärryt.

### 2.3.1 Valjaat

Valjaiden on istuttava hyvin koiralle, ne on aika ajoin uusittava ja ne ovat verrattavissa huippu-urheilutasolla kilpailevan kestävyysjuoksijan käyttämiin juoksukenkiin. Jos valjaat ovat epäsopivat tai materiaalien valinta valjaissa on väärä, vähentää se rekikoiran työskentelytehokkuutta ja saattaa aiheuttaa koiralle kipua, hiertymiä ja väärään vetoasennon muodostumisen. Valjaisiin suunnitellaan asennettavaksi tämän opinnäytetyön mittaava sensorijärjestelmä (Siperianhusky Valjakkoelämää, 2013, 106).

Valjasmallit, joissa on pitkä selkäosa, kriittiset painepisteet ovat kaulan-, lapojen-, kainaloiden- ja selkäosan-alueella. Näiden kohtien istuvuus on erittäin tärkeää, huolimatta siitä onko valjaiden selkäosan malli X- tai H-mallinen (kuva 3). Kaula-aukko saa tuntua hieman ahtaalta valjaita päälle sovitettaessa, mutta sen on oltava päällä hyvin istuva ja suhteellisen napakka. Kuitenkin niin, että kaula-aukko ei paina koiran henkitorvea, vaan loppuu rintalastan etuosaan, josta valjaan rintapalaosuus jakaa tasaisesti vedosta syntyvän paineen koko rintalastalle. Kaula-aukko ei saa missään tapauksessa osua koiran lapojen päälle, jolloin valjas estää etujalkojen työntymisen liikkeessä eteenpäin (Siperianhusky Valjakkoelämää, 2013, 106; Valjakkovarusteet, 2021).

KUVA 3. Vasemmalla X- ja oikealla H-mallin valjaat.

Rintapaloja on erilaisia, kapeampaa ja leveämpää mallia. Yleisesti ottaen mitä rintavampi koira, sitä leveämpi on rintapalan oltava, jotta vedon luoma paine jakautuu tasaisemmin ja laajemmalle pinta-alalle. Näin ehkäistään rintapalan ”luiskahtaminen” koiran kainaloon työskennellessä, jolloin rintapala saattaa aiheuttaa esimerkiksi hiertymiä ja kipua. Pidemmällä aikavälillä koira oppii juoksemaan vinossa ja näin ollen kehittää itselleen toispuoleisen lihaksiston, mikä lopulta rikkoo koiran fysiikan. Useimmissa valjaissa on käytössä kapea rintapala, sillä se sopii varmemmin useimmille koirille. Rintapalan kuuluu ulottua rintalastan etuosasta lähes rintalastan alakärkeen saakka (Siperianhusky Valjakkoelämää, 2013, 106; Valjakkovarusteet, 2021).

Ennen rintalastan loppua valjas nousee koiran kylkiluiden myötäisesti kohti hännän tyveä. Rintapalan ollessa liian lyhyt koiralle, nousee valjas liian aikaisin ja asettuu näin ollen kainaloihin aiheuttaen koiralle herkästi kipua ja hiertymiä työskentelytilanteissa. Liian pitkä rintapala puolestaan aiheuttaa sen, että valjas

ei nouse enää koiran kylkiluiden kanssa samassa linjassa, vaan painaa helposti koiran vatsaa. Tällaisessa tapauksessa useimmiten myöskin valjaan selkäosa ulottuu liian pitkälle. Oikean mittainen rintapala antaa tilaa koiran liikkeelle, asettuu hyvin ja jakaa vetämisen paineen mahdollisimman tasaisesti rintakehälle ilman, että valjaat olisivat liian ahtaat koiran päällä (Siperianhusky Valjakkoelämää, 2013, 106; Valjakkovarusteet, 2021).

Selkäosan kuuluu loppua hännän tyveen, niin että vetoliinan kiinnityslenkki jatkuu hännän tyvestä eteenpäin. Liian lyhyt tai liian pitkä selkäosa useasti tarkoittaa sitä, että valjas ei istu muiltakaan osin ihanteellisesti koiran päälle. Jos valjakossa on suuresti kooltaan eroavia koiria, kannattaa valjaiden vetoliinan kiinnityslenkeillä säädellä koirien valjaiden pituutta niin, että lyhyemmille koirille on pidemmät vetolenkit kuin pidemmille koirille. Näin koirien paikkojen vaihto valjakossa käy helposti ilman, että vetoliinojen pituuksia tarvitsee muuttaa (Siperianhusky Valjakkoelämää, 2013, 106, Valjakkovarusteet, 2021).

Kaikki valjasmallit eivät sovi kaikille koirille, sillä istuvuuteen vaikuttavia tekijöitä ovat mm. koirantyyppi, rakenne, sijoitus valjakossa, sekä juoksutyyli. Siksi valjasmalleja on useita erilaisia ja uusia kehitellään koko ajan. Valjaita on suunniteltu myös erityyppiseen käyttöön, aina hitaasta kuormavedosta sprinttijuoksuun (Siperianhusky Valjakkoelämää, 2013, 106; Valjakkovarusteet, 2021).

Tärkein muistisääntö valjaita sovitettaessa on, että ne ovat aina yksilölliset ja valjaat on aina sovitettava jokaiselle koiralle erikseen. Valjaita pidetään aina pienessä vedossa sovitusvaiheessa. Löysänä päälle puetut valjaat istuvat aina eri tavalla kuin vedossa. Valjaiden on oltava silloin sopivat, kun koira työskentelee, vedossa valjas pitenee ja asettuu paremmin koiran päälle. Hyvien valjaiden materiaali ei lähtökohtaisesti hierrä tai kerää kosteutta, valjaat ovat kevyet ja pysyvät helposti siistinä (Siperianhusky Valjakkoelämää, 2013, 106, Valjakkovarusteet, 2021).

### 2.3.2 Tossut

Koirien tassut ovat koetuksella etenkin pitkiä matkoja kuljettaessa nopealla vauhdilla. Tossujen (kuva 4) käyttö koirilla suojaa anturoita, kynsiä ja tassuja teräviltä esineiltä, sekä rajulta kulumiselta silloin kun kuljetaan kovilla alustoilla. Tossuja löytyy moneen lähtöön, eri alustoille ja vuodenajoille.

KUVA 4. Dragråttanin V.I.P. kumitossumalli (Tossut, 2021).

## 2.4 Siperianhusky

Siperianhuskyn on keskikokoinen koirarotu. Yleisilmeeltään siperianhusky on tarkkaavainen, ystävällinen, sydämellinen ja hieman hassutteleva. Päälaelta on kallo hieman pyöreähkö, joka kaventuu silmien suuntaan. Otsapenger siperianhuskylla erottuu selkeästi ja kuononselkä ulottuu suorana otsapenkereestä keskipitkän kuonon kärkiosaan saakka. Kuonon malli on keskikokoinen ja kapenee maltillisesti kirsun suuntaan. Kuono ei ole kärjestä terävä tai neliön mallinen. Kirsun väri voi olla musta, maksanruskea tai vaaleanpunainen. Purenta on leikkaava, huulet pigmentoituneet ja tiiviit. Silmät ovat ylöspäin kapenevat ja kärjestä

pyöreät, muutamia asteita vinokulmassa ja keskipitkän matkan päässä toisistaan. Silmien väri voi vaihdella ruskeasta, siniseen, myös sekaväriset silmät hyväksytään. Korvat keskikokoiset ja kolmion muotoiset, korkealle kiinnittyneet ja lähellä toisiaan. Kaula on keskipitkä, kaartuva ja koiran seisoessa ylväästi ylhäällä kannettu. Juoksussa kaula ojentuu hieman eteenpäin (Rotuinfo, 2021).

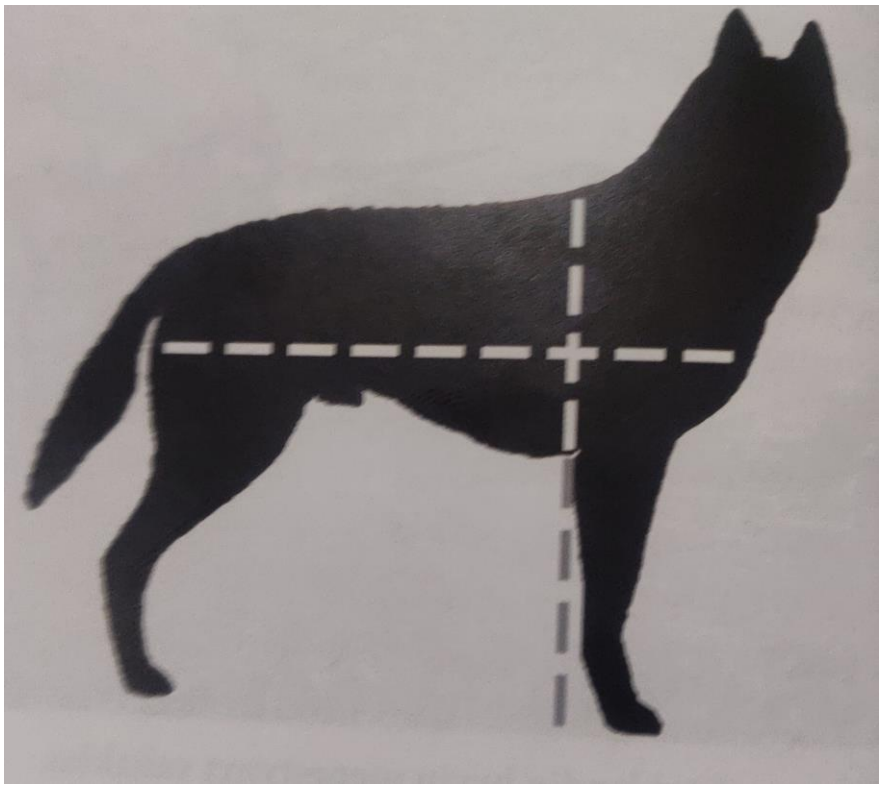
Runko asettuu niin, että keskipitkä selkälinja on vaakasuora ja vahva, ei missään nimessä tukeva tai lyhyt. Lanne on rintakehää kapeampi, kiinteä ja sulava linjainen, lantion asettuu hieman viistoon, ei kuitenkaan koskaan jyrkästi niin, että lantion asento rajoittaisi voimantuottoa takaraajoille. Rintakehä kookas ja voimakasrakenteinen, jonka syvyys ulottuu kyynärvarsien tasalle tai hieman niiden taakse. Kylkiluiden kaartuvuus selkärangasta alkaa sulavasti, kaarevuuden vähetessä kyljissä mahdollistaen vapaat liikkeet. Häntä tuuheakarvainen ja hieman ketun hännän muotoinen, asento selän päällä kauniilla sirpillä (Rotuinfo, 2021).

Eturaajat ovat koiran seisoessa kohtuullisen etäällä toisistaan, mutta yhdensuuntaiset ja suorassa. Tiivis luusto ei ole koskaan raskasrakenteinen. Lapojen sijainti on takana ja hieman viistossa, näyttävästi kehittynyt lihaksisto ja jänteiden kiinnityskohdat sitovat eturaajat tiiviisti rintakehänalueeseen. Kyynärpäät ovat tiiviit, rungon myötäiset maksimaalisen voiman tuottamiseen, eivätkä saa olla sisä- tai ulkokierteiset. Ranteiden on oltava voimakkaat ja joustavat. Takaraajojen sijainti on hieman erillään toisistaan ja samansuuntaiset, reisienalueelta lihaksikkaat ja vahvat, polvien suunta hyvässä kulmassa muuhun vartaloon nähden maksimaalisen voiman tuottamiseksi. Liikkuminen on höyhenen kevyttä ja helppoa, ja koiran liikkeet ovat nopeat ja sulavat (Rotuinfo, 2021).

Karvapeite on kaksinkertainen ja keskipitkä. Pohjavilla on pehmeää ja tiheää tukien pintaturkkia. Pintaturkki aaltoilee suorana rungon myötäisesti, ei missään nimessä karkeaa tai piikikkäästi ulospäin suuntautuvaa. Karvapeitteen kaikki värit ja kuviot ovat sallittuja, sillä tällä ominaisuudella ei ole vaikutusta koirarodun fyysiseen suoriutumiseen. Urosten säkäkorkeus on noin 53-–60 cm ja paino 20–27 kg. Nartuilla säkäkorkeus on noin 51–56 cm ja paino 16–23 kg. Mitat ovat näennäisiä, sillä paino on aina suhteessa korkeuteen ja tärkeintä onkin katsoa koiraa sopusuhtaisena kokonaisuutena, jolloin sillä on tasapainoiset mittasuhteet, vapaat ja vaivattomat liikkeet (Rotuinfo, 2021).

### 2.4.1 Mittasuhteet

Siperianhusky kuvassa kyljensuuntaisesti, pituus mitattaan olkanivelen kohdasta aina istuinluun kärkiosaan. Rungon kokonaispituus on hieman säkäkorkeutta pidempi (kuva 5).



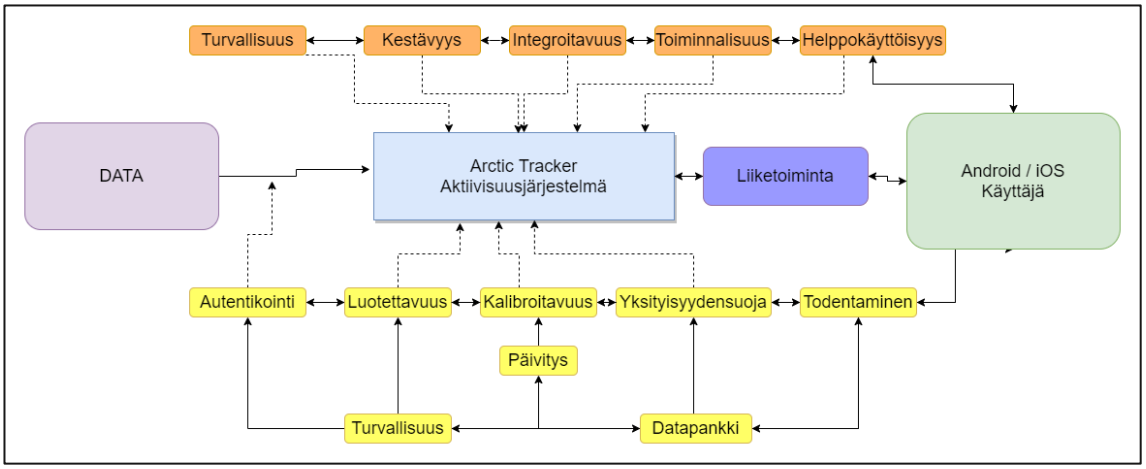
KUVA 5. Mittasuhteita havainnollistava kuva (Siperianhusky valjakkoelämää).

# 3 ARKKITEHTUURISUUNNITELMA

Luvussa kolme esitellään järjestelmä arkkitehtuurisuunnitelma kokonaisuudessaan, arkkitehtuurisuunnitelma toimii yleisrakenteellisena dokumentaationa tehdylle työlle. Arkkitehtuurin tarkoitus on toimia perustana tulevaisuudessa tehtävälle järjestelmän jatkokehitystöille ja toteutusratkaisuille, sekä edistää järjestelmän ymmärrettävyyttä, ylläpidettävyyttä, skaalautuvuutta ja laajennusmahdollisuuksia.

## 3.1 Arkkitehtuurin vaatimukset

Viime vuosikymmenten teknologinen kehitys, varsinkin anturiteknologiassa, on tuonut hyvinvointiteknologia-alalle uusia tuotteita ja palveluita, jotka ovat tähän saakka painottuneet lähinnä ihmisten hyvinvoinnin mittaamiseen. Samaa teknologiaa voidaan hyödyntää myös työkoirien hyvinvoinnin mittaamiseen. Toimintaympäristö minne anturijärjestelmä suunnitellaan, on täynnä luonnon äärioloisuhteita rankimmillaan. Näin ollen, kokonaisarkkitehtuurin vaatimuksia ja toiminnallisuutta oli mietittävä huolella. Tässä opinnäytetyössä päädyttiin tekemään kattava ja yksityiskohtainen lista vaatimuksista, joita järjestelmän kehittäminen ja jatkokehittäminen vaativat. Nämä vaatimukset on jaettu kahteen eri kategoriaan, toiminnalliseen ja ei toiminnalliseen kategoriaan (kuva 6).




KUVA 6. Arkkitehtuurin tärkeimmät vaatimukset kaavio muodossa.

### 3.1.1 Ei-toiminnalliset vaatimukset

- Autentikointi: Antureilta saadun datan todentaminen niin, että voidaan 100 % varmuudella luottaa järjestelmään saatu data on oikeellista, sekä vain ja ainoastaan juuri kyseisen järjestelmän käyttäjälle tarkoitettua dataa.

- Yksityisyydensuoja: Kerätyn datan pitää anonymiteettisuojattua ja ainoastaan loppukäyttäjän on mahdollista kohdistaa halutessaan kaikki kerätty data yksilötasolla. Koskee myös sijaintidataa ja muita tietosuojalain määräyksiä.

- Todentaminen: On pystyttävä varmistamaan, että järjestelmän käyttäjä on sen käyttäjätilin omistaja, kuka hän väittää olevansa ja että ulkopuolisilla ei ole pääsyä hänen käyttäjätilinsä tietoihin tai niiden muokkaamiseen.

- Turvallisuus: Järjestelmä on tietoturvattu niin, että ulkopuolisilla ei ole pääsyä järjestelmädataan.

- Kalibroitavuus: Anturijärjestelmän kalibrointi on oltava mahdollista. Esimerkiksi jos saadun datan todenperäisyyttä saatetaan kyseenalaistaa tai jos käyttöympäristössä tapahtuvat muutokset sitä vaativat.

- Luotettavuus: Järjestelmä toimii niin kuin on suunniteltu ja tuottaa käyttäjälle oikeellista ja luotettavaa mittausdataa perustuen eläinlääketieteelliseen tietotaitoon.

### 3.1.2 Toiminnalliset vaatimukset

- Turvallisuus: Siitä näkökulmasta, että mitään fyysistä tai henkistä vahinkoa ei saa koitua koiralle järjestelmää käytettäessä.

- Kestävyys: Kriteerinä on, että laitteisto säilyttää toimintakykynsä ja tuottaa oikeellista dataa erilaisissa ääriolosuhteissa, joihin liittyy erityisesti suuret lämpötilavaihtelut, kosteus ja likaisuus.

- Integroitavuus: Järjestelmän pitää olla mahdollista implementoida koiran tarpeiden mukaan eri mallisiin ja tyylisiin työskentely valjaisiin.

- Toiminnallisuus: Käyttäjällä on mahdollisuus itse määrittää vapaasti toiminnallisuuksia, joita hän haluaa käyttää ja millaista dataa hän haluaa järjestelmästä saada.

- Helppokäyttöisyys: Käyttöliittymän osalta mahdollisimman intuitiivinen ja miellyttävä käyttää. Käyttäjällä mahdollisuus personoimiseen.

### 3.2 Laitteistoarkkitehtuuri

Tässä opinnäytetyössä päädyttiin käyttämään Suunto Oy:n valmistamaa Move-Sense-sensoriteknologiaa, joka perustuu MEMS-teknologiaan ja jota käytetään mm. Suunto Oy:n valmistamissa sykevöissä ja se soveltuu myös kosteisiin ja talvisiin olosuhteisiin.

Optimaalinen sensorivalinta ei ole, sillä sensoria ei ole kehitetty varsinkaan talvisille ääriolosuhteille, jossa erämaa on toimintaympäristönä. Tämän sensorin valintaan päädyttiin useammastakin syystä. Vaikuttavimmat tekijät kuitenkin olivat työhön käytettävän ajan ja resurssien puitteet. Jolloin hyväksyttiin se, että ensimmäinen prototyyppi voidaan rakentaa Suunto Oy:n sensoriteknologiaa hyödyntäen, sillä onhan yritys kansainvälisesti tunnettu luotettavista laitteistaan, sekä niiden monipuolisesta valikoimasta.

### 3.2.1 MEMS-sensorit

MEMS on teknologiaa, jonka voidaan yleisemmin käsittää koostuvan mikrokokoisista mekaanisista ja elektronisista komponenteista. Komponentit valmistetaan mikrotuotantona. MEMS-laitteet voivat vaihdella fyysiseltä kooltaan alle yhdestä mikrometristä aina useaan millimetriin ja niiden rakenne voi vaihdella myöskin melko yksinkertaisesta rakenteesta, jossa ei ole liikkuvia osia, aina äärimmäisen monimutkaisiin sähköelektroniikan järjestelmiin. Järjestelmissä on useita liikkuvia osia, joita ohjataan integroidulla mikroelektroniikka piirillä. Yksi MEMS määrittelyn pääkriteereitä on, että sensorin jossakin osassa on jonkinlainen mekaaninen toiminto riippumatta siitä, liikkuuko osa tai ei (MEMS & Nanotechnology Exchange, 2021).

MEMS-teknologiaa käytetään hyvin yleisesti laitteissa, joilla halutaan mitata esimerkiksi, lämpötilaa, painetta, kiihtyvyyttä, magneettikenttää, säteilyä, nesteen ja kaasujen virtausta tai tunnistaa eri kemikaaleja. Kokonsa puolesta MEMS-laitteet ovat edullisia valmistaa, niitä voidaan soveltaa lukemattomiin käyttötarkoituksiin ja ympäristöihin (MEMS & Nanotechnology Exchange, 2021).

### 3.2.2 Kiihtyvyysanturi

Kiihtyvyysanturi on mittauslaite, joka havaitsee kiihtyvyyden muutoksia, liikkeessä, tärinässä tai iskuissa. Kiihtyvyys *a* on fysikaalinen vektorisuure ja tarkoittaa nopeuden *v* muutosnopeutta ajan *t* funktiona (kaava 1), sen yksikkönä SI-järjestelmässä on m/s$^2$ (Wilson, J. 2004, 137; SI-opas, 2019, 23).

$$a = \frac{dv}{dt} \tag{1}$$

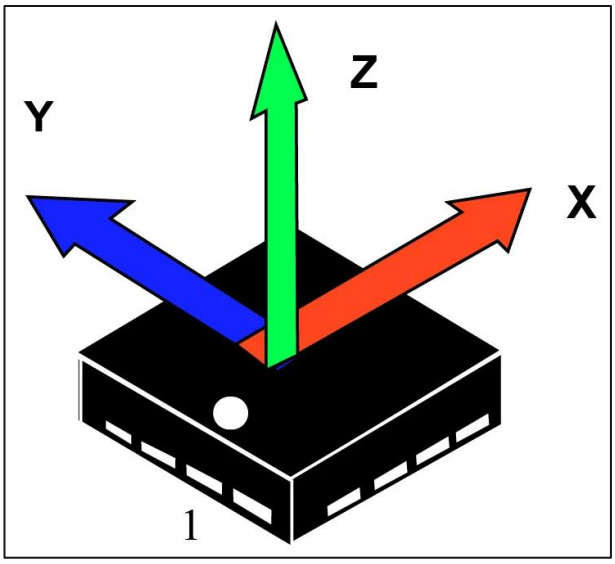
Lepotilassa kiihtyvyysanturi mittaa putoamiskiihtyvyyttä *g* (kaava 2) antaen mittausarvon noin 1. Suure ilmaisee kappaleen kiihtyvyyttä *a* suhteessa maan vetovoimaan nähden. Putoamiskiihtyvyydelle voidaan planeetan pinnalla laskea likimain oikea arvo käyttäen kaavaa (2):

$$g(R) \approx G\frac{M}{R^2} \quad (2)$$

jossa *R* on etäisyys planeetan keskipisteestä, *M* planeetan massa ja *G* gravitaatiovakio (Wilson, J. 2004, 137–138; SI-opas, 2019, 23–29).

Maan vetovoima ei ole todellisuudessa vakio, vaan vaihtelee leveysasteittain maapallon pyörimisliikkeen ja litistyneisyyden seurauksena. Putoamiskiihtyvyydelle on kuitenkin määritetty yleispätevä kansainvälinen standardi normaaliputoamiskiihtyvyys $g_n$ 1 g = 9.80665 m/s$^2$, jota käytettiin tämän opinnäytetyön yhteydessä (SI-opas, 2019, 23–29).

Kiihtyvyysantureita on nykyisin käytössä useissa eri sovelluksissa, on kyseessä sitten tekninen tutkimustyö, mittauskehitystyö tai kuluttajille suunnattu arkikäyttö kuten esimerkiksi älypuhelimet, urheilukellot, ajoneuvojen turvatyyny- tai hälytysjärjestelmät. Kiihtyvyyden mittaamiseen käytetyt anturit ovat kahden (*x, y*) tai kolmen (kuva 7) toisiaan kohtisuoraan mitattavan akselin omaavia, jolloin mittaamisen suunta ilmoitetaan joko negatiivisena tai positiivisena kiihtyvyyden suunnasta riippuen. 3-akseliset kiihtyvyysanturit mahdollistavat mittaamisen pysty-, sivuttais- ja eteen-suunnassa (Wilson, J. 2004, 137; ASM330LHH. 2020).




KUVA 7. 3-akselinen kiihtyvyysanturi (ASM330LHH).

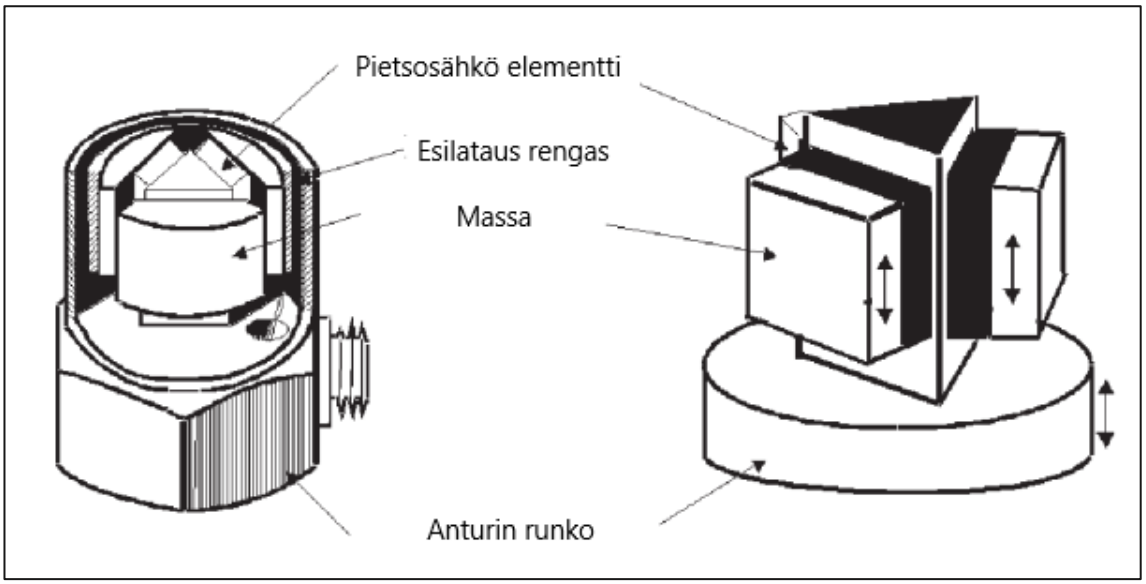
On olemassa useita eri tekniikoilla valmistettuja kiihtyvyysantureita, kuten pietsosähkö-, pietsoresitiivi-, kapasitiivi- tai servotekniikka-antureita. Näistä yleisin käytössä oleva on pietsosähköllä toimiva kiihtyvyysanturi, sillä kyseisellä tekniikalla valmistetulla kiihtyvyysanturilla on mahdollista mitata laajalla skaalalla objektien dynaamisia ominaisuuksia. Pietsosähköanturi omaa erinomaisen kestävyyden verrattuna muilla tekniikoilla valmistettuihin antureihin. Anturilla on korkea herkkyystaso ja se sietää suuriakin lämpötilavaihteluita hyvin. Kiihtyvyysanturi on yksinkertainen rakenteeltaan ja edullinen, sekä omaa erittäin alhaisen sähkönkulutuksen (Wilson, J. 2004, 137–138).

Pietsosähköinen kiihtyvyysanturi on mekaanisesti suunniteltu niin, että se tuottaa suurimman osan käyttämästään sähköstä itse. Anturille tunnusomaisia ominaisuuksia ovat myös laaja ja tasainen taajuusvastealue, sekä suuri lineaarinen amplitudialue. Anturin mekaaninen toiminta perustuu materiaalin polarisoitumiseen, joka perustuu Newtonin 2.lakiin (kaava 3) eli dynamiikan peruslakiin (Wilson, J. 2004, 137–138; SI-opas, 2019, 23–29)

$$\vec{F} = m\vec{a} \tag{3}$$

jossa *F* on voima, *m* on massa ja *a* on kiihtyvyys.

Pietsosähkö kiihtyvyysanturin (kuva 8) materiaalissa tapahtuu mekaanisen jännityksen aiheuttaman ilmiön, joka ilmenee sähköisenä jännitteenä. Tätä kutsutaan polarisoitumiseksi. Näin ollen anturissa voiman *F* kasvaessa myös ulostulojännite kasvaa (Wilson, J. 2004, 137–138; SI-opas, 2019, 23–29).




KUVA 8. Perinteinen pietsosähkö kiihtyvyysanturi (Wilson, J. 2004, 138).

Opinnäytetyössä käytetty kiihtyvyysanturi on 3-akselinen pietsosähköanturi ja toimii niin, että positiivinen kiihtyvyys ilmaisee ylös, oikealle, eteen suuntautuvaa kiihtyvyyttä ja negatiivinen kiihtyvyys alas, vasemmalle, taakse suuntautuvaa kiihtyvyyttä. Kiihtyvyyden mittaamisella pyritään tutkimaan ja ymmärtämään objektien dynaamisia ominaisuuksia paremmin (Wilson, J. 2004, 137; IIS3DHHC. 2020).

### 3.2.3 Gyroskooppianturi

Gyroskooppi rakentuu pyörivästä kiekosta, joka on ripustettu liikkuvan akselikehikon sisäpuolelle. Kehikon liikkuminen ei vaikuta kiekon asentoon tai pyörimiseen, vaan kiekko pyrkii säilyttämään siihen vaikuttavien voimien alkuperäisen suunnan riippumatta siihen kohdistuvasta keskipakovoimasta $F_c$. Keskipakovoima lasketaan käyttäen kaavaa 4 (Sazonov, E. & Neuman, M. 2014, 2.2; SI-opas, 2019, 22–30).

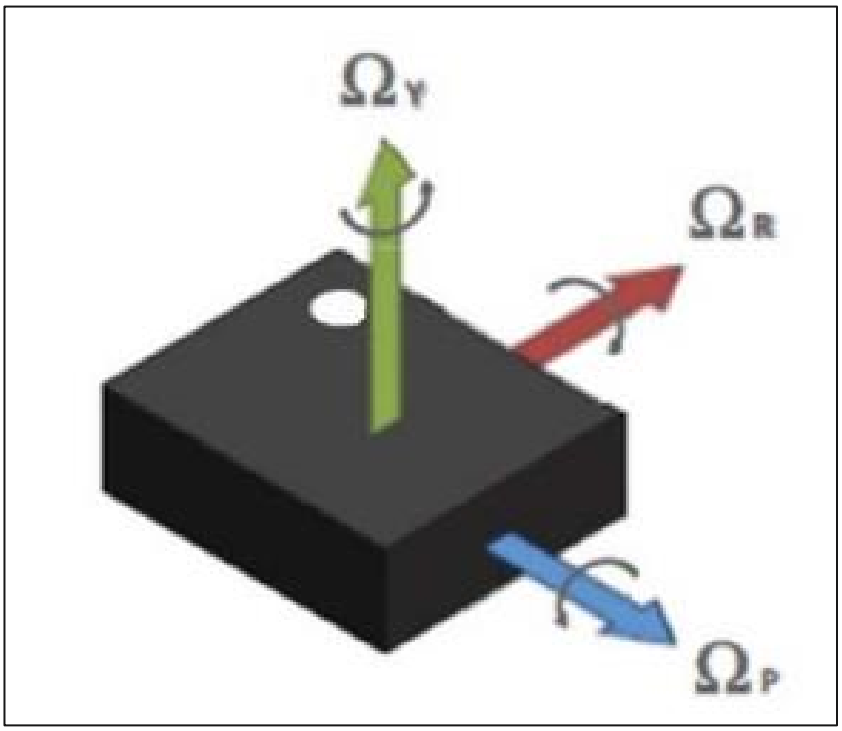
$$Fc = m\omega^2 r \tag{4}$$

missä $m$ on massa, $\omega$ on pyörimisliikkeen kulmanopeus ja $r$ kiertoradan säde.

Akselikehikon suunnan muuttuessa, syntyy kaltevuusakselin pyörimisnopeuteen verrannollinen momentti, jota voidaan käyttää kulmanopeuden havainnollistamiseen. Esimerkkinä kulmanopeuden *ω* mittaamiseen käytetystä anturista (kuva 9), toimii dynaamisesti viritetty gyroskooppi (Sazonov, E. & Neuman, M. 2014, 2.2; SI-opas, 2019, 22–30). Pyörivän tai kiertävän kappaleen kulmanopeus rad/s voidaan laskea kaavalla 5, joka ilmaisee kulmanopeuden tietyllä ajan hetkellä seuraavasti:

$$\omega(t) = \frac{d\varphi}{dt} \tag{5}$$

missä *φ* on *φ(t)* eli kiertokulma ajan funktiona ja *t* on aika sekunteina (SI-opas, 2019, 22–30).




KUVA 9. 3-akselinen gyroskooppi. (ASM330LHH).

Akselinsa ympäri pyörivän kiekon tai renkaan kulmanopeus *ω* voidaan laskea (kaava 6) yksinkertaisemmin, kun tiedossa on kappaleen kehänopeuden *v* ja säde *r* (Sazonov, E. & Neuman, M. 2014, 2.2; SI-opas, 2019, 22–30).

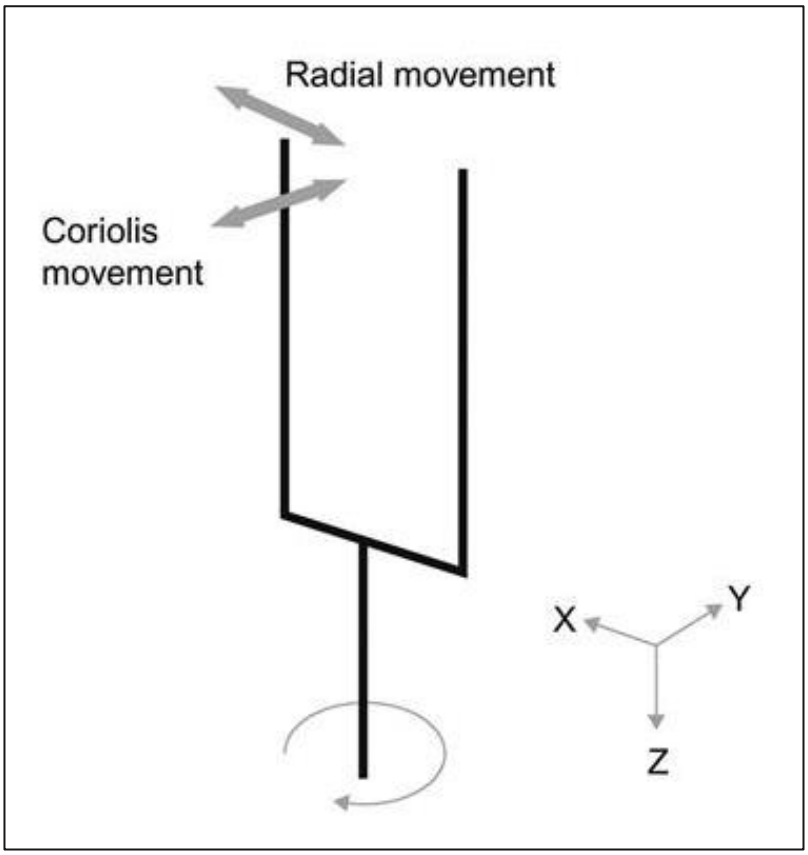
$$\omega = \frac{v}{r} \tag{6}$$

Tyypillisesti MEMS -gyroskooppia käytetään kiihtyvyysantureissa, jotka mittaavat värähtelyä ja Coriolis -ilmiötä, joka voidaan esittää myös Coriolis -voimana (kuva 10) $\vec{F}c$ seuraavasti kaavalla 7 (Sazonov, E. & Neuman, M. 2014, 2.2):

$$\vec{F}c = 2m(\vec{v} \times \vec{\omega}), \tag{7}$$

jossa *m* on kappaleen massa, $\vec{v}$ on nopeus pyörivän pinnan suhteen, *x* tarkoittaa vektorien ristituloa ja $\vec{\omega}$ on pyörivän pinnan kulmanopeutta (Thornton, S. & Marion, J. 2004, 391–393).

Perinteinen värähtelygyroskoopin massa on jousitettu, jotta liike kahteen kohtisuoraan suuntaan on mahdollista. Coriolis -voiman (kuva 10) mittaamiseksi on massan oltava liikkeessä, joten massa pakotetaan elektronisesti liikkeeseen piirin tasopinnan suuntaisesti. Gyroskooppipiiriä pyöritettäessä kohtisuoran akselinsa ympäri, saa Coriolis -voima massan taipumaan vastakkaiseen suuntaan tietyllä momentilla, tämä on suoraan verrannollinen resistanssiin *R*. Värähtelytaipumisen amplitudi on suoraan verrannollinen pyörimisnopeuteen, jota voidaan mitata käyttäen kapasitiivista menetelmää muuntamalla ulostulojännite pyörimisnopeudeksi (Sazonov, E. & Neuman, M. 2014, 2.2).



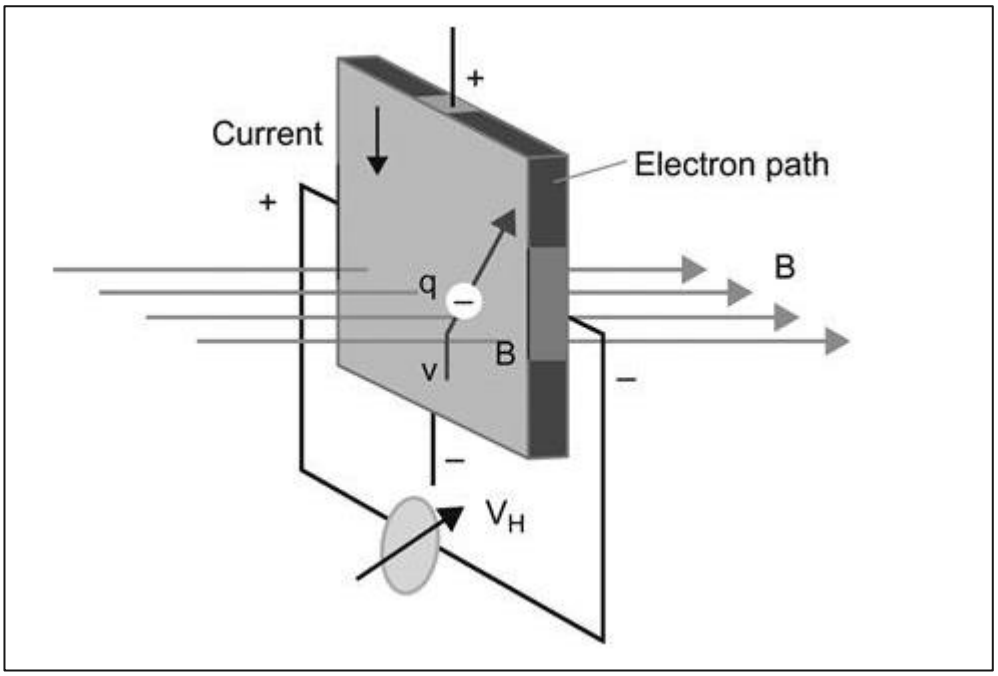

KUVA 10. Coriolis -voima gyroskoopissa (Sazonov, E. & Neuman, M. 2014, 2.2).

### 3.2.4 Magnetometri

Magnetometri voidaan rakentaa usealla eri tavalla ja niitä valmistetaan usealla eri herkkyysasteella. Yleisin kaupallisessa käytössä oleva on magnetometri, joka perustuu Hall -ilmiöön (kuva 11) ferromagneettisessa kelassa, tuottaen magneettisen impedanssin ja magneettisen vastuksen kantoaaltojen ja ulkoisen magneettikentän välisen vuorovaikutuksen seurauksena. Anturi siis mittaa magneettista voimaa *F,* joka vaikuttaa elektronin kulkiessa magneettikentän *B* läpi, joka voidaan laskea käyttäen kaavaa 8 seuraavasti:

$$F = qvB, \tag{8}$$

jossa *q* on alkeisvaraus, *v* on elektronin nopeus ja *B* on magneettikentän voimakkuus (Wilson, J. 2004, 232–233; Sazonov, E. & Neuman, M. 2014, 2.2; SI-opas, 2019, 42–50).




KUVA 11. Hall -ilmiön kuvaus (Sazonov, E. & Neuman, M. 2014, 2.2).

Ulkoisen magneettikentän puuttuessa, virta syötetään kelan läpi muodostaen magneettikentän, polarisoiden ferromagneettisen materiaalin luoden hystereesin, jossa materiaalin magneettiset momentit järjestäytyvät samaan suuntaan (Wilson, J. 2004, 232–233).

Magnetometrillä voidaan siis mitata magneettikenttiä ja niiden vaihtelevuutta, sähkövirtojen luomia sähkökenttiä ja lämpötilaa. Suunnon MoveSense-anturijärjestelmän magnetometrissa on oma sisäinen lämpötila-anturi, jota voidaan käyttää kehon lämpötilan mittaamiseen. Lämpötila-anturi soveltuu lämpötila muutosten (*ΔT)* mittaamiseen 0ºC – +65ºC ±0.5ºC tarkkuudella ja -40ºC – +85ºC ±1.0ºC tarkkuudella. Lämpötila-anturin taattu toiminta-alue lämpötilan puolesta on -40ºC – +85ºC. Käyttölämpötila ei ole optimaalinen tässä opinnäytetyössä tehtävän järjestelmän kannalta, sillä järjestelmän tulisi säilyä toimintakykyisenä myöskin todellisissa ääriolosuhteissa kuten -50ºC – -60ºC, joissa rekikoirien kanssa ajoittain saatetaan joutua toimimaan (AN5192. 2020; MoveSense sensor. 2021).

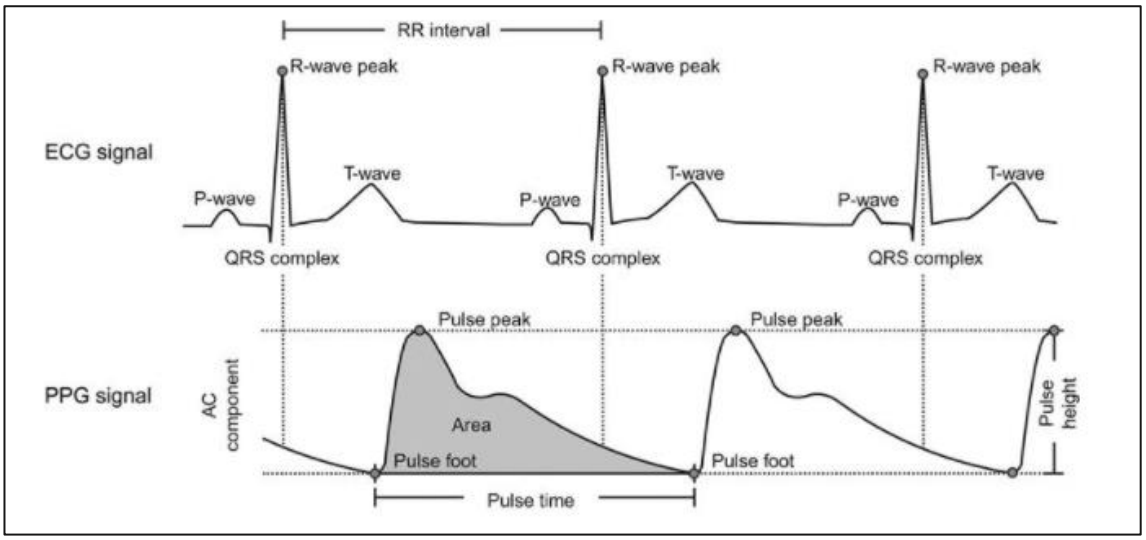
Näitä ominaisuuksia hyväksi käyttäen on siis mahdollista mitata esimerkiksi, onko koiran vasemman tai oikean kyljen raajoissa liikkuvuus- tai tehollisuuseroja harjoitus- ja työskentelytilanteissa.

### 3.2.5 Sykemittari

Mittaamalla sydämen sykettä, on mahdollista analysoida ihmisen tai eläimen kehon reaktioita harjoitteisiin tai harjoituksen tuottamiin stressireaktioihin, jolloin voidaan systemaattisesti arvioida mittavan kohteen sen hetkistä kuntotasoa. Sydämen sykkeen mittaaminen on mahdollista usealla eri tekniikalla, joista yleisimmät ovat elektrokardiografia (ECG), fonokardiografia (PCG), impedanssikardiokrafia (ICG) ja photoplethysmografia (PPG). Tässä kappaleessa käydään pääasiallisesti läpi ainoastaan ECG- ja PPG-tekniikkaa, jotka ovat käytössä MoveSense -anturijärjestelmässä (Sazonov, E. & Neuman, M. 2014, 2.3).

Sydän on lihas, joka vastaa veren kierrättämisestä kehon verisuonistossa. Valtimoita pitkin liikkuu happipitoinen veri sydämestä poispäin kuljettaen happea ja ravintoaineita kehon kudoksille, ja laskimot kuljettavat vähähappisen veren kudoksesta takaisin sydämeen. Sydämen sykkeen säätelystä vastaa pääasiallisesti autonominen hermosto, joka koostuu sympaattisista ja parasympaattisista toisiaan täydentävistä toiminnoista. Sympaattinen hermosto huolehtii kehon stressivasteista, kuten sydämen sykkeen kohoamisesta ja verisuonten supistumisesta. Parasympaattinen hermosto puolestaan huolehtii ja edistää kehon ylläpitoa levossa mm. hidastaen sydämen sykettä ja laajentaen verisuonia (Sazonov, E. & Neuman, M. 2014, 2.3).

Elektrokardiografia (ECG) on biopotentiaalinen tekniikka, joka seuraa sydämen sähköistä aktiivisuutta. Sydänsolujen supistumis- ja rentoutumisjakso voidaan liittää sähköistä potentiaalia lisääviin ja sähköistä potentiaalia pienentäviin jaksoihin. Nämä jaksot luovat paikallisia sähköisiä dipoleja, jotka aiheuttavat pintapotentiaalia rintakehän tietyillä osa-alueilla, jota ECG-tekniikka mittaa. Terveen sydämen ECG-signaali voidaan tunnistaa viiveestä sähköisen potentiaalin lisääntymisen ja sähköisen potentiaalin pienentymisen välillä, joka tuottaa tyypillisiä PQRST-aaltoja (kuva 12). Sydämen syke ilmaistaan yleisesti lyönteinä minuutissa, mutta kyseinen arvo on kuitenkin johdettu sydämen peräkkäisestä sykevälistä millisekunteina ilmaistuna, joka ECG-signaaleilla mitataan kahden R-aallonhuipun välisenä viiveenä. Tätä kutsutaan RR-aikaväliksi (Sazonov, E. & Neuman, M. 2014, 2.3).

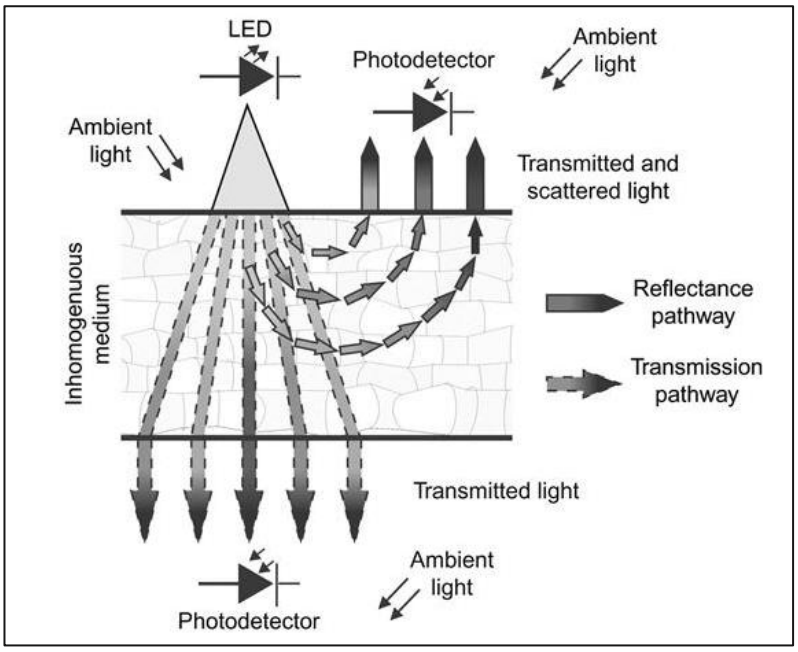



KUVA 12. Synkronoidut ECG- ja PPG-aaltomuodot vertailtuna (Sazonov, E. & Neuman, M. 2014, 2.3).

Photoplethysmografia (PPG) on puolestaan optinen tekniikka (kuva 13), joka mittaa kudoksessa tapahtuvia valon etenemisnopeuden muutoksia sydämenlyöntisyklin aikana. Tämä voidaan laskea Beer-Lambertin lain mukaan kaavalla (9) seuraavasti:

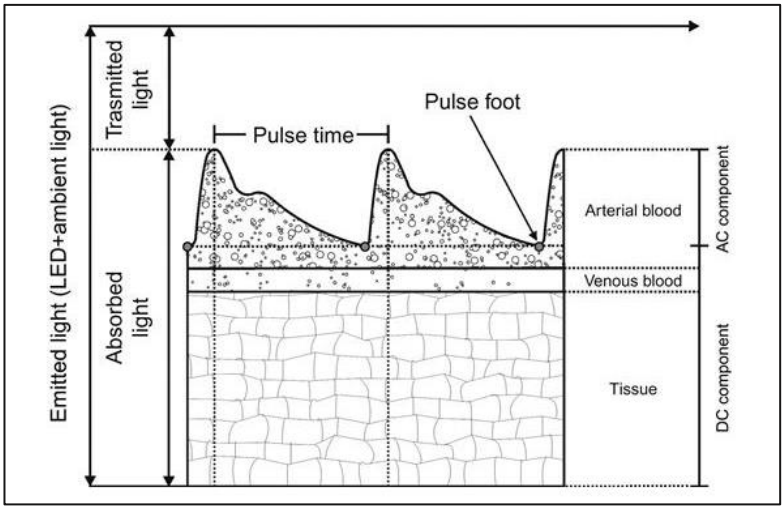
$$I = I_0 e^{-\alpha l}, \tag{9}$$

jossa $I$ on läpi päässeen säteilyn intesiteetti, $I_0$ on alkuperäisen säteilyn intesiteetti, $\alpha$ on valon absorptiokerrointa tietyllä aallonpituudella ja $l$ on valon kulkema matka kudoksessa (Sazonov, E. & Neuman, M. 2014, 2.3).




KUVA 13. PPG-tekniikan yleiskuvaus (Sazonov, E. & Neuman, M. 2014, 2.3).

Sovellusta käytetään happisaturaation seurantaan. Tätä varten käytetään kahta eri aallonpituudella esiintyvää valoa, joiden perusteella voidaan arvioida valtimoveren absorbanssia liittyen veren hapettumistasoon (kuva 14), sekä sydämen sykettä. Tätä mitattaessa voidaan tarkkailla mikrovaskulaarisen kerroksen tilavuusmuutoksia, jolla saadaan tietoa valtimoiden pulssiherkkyydestä (Sazonov, E. & Neuman, M. 2014, 2.3).




KUVA 14. Yksinkertaistettu PPG-signaali (Sazonov, E. & Neuman, M. 2014, 2.3).

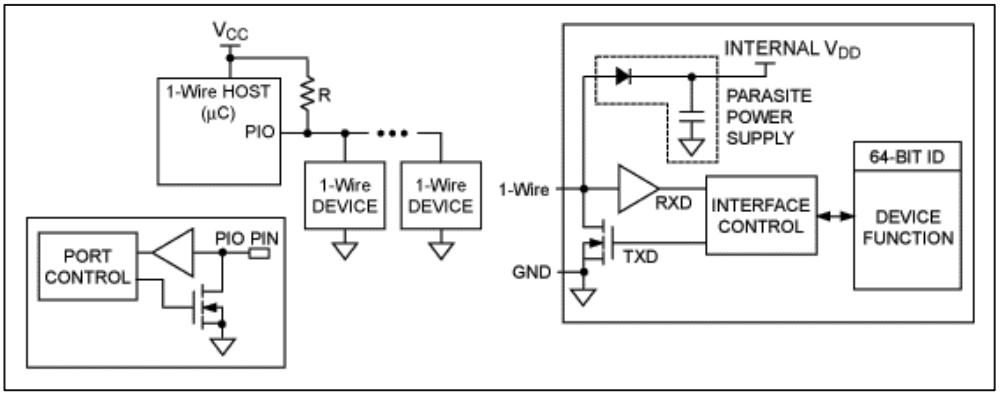
ECG-tekniikan ja PPG-tekniikan hyödyntäminen tämän opinnäytetyön yhteydessä voidaan perustella erityisesti sillä, että ne mahdollistavat mittauspisteiden

valinnan huomattavasti vapaammin, kuin mitä muita tekniikoita käyttäen olisi mahdollista.

### 3.2.6 1-Wire tiedonsiirto

1-Wire on Dallas Semiconductor Oy:n (nykyisin Maxim Oy) kehittämä hidas tiedonsiirronväylä, väylän tiedonsiirtonopeus on 16.3 kbit/s. 1-Wire muistuttaa konseptiltaan $I^2C$ -tiedonsiirtoväylää (Inter-Integrated Circuit), jonka kehitti Philips Semiconductor Oy (nykyisin NXP Semiconductors Oy) vuonna 1982, sillä erotuksella, että 1-Wiren virrankulutus ja tiedonsiirto nopeus ovat alhaisempia, mutta kantama pidempi kuin $I^2C$ -tiedonsiirtoväylällä. Väylän tyypillinen käyttötarkoitus soveltuukin parhaiten ohjaamaan yksikertaisia, vähänvirtaisia laitteita, kuten kiihtyvyysanturit, gyroskoopit, magnetometrit, lämpöanturit tms. (Guide to 1-Wire, 2008; Reading and Writing 1-Wire, 2009).

1-Wire tiedonsiirtoväylän erikoisuutena voidaan mainita, että se tarvitsee vain kaksi johdinta tiedonsiirtoon. Yhden johtimen datalinjalle ja toisen johtimen maapotentiaaliksi. Tiedonsiirtoväylästä virta varataan kondensaattorin, jonka virralla kytketty laite toimii tiedonsiirtoliikenteen ajan. Itse kaksisuuntainen tiedonsiirtoväylä toimii päälaite- ja orjalaite-periaatteella, jolloin päälaite ohjaa muita siihen kytkettyjä 1-Wire-laitteita (kuva 15). Jokaisella 1-Wire orjalaitteella on yksilöllinen, ei muutettavissa oleva, tehtaalla ohjelmoitu 64-bittinen tunnistusnumero (ID), joka toimii laitetunnisteena 1-Wire tiedonsiirtoväylässä (Guide to 1-Wire, 2008).

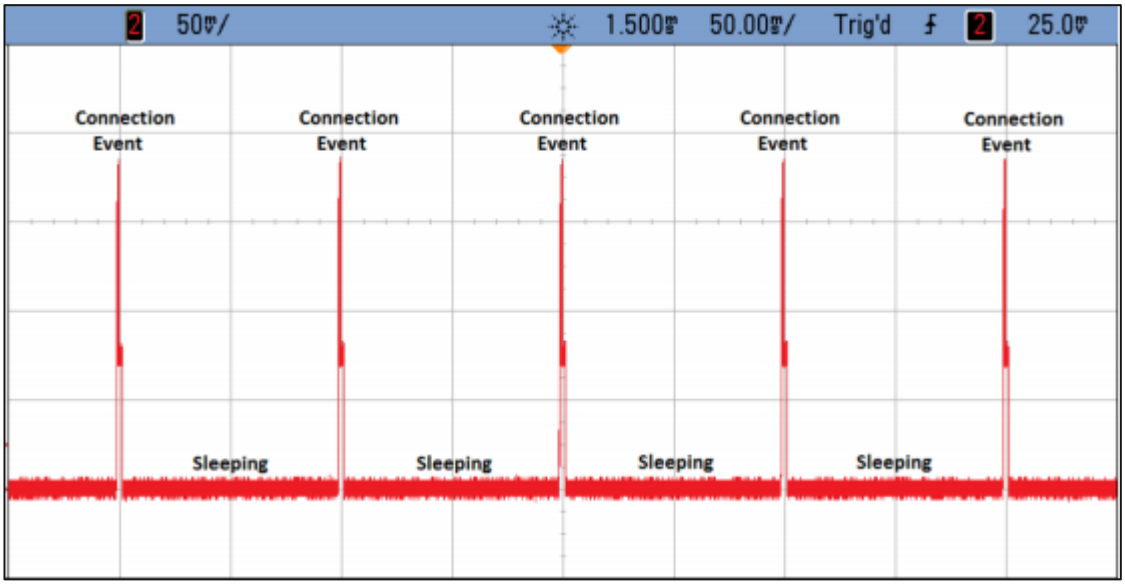



KUVA 15. 1-Wire päälaite- / orjalaite-periaate piirikaaviossa (Guide to 1-Wire, 2008).

1-Wiren on osa MoveSense-anturiyksikköä. 1-Wiren yksinkertainen rakenne, toimintavarmuus, vähäinen sähkönkulutus, sisäinen muistiyksikkö ja turvallisuus tekevät siitä sopivan valinnan anturiyksikön käyttötarkoituksen huomioon ottaen.

### 3.2.7 Bluetooth 4.0

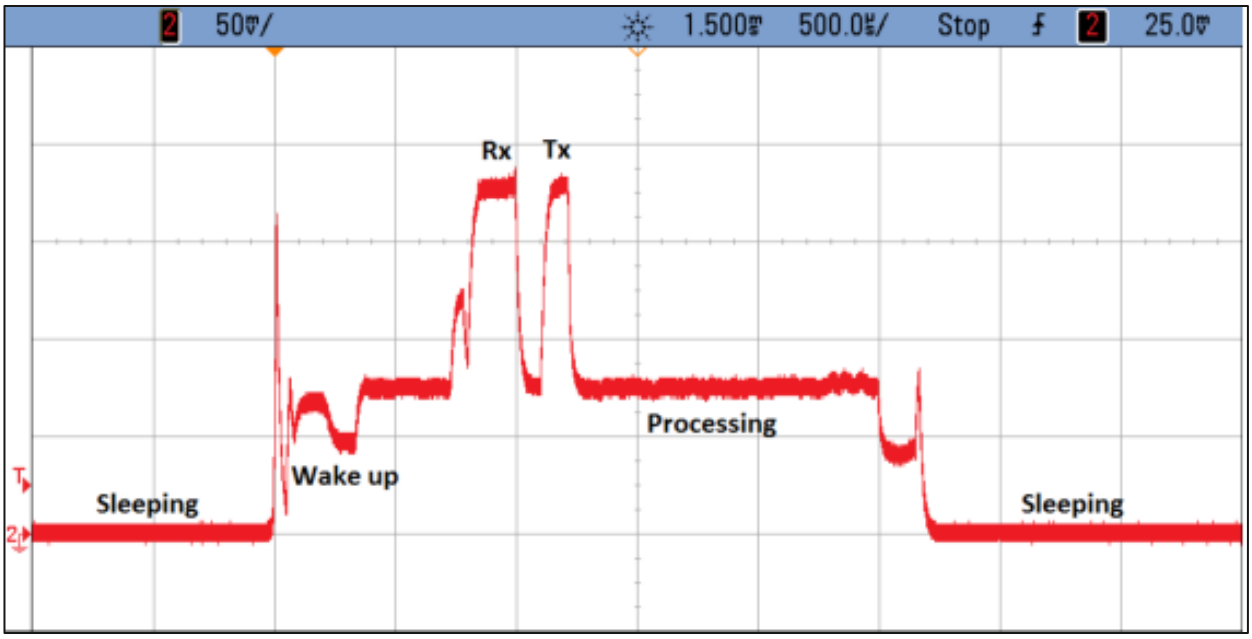
Bluetooth 4.0 Low Energy (BLE) on Nokia Oy:n alun perin kehittämä matalan virrankulutuksen omaava langaton tiedonsiirtotekniikka (Wibree), jonka yleisimmät käyttökohteet ovat tietokoneet, matkapuhelimet, langattomat kuulokkeet, sekä älykellot. BLE-tiedonsiirtotekniikka käyttää hyväkseen radioaaltoja taajuusalueella 2.4–2.48 GHz. Taajuusalue on jaoteltuna 40 eri kanavalle 2 MHz taajuusvälillä, lähettäen pieniä määriä dataa kerrallaan (kuva 16). Suuren datamäärän siirtämiseen BLE osaa käyttää hyödykseen Wi-Fi yhteyksiä saavuttaen jopa 24 Mb/s siirtonopeuden tarvittaessa (Learn about Bluetooth, 2021).




KUVA 16. BLE-yhteyden toiminta kuvattuna virrankulutuksena (BLE Software Developer's Guide, 2021).

BLE-yhteyden hetkellinen huippuvirrankulutus lähetystilassa (TX) on noin 15 mA ja lähetysväli on noin 100 ms. Lepotilassa virrankulutus on ainoastaan 1 µA

luokkaa. Yksittäisen lähetystapahtuman (kuva 17) aikana laite on alkuun lepotilassa (Sleep Mode), kunnes herätyssignaalin piikki antaa herätyskäskyn ja laite herää (Wake up), vastaanottaa datapaketin (RX), jonka jälkeen laite lähettää tiedon (TX) yhteyslaitteelle, että seuraava datapaketti voidaan vastaanottaa. Tämän jälkeen saatu datapaketti prosessoidaan (Processing) ja kun prosessointi on suoritettu, siirtyy laite takaisin lepotilaan (Sleep Mode) odottamaan seuraavan datapaketin vastaanottoa (BLE Software Developer's Guide, 2021).




KUVA 17. BLE yksittäinen yhteystapahtuma (BLE Software Developer's Guide, 2021).

Havainnollistavan esimerkin BLE:n matalasta virrankulutuksesta tuo hyvin esiin Joseph C. Decuir (2014). Oletetaan yhden sensoridatan lähetystapahtuman keston olevan 3 ms/lähetys, yleinen lähetysteho olkoon noin 15 mW ja virtalähteenä toimii paristo, jonka jännite on 1.55 V ja sähkövaraus 180 mAh (Decuir C. Joseph, 2014). Lasketaan kuinka paljon paristo kuluttaa virtaa *I* kaavaan (10) avulla:

$$P = U \times I, \tag{10}$$

jossa *P* on tarkoittaa lähetystehoa, *U* on pariston jännite ja halutaan tietää pariston *I* virrankulutus, niin muutetaan kaava muotoon (11):

$$I = \frac{P}{U} = \frac{15\text{mW}}{1.55\text{V}} \approx 9.68\ \text{mA}, \quad (11)$$

josta saadaan pariston kulutusvirta-arvoksi *I* noin 9.68 mA (SI-opas, 2019, 35).

Seuraavaksi lasketaan (kaava 12), kuinka pitkän ajan *T* pariston sähkövaraus *Q* kestää, jatkuvan lähetyksen virrankulutusta *I* (SI-opas, 2019, 42–50):

$$T = \frac{Q}{I} = \frac{180\ \text{mAh}}{9.68\ \text{mA}} \approx 18.6\text{h}, \quad (12)$$

Näin saadaan pariston kestoajaksi 18.6 tuntia jos jatkuva lähetys olisi käytössä. Tunnit kun muutetaan sekunneiksi, on aika 66 960 sekuntia jatkuvaa lähetysaikaa. Sekunnit jaetaan lähetyksen kestolla 3 ms/lähetys, niin mahtuu lähetyskertoja 18.6 tuntiin 22,32 miljoonaa kappaletta (Decuir C. Joseph, 2014).

Oletetaan, että dataa lähetettäisiin sensorilta 1 minuutin aikavälillä. Vuorokaudessa on 1440 minuuttia ja kun lähetyskertojen kappalemäärä jaetaan vuorokauden minuuteilla, saadaan lähetyspäiviä 15 500 päivää. Jaetaan vielä päivät 365 päivällä, jolloin saadaan yhden pariston laskennalliseksi virrankulutusiäksi yli 42 vuotta, joka ylittää kaupallisen nykypariston elinikäodotteen reilusti. Tästä voidaankin siis päätellä, että BLE mahdollistaa pitkän käyttöiän sensoreille ja järjestelmille, joiden kriteereistä yksi on energiatehokkuus (Decuir C. Joseph, 2014).

### 3.2.8 Suunto MoveSense

Suunto Oy:n MoveSense-anturijärjestelmä (kuva 18) on Suomessa kehitetty sekä valmistettu, pieni, kestävä, vedenpitävä, monipuolinen ja pienen virrankulutuksen omaava anturijärjestelmä, joka soveltuu aktiivisuuden ja työsuoritteiden seurantaan monipuolisissa ympäristöissä. Anturijärjestelmällä voidaan mitata mm. sydämen sykettä kahdella eri tekniikalla, yksikanavaisella ECG-tekniikalla ja optisella PPG-tekniikalla. Kehon lämpötilaa voidaan mitata magnetometrin ja optisen PPG-tekniikan avulla. Kokonaisvaltaisen liikkeen mittaaminen on-

nistuu 9-akselisen mittausjärjestelmän avulla. Mittausjärjestelmän sisältää kiihtyvyysanturin, gyroskoopin, magnetometrin ja sykemittarin (Suunto, 2021; MoveSense sensor, 2021).

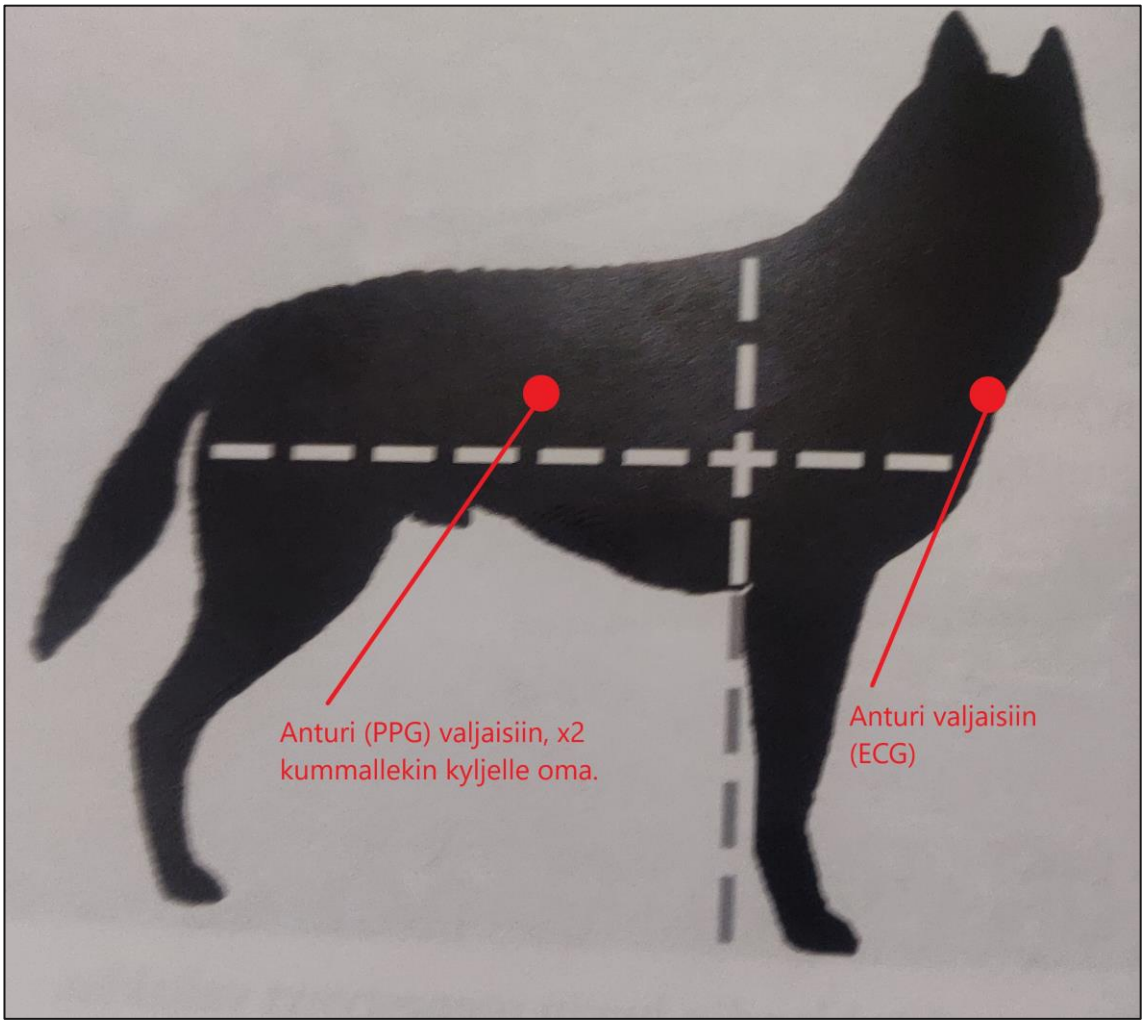

KUVA 18. MoveSense-anturijärjestelmä (MoveSense sensor, 2021).

Anturijärjestelmä on pienen kolikon muotoinen, joka on halkaisijaltaan 36.6 mm, paksuudeltaan 10.6 mm ja painaa pariston kanssa noin 9.4 g. Virtalähteenä anturijärjestelmä käyttää CR 2025 Litium-nappiparistoa. Yksilöllisen ID-tunnuksen ja avoimen API-rajapinnan ansiosta voidaan useita anturijärjestelmiä linkittää yhteen, sekoittamatta antureiden mittaamaa dataa. Mittausdata voidaan yksilöidä jokaisen anturin yksilöllisen ID-tunnusten perusteella. Koirien tapauksessa esimerkiksi, rinta, vasen kylki, oikea kylki tai vasen etutassu, oikea etutassu, vasen takatassu, oikea takatassu (Suunto, 2021; MoveSense sensor, 2021).

MoveSense-anturijärjestelmän järjestelmäpiiri nRF52832 on Nordic Semiconductor Finland Oy:n valmistama. Järjestelmäpiiri pitää sisällään 32-bittisen ARM (Advanced RISC Machines) Cortex -tuoteperheeseen kuuluvan M4 prosessorin,

3 Mbit jono-puskurimuistia väliaikaisen mittausdatan tallentamiseen, 64 kB sisäistä RAM-muistia (Random Access Memory), sekä 512 kB sisäistä FLASH-muistia. Muistien käyttö jakaantuu MoveSense-käyttöjärjestelmälle ja mobiilisovellukselle. Anturijärjestelmän tiedonsiirto tapahtuu 1-Wire tiedonsiirtoväylää pitkin ja langatonta BLE-teknologiaa (Bluetooth Low Energy) hyödyntäen, joka mahdollistaa reaaliaikaisen tiedon siirron ja seuraamisen esimerkiksi mobiilisovellusta käyttäen (Suunto, 2021; MoveSense sensor, 2021).

Alla kuvaus (kuva 19) järjestelmän sensoreiden sijoituspaikoista rekikoirille. Katso myös luvun 2 kohta 2.3.1 Valjaat, jossa esitellään rekikoirien työvaljaita.

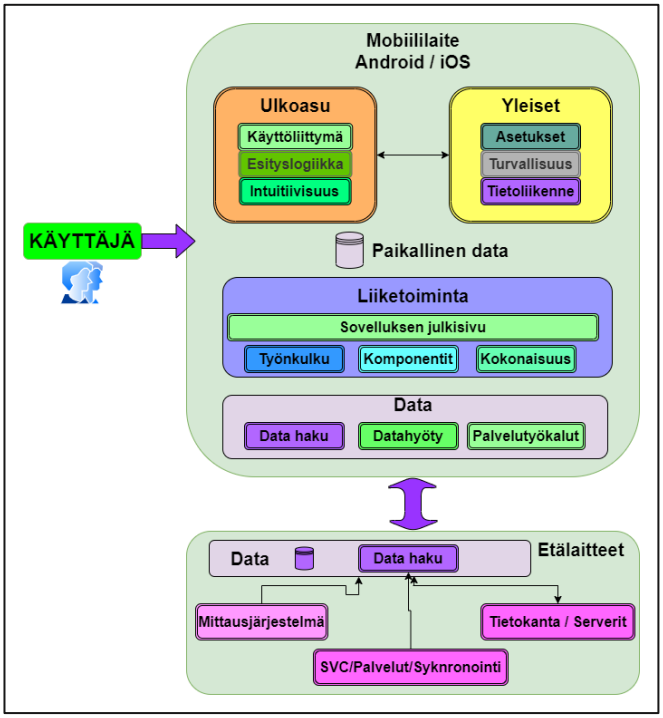



KUVA 19. MoveSense-anturien fyysinen sijoituspaikka (Siperianhusky valjakkoelämää).

Lisätietoja MoveSense-anturijärjestelmästä löytyy virallisesta teknisestä dokumentaatiosta, jonka löydät liitteestä 1. Vaikka Suunnon MoveSense-anturijärjestelmä vaikuttaa optimaaliselta valinnalta speksien perusteella, on muutamia ominaisuuksia, joiden speksit eivät ole optimaalisia suunniteltavaa anturijärjestelmää varten. Erityisesti anturijärjestelmän säävaihteluihin sopeutuminen ja kylmän sietokyky. Anturijärjestelmän on pysyttävä toimintakykyisenä ennalta arvaamattomissakin sääolosuhteissa ja näiden speksien perusteella se ei välttämättä ole sitä. Toinen huolenaihe on, että Bluetoothin yhteyden kantama ei yksinkertaisesti riitä jäljittämään koiria, jos koirat pääsevät karkuteille, voivat ne kulkea useiden kymmenien kilometrien matkan karkuteillä ollessaan.

## 3.3 Ohjelmistoarkkitehtuuri

Ohjelmistoarkkitehtuuri on kuvaus, joka koostuu ohjelmistoon luoduista rakenteista, elementeistä ja niiden välisistä suhteista toisiinsa. Ohjelmistokokojen kasvaessa suunnitteluongelmat ylittävät algoritmien ja tietorakenteiden määrittelyn. Suurissa järjestelmissä huonosti suunniteltu kokonaisuus ja sen määrittely esiintyy ongelmana (Fowler, Martin, 2019). Alla Arctic tracker -ohjelmiston arkkitehtuuri opinnäytetyön tekohetkellä (kuva 20).

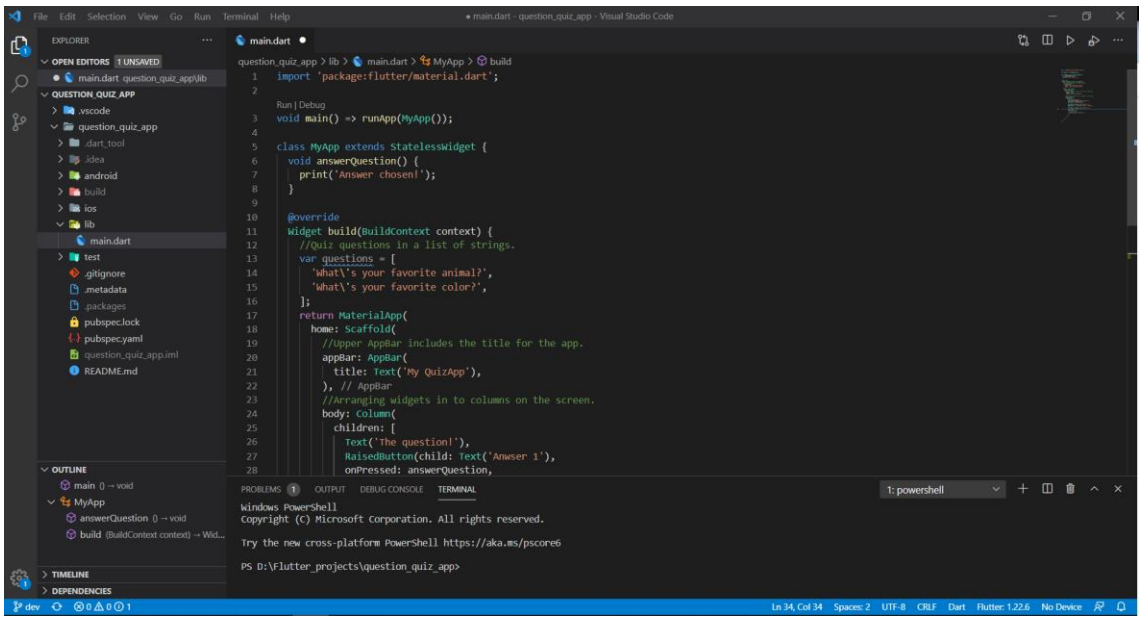



KUVA 20. Ohjelmistoarkkitehtuuri kuvaus.

### 3.3.1 Ohjelmistotyökalut

Opinnäytetyössä kaikki koodi luotiin ja käsiteltiin käyttäen VS Code (Visual Studio Code) tekstieditoria (kuva 21). VS Code valikoitui käytettäväksi, koska se on kevytkäyttöinen, avoimen lähdekoodin omaava ja laajalti käyttäjän tarpeiden mukaan mukautettava, monialustainen ohjelmoijille suunnattu ilmainen tekstieditori. VS Code on saatavilla Linux-, MacOS- ja Windows -käyttöjärjestelmille. VS Code omaa sisäänrakennetun tuen JavaScript- ja TypeScript-ohjelmointikielille, Node.js ajoympäristölle, virheenkorjaukselle, Git -versionhallinnalle, syntaksi korostukselle, snippeteille ja refaktoroinille.

Itse asennettavien tukipakettien avulla VS Codeen pystyy tekemään haluamiaan laajennuksia aina useista ohjelmointikielistä (C, C#, C++, Python, Java, Go, Dart, Ruby, Rust…), haluttuihin teemoihin tai vaikka koodin arkkitehtuurin visualisoimiseen. Opinnäytetyön tekijän näkökulmasta VS Code on erittäin monikäyttöinen, kevyt ja hyvä tekstieditori ohjelmointiin (VS Code, 2021).




KUVA 21. Visual Studio Code.

Opinnäytetyön käyttöjärjestelmävaatimusten valossa, mobiilisovelluksen tekemiseen oli järkevintä käyttää Flutter -ohjelmistopakettia (SDK), joka soveltuu mobiili-, web- ja työpöytäsovellusten käyttöliittymien toteuttamiseen Android-, iOS-, Linux-, Mac-, Windows- ja Fuchia -käyttöjärjestelmille, ja jonka käyttämiseen VS Code tarjoaa näppärän laajennuspaketin (Flutter, 2021).

Flutter on Googlen vuonna 2017 julkaisema avoimen lähdekoodin ohjelmistopaketti, jonka haastavin kilpailija on Facebook Oy:n luoma React Native -mobiilisovelluskehys. Kehittäjät voivat muokata Flutteria omiin tarpeisiinsa sopivaksi ja se on nimenomaan kehitetty modernin mobiilikehityksen tarpeisiin kuten PWA-sovellukset (Progressive Web App). Opinnäytetyön resurssien ja tavoitteiden näkökulmasta, Flutter oli optimaalinen valinta Arctic tracker -ohjelmiston luomiseen. Flutterin avulla valmistuu yhdestä lähdekoodista mobiilisovellus sekä Android- että iOS -käyttöjärjestelmille, joka oli yksi tämän opinnäytetyön tavoitteista (Flutter, 2021).

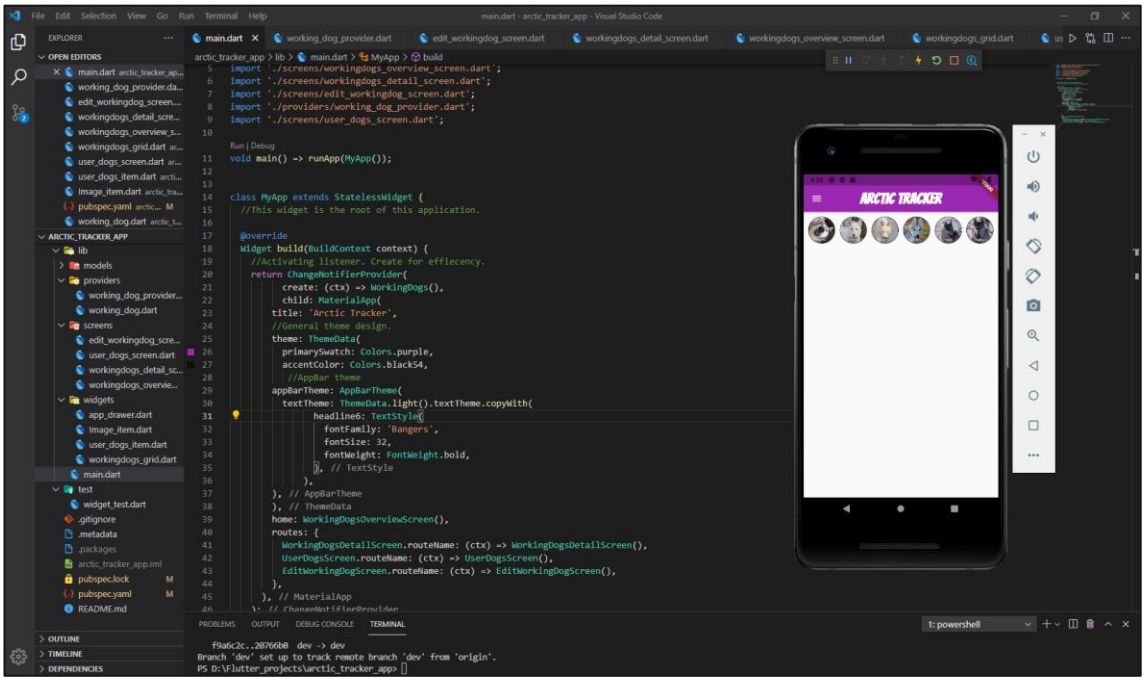
Flutterin pääasiallinen ohjelmointikieli on Googlen kehittämä Dart -ohjelmointikieli, mutta sen lisäksi Flutterin taustaohjelmointi on pääsiallisesti tehty C- ja C++ -ohjelmointikieltä käyttäen. Muun muassa Flutter-moottori ja sen käyttämä Skia-grafiikkamoottori ovat täysin kirjoitettu C- ja C++ -ohjelmointikielellä. Flutterin tärkein etu on, että yhdestä lähdekoodista valmistuu mobiilisovellus niin Android- kuin iOS -käyttöjärjestelmille, eikä lähdekoodia tarvitse kirjoittaa erikseen useille käyttöjärjestelmille. Tämä nopeuttaa kehittämistä ja vapauttaa resursseja muuhun käyttöön. Flutterilla toteutettavat sovellukset rakentuvat widgettejä (ohjelmistokomponentti) käyttäen, joiden tarkoituksena on ryhmitellä sisältöä tietyn ominaisuuden tai käyttötarkoituksen mukaan ja koska Flutter on avointa lähdekoodia, voi kehittäjä luoda uusia widgettejä itse tarpeensa mukaan (Flutter, 2021).

Widgetit ovat kuin laatikoita, joihin on sijoitettu eri sisältöä kuten, tekstiä, animaatioita, AR-komponentteja (Augmented Reality), toiminnallisuuksia ja eri elementtien asetteluja. Widgettejä voidaan muokata ja yhdistellä siten, että lopputuloksena on yhtenäinen, helppokäyttöinen ja visuaalisesti miellyttävä sovellus (Flutter, 2021).

Flutterin muita etuja kilpailijaansa React Nativeen nähden, ovat Flutterin koodin esikatselutoiminto ”hot reload” -ominaisuus, jolloin koodin muokkaaminen sovelluksen käyttöliittymään voidaan tarkistaa reaaliajassa. Tämän lisäksi Flutter kääntyy suoraan natiivikoodiksi, jolloin se on React Nativea suorituskykyisempi vaihtoehto (Flutter, 2021).

### 3.3.2 Dart-ohjelmointikieli

Dart on Lars Bakin suunnittelema ja Googlen kehittämä, vuonna 2011 julkaistu ohjelmointikieli, joka on vaihtoehto JavaScript-ohjelmointikielelle. Se on luotu erityisesti silmällä pitäen nopeaa ja tehokasta sovelluskehitystä, jonka tavoitteena tarjota kehittäjille mahdollisimman asiakasoptimoitu ohjelmointikieli monitasoiseen ajonaikaiseen kehitystyöhön (kuva 22), sisältäen mm. pikalataustoiminnon (Dart, 2021).




KUVA 22. Dart -ohjelmointikielen kuvaus projektista.

Ohjelmointikielet määritellään niiden teknisen kehyksen perusteella, jonka kehityksen aikana määritellään tulevan ohjelmointikielen ominaisuuksia ja vahvuuksia. Dart -ohjelmointikielen tekninen kehys on määritelty niin, että se soveltuu erityisesti useille eri alustoille (web, mobiili ja työpöytä) kehitettäville sovelluksille. Dart -ohjelmointikieli tukee kehittäjille tärkeitä ominaisuuksia, kuten sovellusten muotoilua, koodin analysointia ja testausta (Dart, 2021).

Dart -ohjelmointikieli on kirjoitusasultaan turvallinen, sillä ohjelmointikielessä käytetään staattista syntaksin tarkistusta vertaamaan, että muuttujan kirjoitusasu

vastaa staattista syntaksia. Dart -ohjelmointikieli takaa myöskin täyden nolla-arvo turvallisuuden, joka tarkoittaa, että ohjelmointiarvoina ei voi olla nolla-arvoja, jos ei niitä kehittäjä ole tarkoituksella koodiin määrittänyt. Dartin joustavuus sallii dynaamisen syntaksin käytön yhdistettynä ajonaikaiseen tarkistamiseen, mistä on hyötyä erityisesti dynaamisen koodin testaamisessa. Dart -ohjelmointikielen ollessa avoimeen lähdekoodiin perustuva ohjelmointikieli, on siitä tarjolla kattavasti lisää tietoa syntaksista, kirjastoista ja toiminnallisuudesta. Tämä tieto löytyy internet sivustolta, jonka linkin löydät lähdeluettelosta (Dart, 2021).

### 3.3.3 C++/C-ohjelmointikieli

Ohjelmointikielinä C- ja C++ -ohjelmointikieli ovat lähellä toisiaan, sillä onhan C++ -ohjelmointikieli kehitetty Bjarne Stroustrupin toimesta 1980-luvulla C-ohjelmointikielen pohjalta, lisäämällä ohjelmointikieleen ominaisuuksia, jotka mahdollistavat geneerisen ja olio-ohjelmointiperustaisen ohjelmointityylin. Ohjelmointityyliä voidaan käyttää useilla eri alustoilla ja lukuisissa käyttöjärjestelmissä. C- ja C++ -ohjelmointikieli ovat todella monipuolisia ja laiteläheisiä ohjelmointikieliä, jotka sopivat niin matalan tason ohjelmointiin, kuin kokonaisten käyttöjärjestelmien ja laiteajureidenkin kirjoittamiseen. Ei siis tule yllätyksenä, että TIOBE:n (The Importance of Being Earnest) vuosittain päivittämässä indeksissä, joka mittaa kaiken julkisen kirjoitetun koodin määrää. Ovat C -ohjelmointikieli sijalla 1. ja C++ -ohjelmointikieli sijalla 4 (Stroustrup, B. 1994; Stroustrup, B. 2015; TIOBE, 2021).

C++ -ohjelmointikieli on siis käytännössä C-ohjelmointikielestä paranneltu versio, sisältäen toiminnallisuuksia kuten isomman standardikirjaston, luokat, virtuaalisen moniperintätuen, mallit, poikkeukset, syntaksin tarkastuksen ja nimiavaruustuen, jota ei C-ohjelmointikielestä löydy. C++ -ohjelmointikieli on siis moniparadigmainen, joka mahdollistaa mm. proseduraalisen-, olio- ja geneerisenohjelmoinnin mallien käytön. C++ muistinhallinta perustuu RAII-konseptiin (Resource acquisition is initialization), jossa muistin resurssia varataan rakentajassa, vapautetaan purkajassa (kuva 23) ja tämän muistinhallintakonseptin ansiosta C++ -ohjelmointikieltä pidetäänkin tehokkaana muistinoptimointiohjelmointikielenä (Stroustrup, B. 1994; Stroustrup, B. 2015).

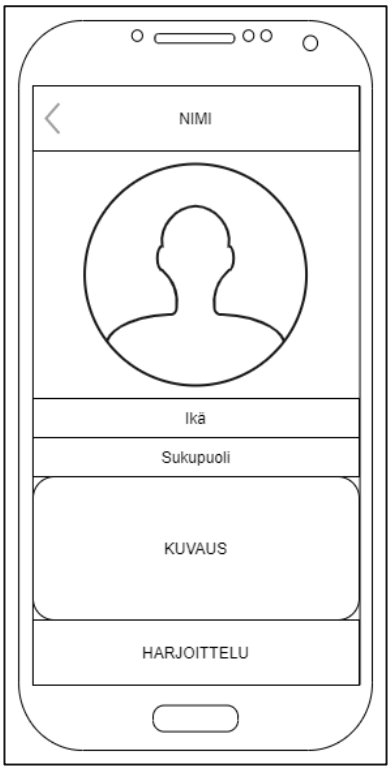
```
class Vector
{
private:
    double *elem;
    int sz;
public:
    Vector(int s) : elem{new double[s]}, sz{s} // constructor
    {
        for (int i = 0; i != s; ++1) elem[i] = 0;
    }
    ~Vector() { delete [] elem; } // destructor

    double& operator[](int i);
    int size() const { return sz; }
};

void fct(int n)
{
    Vector v(n); // acquire and initialize
    // .. use v ..
    {
        Vector v2(n*2); // acquire and initialize
        // .. use v2 ..
    } // v2 destroyed
    // .. use v ..
} // v destroyed
```

KUVA 23. RAII-muistinhallinta esimerkki (Stroustrup, B. 2015)

### 3.3.4 Design

Alustavan design-suunnittelun tekemiseen käytettiin diagrams.net palvelua, joka on verkosta löytyvä pilvipalveludiagrammiohjelmisto. Positiivisesti yllättänyt alustan toiminta, helppokäyttöisyys, liitettävyys Google Drive -palvelun kanssa ja alustan ominaisuudet sopivat erittäin hyvin mobiiliohjelmiston mockup ja wireframe suunnittelun tarpeisiin (kuva 24–25).


KUVA 24. Arctic tracker -ohjelmiston alustava wireframe.




KUVA 25. Arctic tracker -ohjelmiston alustava wireframe.

### 3.3.5 Kirjastot

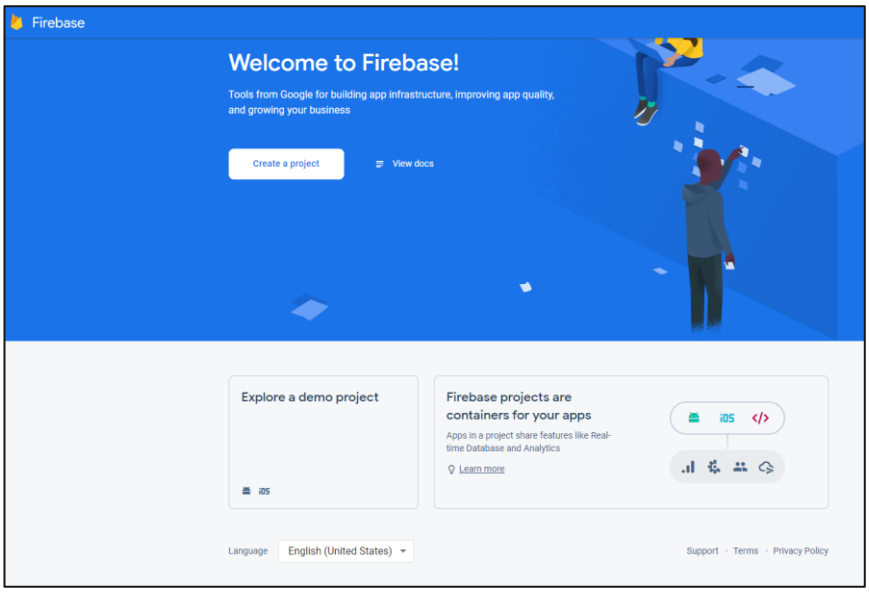
Googlen luoma Dart -ohjelmointikieli on luotu nopean ohjelmistokehittämisen tarpeisiin. Ohjelmointikieleen saatavilla olevien kirjastojen määrä on vakuuttava ja niiden toiminnallisuuden kirjo kasvaa koko ajan. Ohjelmistopaketit kuten Flutter, pystyy hyödyntämään näitä Dart -ohjelmointikieleen tarjolla olevia kirjastoja. Tämä nopeuttaa ohjelmistokehittämistä, sillä kirjastoja käytettään erilaisten tukipalveluiden luomiseksi itsenäisesti suorittavien ohjelmien käyttöön. Ohjelmistokehittäjän ei siis tarvitse kirjoittaa koodia jokaiselle aliohjelmalle ja luokalle itse, vaan ne voidaan implementoida kirjastojen avulla pääohjelman käyttöön. Opinnäytetyöntekijä on kuitenkin havainnut kehitystyössään näiden kirjastojen päivitysten ja Flutter -ohjelmistopaketin satunnaisen datapaketti häviöilmiön, osa kirjastoista ei suostu päivittymään kuin pakotuskomennolla (--force), jolloin kirjatoissa oleva data saattaa vahingoittua ja näin aiheuttaa ongelmia kirjastoja käytettäessä. Alla luettelo tässä opinnäytetyössä käytetyistä kirjastoista.

- http: 0.13.3, kirjasto http-pyyntöjen tekemiseen tietokantaan.

- image_picker: ^0.7.5+2, kirjasto, joka mahdollistaa kuvien hakemisen kuvakirjastosta tai kameralla uusien kuvien ottamisen.
- path_provider: ^2.0.1, kirjasto suosikkireitit karttanäkymälle.
- path: ^1.8.0, kirjasto reittisuunnittelulle.
- sqflite: ^2.0.0+3, kirjasto SQLite database moottori.
- location: ^4.1.1, kirjasto reaaliaikaiselle sijainnille ja virrankulutusoptimoinnille.
- google_maps_flutter: ^2.0.5, kirjasto Google karttapalvelulle.
- provider: ^5.0.0, kirjasto, joka helpottaa widgettien periyttämistä.
- cupertino_icons: ^1.0.0, kirjasto ikoneita design-käyttö.
- intl: ^0.17.0, kirjasto mahdollistaa käännökset ja parsimisen.
- OpenSans, fonttipaketti.
- Quicksand, fonttipaketti.
- Bangers, fonttipaketti.

### 3.3.6 Database

Opinnäytetyön yhteydessä käytettiin tietokantana Googlen tarjoamaa ilmaista Firebase -pilvipalvelualustaa (kuva 26). Firebase -pilvipalvelualusta sisältää useita eri komponentteja. Firebasen -pilvipalvelualustaan päädyttiin, koska kyseessä on tietokanta, jota Flutter -ohjelmistopaketti tukee kattavasti. Koska kyseessä oli ilmainen palvelu, päätti opinnäytetyöntekijä, että Firebase -pilvipalvelualusta sopii prototyyppiohjelmiston kehittämiseen ja testaamiseen tämän opinnäytetyön puitteissa. Lisätietoja Firebase -pilvipalvelualustasta löytyy lähdeluettelon linkistä (Firebase, 2021).



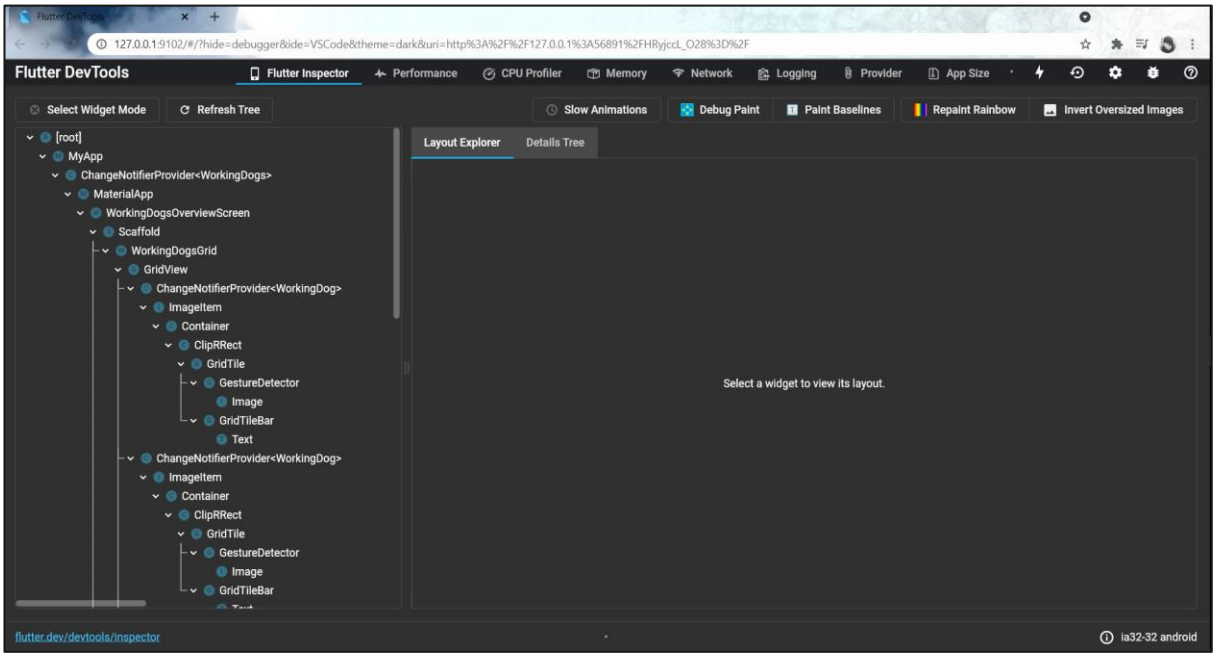

KUVA 26. Firebase -pilvipalvelualustan käyttöliittymä (Firebase, 2021).

# 4 OHJELMISTOTESTAUS

Viimeinen luku käsittelee opinnäytetyön aikana edenneen Arctic tracker -mobiiliohjelmiston debuggaamista, sekä testaamista käyttäen Dart DevTools -työkalua, OnePlus 6T Android -käyttöjärjestelmän omaavaa mobiililaitetta ja iPhone 6 iOS -käyttöjärjestelmän omaavaa mobiililaitetta.

## 4.1 Dart DevTools

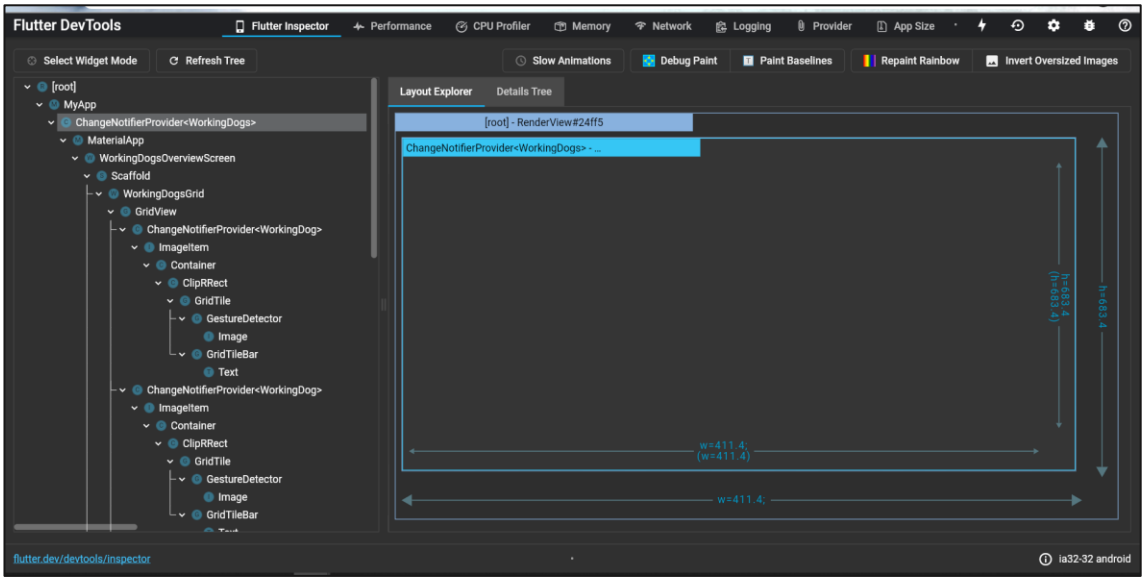
Dart DevTools on Dart -ohjelmointikielelle ja Flutter -ohjelmistopaketille suunniteltu virheenkorjaus- ja suorituskykytyökalu. Dart DevTools (kuva 27) mahdollistaa mm. Dart -ohjelmointikielellä ja Flutter -ohjelmistopaketin avulla luotujen käyttöliittymien ulkoasun ja tilan tarkastelua. Se auttaa diagnosoimaan suorituskykyongelmia, profiloimaan prosessorin, muistin ja tietoverkkojen käytön tehokkuutta, mahdollistaa lähdekoodin diagnosoinnin ja virheenkorjaukset. Sillä voidaan myös analysoida koodia, ohjelmiston kokoa ja yleistä diagnostiikkaa (Dart DevTools, 2021).




KUVA 27. Dart DevTools yleisnäkymä.

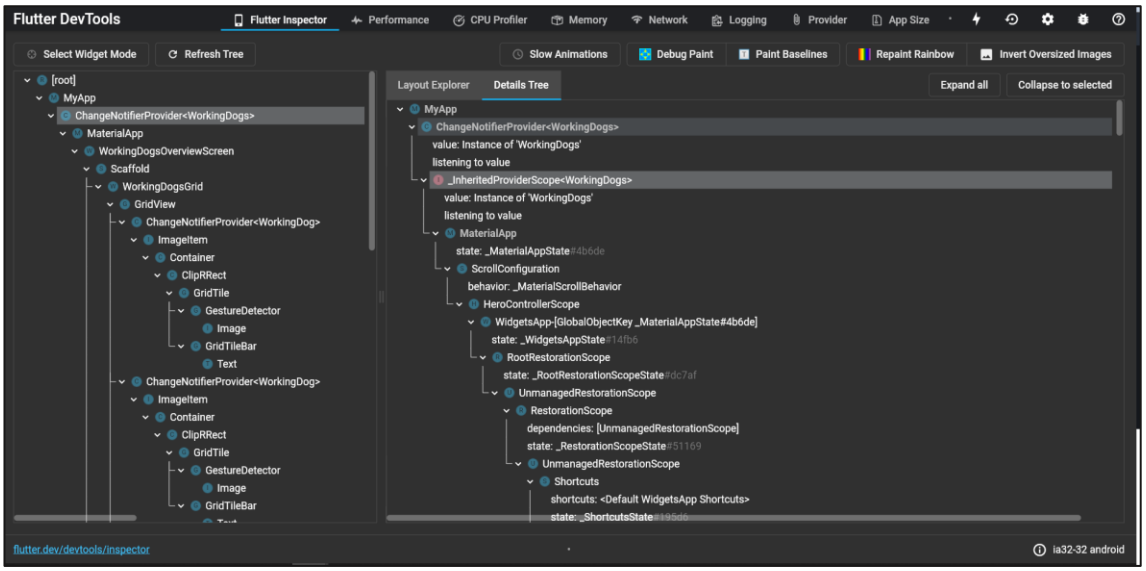
### 4.1.1 Ohjelmistopuun analysointi

Ohjelmistopuuta analysoitaessa Arctic tracker -mobiiliohjelman ohjelmistopuusta valittiin haluttu widget, jonka dimensiot ilmestyivät asettelunhallintaikkunaan (Layout Explorer). Asettelunhallintaikkunassa havainnollistettiin, kuinka valitut widgetit ja niiden childrenit ovat järjestelty koodissa. Analysointiohjelma tunnisti widgettien dimensiot pysty- ja poikittaisakselin suunnissa, sekä sen mistä widget alkaa, minne se loppuu, ja oliko widgetillä välitilaa. Analysointiohjelmalla tarkasteltiin myöskin Arctic tracker -mobiiliohjelmiston asettelun rajoitteita ja joustavuutta. Ohjelmistopuuta analysoitaessa ei toistaiseksi löydetty asettelurajoitteita, renderöintivirheitä, yhteensopivuusongelmia tai ylivuotovirheitä (kuva 28).




KUVA 28. Ohjelmistopuun analysointi.

Asettelurajoitukset esiintyisivät analysointiohjelmassa punaisella värillä ja ylivuotovirheet ”keltainen-teippi”-kuviossa. Dart DevTools -työkalun havainnollistama ohjelmistopuun rakenne (kuva 29) osoitti opinnäytetyön tekijälle monia jatkokehityksen kohteita, ohjelmiston prototyyppiversiota 1.1 ajatellen.




KUVA 29. Ohjelmistopuun rakenteen analysointi.

### 4.1.2 Suorituskyvyn testaaminen

Suorituskykyä mitattiin Dart DevTools -työkalulla, käyttämällä Arctic tracker -sovellusta Android- ja iOS-laitteen, sekä virtuaalisen emulaattorin avulla. Suorituskykynäkymässä Dart DevTools tarjoaa suorituskykytietoa, joka koostuu kolmesta osasta: Flutter-kehyskaaviosta (frame/ms), aikajanatapahtumakaaviosta (UI) ja prosessoriprofiloinnista (Raster).

Suorituskykynäkymään muodostuva kaavio (kuva 30) esittää Arctic tracker -mobiilisovelluksen kehystietoja. Jokainen kaavioon asetettu palkki edustaa yhtä Flutter-kehystä. Palkit ovat värikoodattu edustamaan eri osa-alueita työstä, joka tapahtuu Flutter-kehyksen renderöinnissä ohjelman ajoaikana. Nämä palkit kuvaavat esimerkiksi mitä muutoksia tapahtuu käyttöliittymän threadissa (vaalean sininen, UI thread), raster threadissa (tumman sininen, GPU thread), tai kehyksen renderöintinopeudessa (punainen, Jank, slow frame) ohjelman ajoaikana (Dart DevTools, 2021).

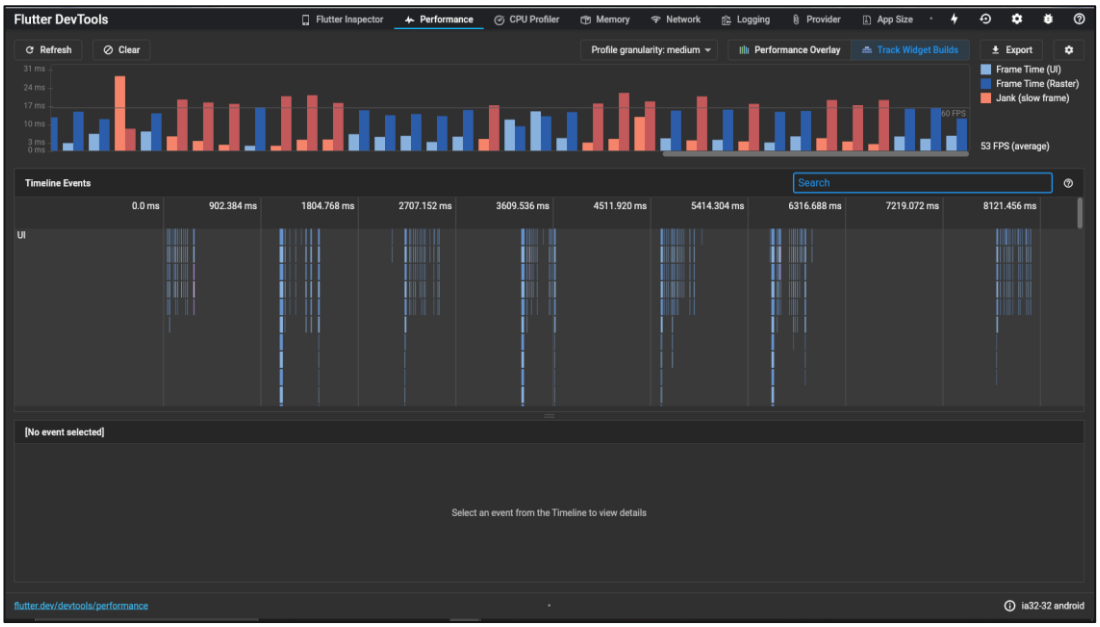

KUVA 30. Dart DevTools suorituskykynäkymä.

Google ilmoittaa toivotut raja-arvot kehyksen renderöintinopeudelle (Jank). Ohjelmiston katsotaan olevan ”janky” eli hidasframinen, jos kehyksen valmistumisajan keskiarvo ylittää 16 ms/frame (60 FPS-laitteella). 60 FPS (ruutua per sekunti) kehysnopeuden saavuttamiseksi, on testattavan ohjelman pystyttävä renderöimään kehyksiä 16 ms/frame nopeudella tai sen alle. Muussa tapauksessa loppukäyttäjä saattaa kokea ohjelmistoa käyttäessään epämiellyttävää käyttäytymistä ruudulla ja ruutuviivettä (Dart DevTools, 2021).

Vertailtaessa tuloksia voidaan päätellä (kuva 31–33), että virtuaalisella emulaattorilla ohjelmiston suorituskyvyn mittaaminen ei anna todenmukaista tietoa. Hitaiden ruutujen keskiarvon heitellessä aina yli 30 ms/frame saakka. Voidaan päätellä, että mittaustulokseen vaikuttaa laitteiston suorituskyky, jolla virtuaaliympäristö oli luotu.

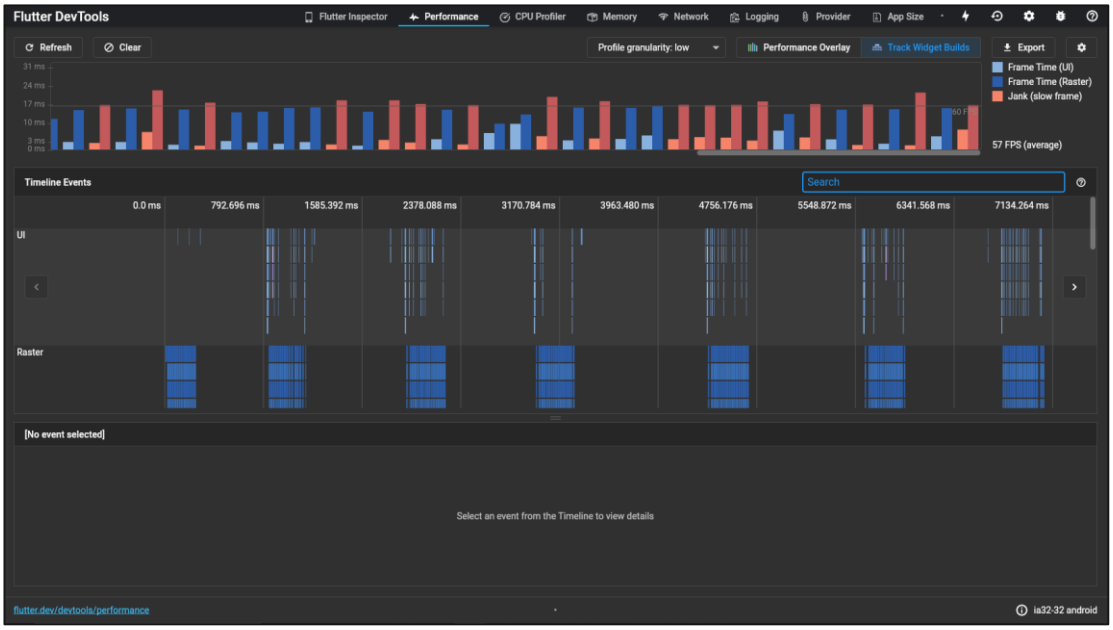

KUVA 31. Emulaattorin suorituskyvyn mittaaminen.

Android- ja iOS -mobiililaitteiden testituloksista voitiin havaita selkeä ero suorituskykytestissä varsinkin hitaiden framien kohdalla, verrattuna virtuaaliemulaattoriin. Suorituskyvyn testaaminen suoritettiin liittäen OnePlus 6T Android-mobiililaite tietokoneeseen käyttäen USB-C-kaapelia ja iPhone 6 iOS -mobiililaite tietokoneeseen käyttäen Lightning-kaapelia.




KUVA 32. Android -mobiililaitteella mitatut suorituskyky arvot.

Vertaillessa Android- ja iOS -mobiililaitteita keskenään, olivat hitaiden framien keskiarvot melko maltillisia, alle 16 ms/frame. Suuria suorituskykyeroja Arctic tracker -sovelluksessa ei ole havaittavissa Android- ja iOS -mobiililaitteen välillä (kuva 32–33).

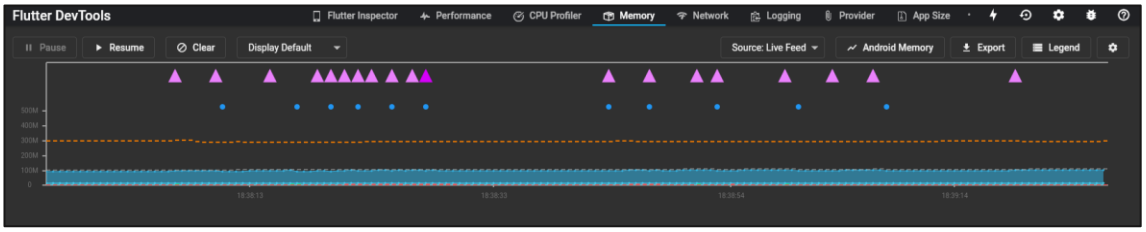



KUVA 33. iOS -mobiililaitteella mitatut suorituskyky arvot.

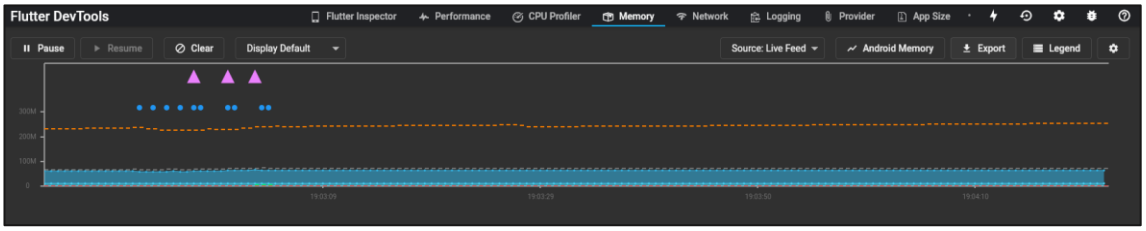
### 4.1.3 Keskusmuistin testaaminen

Suorituskykytestin perusteella, opinnäytetyön tekijä päätti jättää Arctic tracker -mobiilisovelluksen testaamisen virtuaalisella emulaattorilla kokonaan pois lopuista testeistä, harhaanjohtavien mittaustulosten mahdollisuuden perusteella.

Keskusmuistin (RAM) testaaminen ja analysointi Dart DevTools -työkalun avulla oli melko vaivatonta ja helppoa. Analyysin avulla pystyttiin tunnistamaan kaikki muistimallit, jotka voivat aiheuttaa muistivuotoja tai johtaa sovellusten kaatumiseen, silloin kun keskusmuistin rajat tulevat vastaan. Tehoton muistinkäyttö voi esimerkiksi johtua huonosta koodista, jossa suuria resursseja ladataan tehottomasti. Keskusmuistin käyttöä voi parantaa esimerkiksi dekoodaamalla suuria resursseja pienemmäksi ja pilkkomalla käytettävät resurssit pienempiin paketteihin.




KUVA 34. OnePlus 6T keskusmuistin mittaustulokset.




KUVA 35. iPhone 6 keskusmuistin mittaustulokset.

Mittaustulosten vertailussa voidaan todeta (kuva 34–35), että iOS -käyttöjärjestelmän tehokkaan muistinkäytön ominaisuudet, ovat huomattavissa verrattuna Android -käyttöjärjestelmään noin 30 Mbit erolla testituloksissa. Yleisesti ottaen mittaustulokset viittaavat siihen, että muistivuotoja ei ole havaittavissa Arctic tracker -sovelluksessa, sillä kummankaan käyttöjärjestelmän kohdalla muistivaraus ei kasva ajan kuluessa, vaan muistinvaraus pysyy maltillisesti alle 100 Mbit.

# 5 POHDINTA

Tämän opinnäytetyön päätavoitteena oli selvittää mahdollisuudet valmistaa rekikoirille suunnattu aktiivisuusjärjestelmä ja luoda kokonaisuudessaan järjestelmän prototyyppisuunnitelma. Lisäksi tavoitteena oli toteuttaa siihen liittyen ensimmäinen prototyyppi alustariippumattomasta mobiiliohjelmasta, joka opinnäytetyön valmistumisen aikaan ei ollut vielä saavuttanut loppukäyttäjälle julkaisukelpoista vaihetta.

Opinnäytetyön tuloksena valmistui rekikoirille tarkoitetun aktiivisuusjärjestelmän arkkitehtuurisuunnitelmakokonaisuus, jonka pohjalta opinnäytetyöntekijä loi aktiivisuusjärjestelmän mobiiliohjelmiston prototyyppi 1.0 version. Ohjelmiston testaamisen tulokset olivat rohkaisevia, sillä ohjelmiston prototyyppiversio 1.0 osoittautui ohjelmistopuultaan virhevapaaksi. Flutter -ohjelmistopaketin avulla kehitettyjen julkaistavien mobiiliohjelmistojen toivottavaksi tehokkuus raja-arvoksi Googlen ilmoittaa maksimissaan 16 ms/frame. Testitulosten perusteella voimme todeta, että mobiiliohjelmiston prototyyppiversio 1.0 täyttää Googlen tehokkuusvaatimuksen alittaen tuon 16 ms/frame tehokkuusvaatimuksen.

Arkkitehtuurin suunnittelu ja selkeys onnistuivat hyvin, vaikka aihe on laaja ja vaatii tietämystä monilta eri osa-alueilta, aina ohjelmisto- ja tietoliikennetekniikasta rekikoiriin ja eläinlääketieteeseen. Mobiiliohjelmiston koodaaminen onnistui hyvin testivalmiiksi, mutta Dart -ohjelmointikielen ja Flutter -ohjelmistopaketin kriittiset päivitykset keväällä 2021 rikkoivat osittain kehitysvaiheessa olevan ohjelmistoprojektin koodin, koodi osoittautui haasteellisemmaksi korjata tämän opinnäytetyön puitteissa. Opinnäytetyön aikataulu ja jo saavutettu laajuus huomioiden, päätettiin osa sovelluksen toiminnallisuuksista jättää pois. Tämä oli näkökulma, jota opinnäytetyöntekijä ei ollut osannut ottaa huomioon arkkitehtuurisuunnitelmaa luodessaan.

Työn rajoittaviin tekijöihin voidaan lukea aiheen laajuus verrattuna käytettävissä oleviin resursseihin. Työ toteutettiin kokonaisuudessaan yksilötyönä ja testattiin ainoastaan kahdella mobiililaitteella. Näin jälkeen päin katsottuna, tavoite oli kunnianhimoinen, sillä aiheen laajuuden huomioon ottaen, aiheesta olisi voinut

tehdä useampikin henkilö opinnäytetyön. Työn jatkokehityksen kannalta on tärkeää, että resursseja saadaan lisättyä, jolloin yksilöllisen mittausjärjestelmän valmistaminen on mahdollista ilman kompromissiratkaisuja.

**LÄHTEET**

AN5192. 2020. Data Sheet. STMicroelectronics. Verkkoaineisto. Luettu 19.03.2021 https://www.st.com/resource/en/application_note/dm00517282-lsm6dso-alwayson-3d-accelerometer-and-3d-gyroscope-stmicroelectronics.pdf

ASM330LHH. 2020. Data Sheet. STMicroelectronics. Verkkoaineisto. Luettu 19.03.2021. https://www.st.com/resource/en/datasheet/ asm330lhh.pdf

Bluetooth Low Energy Software Developer's Guide. 2021. Verkkoaineisto. Texas Instruments Inc. Luettu 30.04.2021. http://software-dl.ti.com/lprf/simplelink_cc2640r2_sdk/1.30.00.25/exports/docs/blestack/ble_sw_dev_guide/html/cc2640/index.html#

Cary, B. 2009. Born to pull: the glory of sled dogs. Minneapolis University of Minnesota Press.

Dart. 2021. Dart documentation. Verkkoaineisto. Google 2021. Luettu 02.05.2021. https://dart.dev/guides

DartDev. 2021. Dart opensource. Verkkoaineisto. Google 2021. Luettu 28.05.2021. https://dart.dev/

Dart DevTools. 2021. Dart Documentation. Verkkoaineisto. Google 2021. Luettu.02.05.2021 https://flutter.dev/docs/development/tools/devtools/overview

Firebase. 2021. Firebase Documents. Verkkoaineisto. Google 2021. Luettu 10.05.2021. https://firebase.google.com/

Flutter. 2021. Flutter Developer Documents. Verkkoaineisto. Google 2021. Luettu 01.05.2021. https://flutter.dev/docs/resources/faq

FlutterDev. 2021. Flutter opensource. Verkkoaineisto. Google 2021. Luettu 28.05.2021. https://flutter.dev/

Fowler, Martin. 2019. Software Architecture Guide. Verkkoaineisto. Martin Fowler ThoughtWorks. Luettu 30.04.2021. https://www.martinfowler.com/architecture/

Guide to 1-Wire communication. 2008. Technical Documents. Verkkoaineisto. Maxim Integrated Products Inc. Luettu 24.03.2021 https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html

Huson, H. Parker, H. Runstadler, J. Ostrander, E. 2010. Verkkoaineisto. A genetic dissection of breed composition and performance enhancement in the Alaskan sled dog. Luettu 4.3.2021. https://bmcgenomdata.biomedcentral.com/track/pdf/10.1186/1471-2156-11-71.pdf

Joseph C. Decuir. 2014. Bluetooth 4.0: Low Energy. Verkkoaineisto. IEEE.

Luettu 01.05.2021. https://californiaconsultants.org/wp-content/uploads/2014/05/CNSV-1205-Decuir.pdf

Kennelliitto. Koirat & Koiraharrastaminen. Verkkoaineisto. OmaKoira-palvelu. Luettu 5.3.2021.
https://www.kennelliitto.fi/

Learn About Bluetooth. 2021. Verkkoaineisto. Bluetooth SIG Inc. Luettu 30.04.2021 https://www.bluetooth.com/learn-about-bluetooth/

MNX Microsystem Foundry. Verkkoaineisto. MEMS & Nanotechnology Exchange. Luettu 03.04.2021. https://www.mems-exchange.org/MEMS/what-is.html

MoveSense sensor. 2021. Data Sheet. Suunto Oy. Luettu 19.03.2021.
https://www.movesense.com/wp-content/uploads/2020/12/Movesense-Spec-Sheet-12-2020.pdf

Reading and Writing 1-Wire Devices through Serial Interfaces. 2009. Verkkoaineisto.Technical Documents. Maxim Integrated Products Inc. 24.03.2021
https://www.maximintegrated.com/en/design/technical-documents/app-notes/7/74.html

Sazonov, E. Neuman, M. 2014. E-kirja. Wearable Sensors Fundamentals, Implementation and Applications. USA, San Diego: Academic Press.

Science Direct. Verkkoaineisto. Heart rate & RR intervals. Luettu 19.3.2021.
https://www.sciencedirect.com/topics/medicine-and-dentistry/rr-interval

Store Norske Leksikon. Verkkoaineisto. Leonhard Seppala. Luettu 4.3.2021.
https://snl.no/Leonhard_Seppala

Stroustrup, B. 1994. The design and evolution of C++. Addison-Wesley.

Stroustrup, B. 2015. The C++ Programming Language. 4.painos. Addison-Wesley.

Suomalainen Siperianhusky-seura – Finska Siberian Husky-sällskapet ry. 2013. Siperianhusky. Valjakkoelämää.

Suomalainen Siperianhusky-seura – Finska Siberian Husky-sällskapet ry. Verkkoaineisto. Rotuinfo. Luettu 5.3.2021. http://www.siperianhusky.fi/wp/

Suomen standardisoimisliitto SFS, SI-opas. 7.painos. SFS Standardisointi, 2019.

Suunto. 2021. Verkkoaineisto. Movesense Active-anturijärjestelmä. Luettu 19.3.2021.https://www.movesense.com/

Thornton, S. & Marion, J. 2004. Classical Dynamics of particles and systems. 5. painos. USA, Belmont: Brooks/Cole–Thomson Learning.

TIOBE. 2021. TIOBE index. Verkkoaineisto. TIOBE 2021. Luettu 11.05.2021. https://www.tiobe.com/tiobe-index/

Traildog. Valjakkourheiluun. Verkkoaineisto. Tossut. Luettu 4.3.2021. https://www.traildog.fi/tuoteryhma/tossut-vetourheiluun

VS Code. 2021. Visual Studio Documents. Verkkoaineisto. Microsoft 2021. Luettu 01.05.2021. https://code.visualstudio.com/docs

Wilson, J. 2004. E-kirja. Sensor Technology Handbook. UK, Oxford: Elsevier Science & Technology.

**LIITTEET**

Liite 1. MoveSense sensori Data Sheet.

MoveSense sensor. 2021. Data Sheet. Suunto Oy. https://www.move-sense.com/wp-content/uploads/2020/12/Movesense-Spec-Sheet-12-2020.pdf

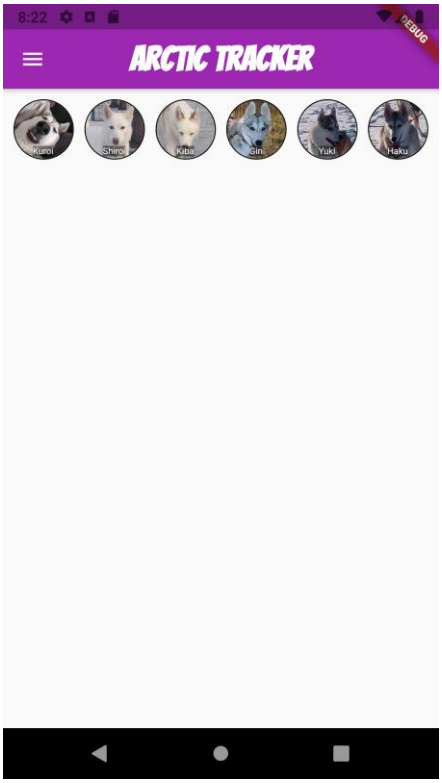
Liite 2. Kiihtyvyysanturi Data sheet.

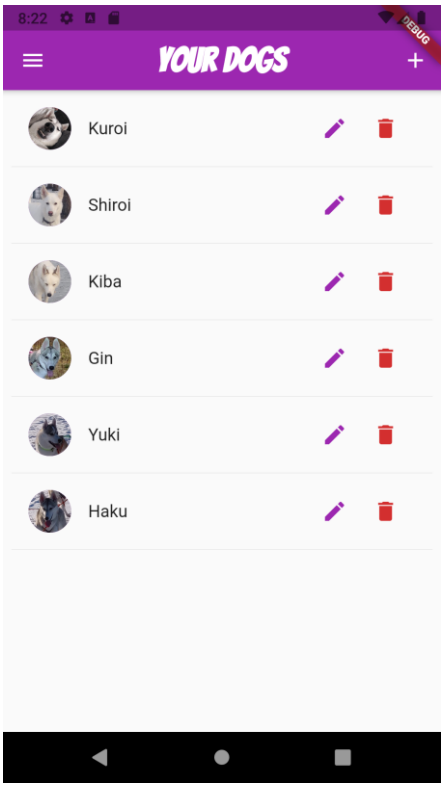
ASM330LHH. 2020. Data Sheet. STMicroelectronics. https://www.st.com/resource/en/datasheet/ asm330lhh.pdf

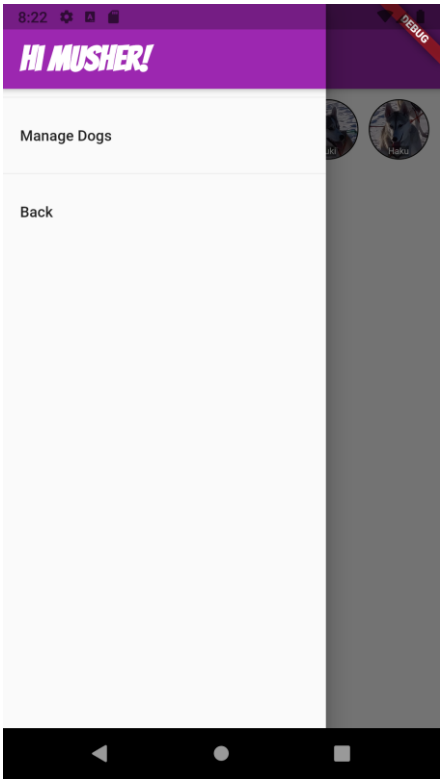
Liite 3. Gyroskooppi Data Sheet.

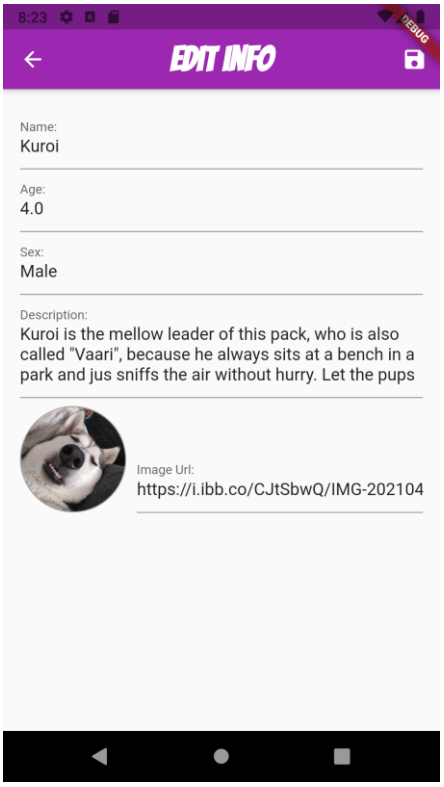
AN5192. 2020. Data Sheet. STMicroelectronics. https://www.st.com/resource/en/application_note/dm00517282-lsm6dso-alway-son-3d-accelerometer-and-3d-gyroscope-stmicroelectronics.pdf

Liite 4. Mobiilisovelluksen kuvankaappaukset.




Liite 5. Mobiilisovelluksen lähdekoodi.

Sovelluksen luomisessa on osittain hyödynnetty Dart -ohjelmointikielen ja Flutterin tarjoamaa avointa lähdekoodia (DartDev; FlutterDev).

```
import 'package:arctic_tracker_app/screens/user_dogs_screen.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import './screens/workingdogs_overview_screen.dart';
import './screens/workingdogs_detail_screen.dart';
import './screens/edit_workingdog_screen.dart';
import './providers/working_dog_provider.dart';
import './screens/user_dogs_screen.dart';

void main() => runApp(MyApp());


class MyApp extends StatelessWidget {
  //This widget is the root of this application.

  @override
  Widget build(BuildContext context) {
    //Activating listener. Create for effiecency.
    return ChangeNotifierProvider(
          create: (ctx) => WorkingDogs(),
          child: MaterialApp(
        title: 'Arctic Tracker',
        //General theme design.
        theme: ThemeData(
          primarySwatch: Colors.purple,
          accentColor: Colors.black54,
           //AppBar theme
        appBarTheme: AppBarTheme(
          textTheme: ThemeData.light().textTheme.copyWith(
                headline6: TextStyle(
                  fontFamily: 'Bangers',
                  fontSize: 32,
                  fontWeight: FontWeight.bold,
                ),
              ),
        ),
        ),
        home: WorkingDogsOverviewScreen(),
        routes: {
          WorkingDogsDetailScreen.routeName: (ctx) =>
WorkingDogsDetailScreen(),
          UserDogsScreen.routeName: (ctx) => UserDogsScreen(),
          EditWorkingDogScreen.routeName: (ctx) => EditWorkingDogScreen(),
        },
      ),
    );
  }
}
```

```dart
import 'package:flutter/material.dart';

import '../widgets/workingdogs_grid.dart';
//import '../screens/edit_workingdog_screen.dart';
import '../widgets/app_drawer.dart';

class WorkingDogsOverviewScreen extends StatefulWidget {
  @override
  _WorkingDogsOverviewScreenState createState() => _WorkingDogsOverviewScreenState();
}

class _WorkingDogsOverviewScreenState extends State<WorkingDogsOverviewScreen> {


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // A Bar shown on top of your app.
      appBar: AppBar(
        // App title center.
        centerTitle: true,
        title: Text(
          'Arctic Tracker',
        ),
        //Adding widgets to Appbar.
        actions: <Widget>[
          //Adding new profile button for design or userpic.
         /* IconButton(
            icon: Icon(Icons.add),
            //Creating a new profile when pressed.
            onPressed: () {
              Navigator.of(context).pushNamed(EditWorkingDogScreen.routeName);
            },
          ),*/
        ],
      ),
      drawer: AppDrawer(),
      //Ruuduko koirien tiedoille.
      body: WorkingDogsGrid(),
    );
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/working_dog_provider.dart';
import '../widgets/app_drawer.dart';
import './edit_workingdog_screen.dart';
import '../widgets/user_dogs_item.dart';

class UserDogsScreen extends StatelessWidget {
  static const routeName = '/user-dogs';

  @override
  Widget build(BuildContext context) {
    final WorkingDogs workingDogsData = Provider.of<WorkingDogs>(context);
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: const Text('Your Dogs'),
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.add),
            onPressed: () {
              Navigator.of(context).pushNamed(EditWorkingDogScreen.routeName);
            },
          ),
        ],
      ),
      drawer: AppDrawer(),
      body: Padding(
        padding: EdgeInsets.all(8),
        child: ListView.builder(
          itemCount: workingDogsData.loadedogs.length,
          itemBuilder: (_, i) => Column(
                children: [
                  UserDogItem(
                    workingDogsData.loadedogs[i].id,
                    workingDogsData.loadedogs[i].name,
                    workingDogsData.loadedogs[i].imageUrl,
                  ),
                  Divider(),
                ],
              ),
        ),
      ),
    );
  }
}
```

```dart
import '../providers/working_dog_provider.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
//import './edit_workingdog_screen.dart';

class WorkingDogsDetailScreen extends StatelessWidget {
  //final String id;

  //WorkingDogsDetailScreen(this.id);

  static const routeName = '/dog-detail';

  @override
  Widget build(BuildContext context) {
    //Gives us the id for the right pointer of detail info.
    final workingdogId =
        ModalRoute.of(context)!.settings.arguments as String;
    final loadedWorkingDog = Provider.of<WorkingDogs>(
      context,
      listen: false,
    ).findById(workingdogId);
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: Text(loadedWorkingDog.name),
        //Adding widgets to Appbar.
        actions: <Widget>[
          //Adding new profile button for design or userpic.
          /* IconButton(
            icon: Icon(Icons.edit),
            //When pressed, let's the user change profile data.
            onPressed: () {
              Navigator.of(context).pushNamed(EditWorkingDogScreen.routeName, arguments: workingdogId);
            },
          ),*/
        ],
      ),
      body: SingleChildScrollView(
        child: Column(
          children: <Widget>[
            Container(
              height: 310,
              width: double.infinity,
              alignment: Alignment.center,
              child: Container(
                alignment: Alignment.center,
                height: 300,
                width: 300,
                margin: EdgeInsets.only(),
                decoration: BoxDecoration(
                  border: Border.all(
                    width: 6,
                    color: Colors.black87,
                  ),
                  borderRadius: BorderRadius.all(
                    Radius.circular(1200),
                  ),
                ),
                child: ClipRRect(
                  borderRadius: BorderRadius.circular(1200),
                  child: Image.network(
                    loadedWorkingDog.imageUrl,
                    fit: BoxFit.cover,
                  ),
                ),
              ),
            ),
            //Display information.
            SizedBox(
              height: 10,
            ),
            Text(
              'Age: ' '${loadedWorkingDog.age}' ' years',
              style: TextStyle(
                  color: Colors.black,
                  fontSize: 28,
                  fontWeight: FontWeight.bold),
            ),
            SizedBox(
              height: 10,
            ),
            Text(
              'Sex: ' '${loadedWorkingDog.sex}',
              style: TextStyle(
                  color: Colors.black,
                  fontSize: 28,
                  fontWeight: FontWeight.bold),
            ),
            SizedBox(
              height: 10,
            ),
            Container(
              width: double.infinity,
              padding: EdgeInsets.symmetric(horizontal: 10),
              decoration: BoxDecoration(
                  border: Border.all(
                    width: 6,
                    color: Colors.black87,
                  ),
                  borderRadius: BorderRadius.all(
                    Radius.circular(30),
                  ),
                ),
              child: Text(
                loadedWorkingDog.description,
                textAlign: TextAlign.center,
                softWrap: true,
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 22,
                  fontWeight: FontWeight.normal,
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../screens/edit_workingdog_screen.dart';
import '../providers/working_dog_provider.dart';

class UserDogItem extends StatelessWidget {
  final String id;
  final String title;
  final String imageUrl;

  UserDogItem(this.id, this.title, this.imageUrl);

  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Text(title),
      leading: CircleAvatar(
        backgroundImage: NetworkImage(imageUrl),
      ),
      trailing: Container(
        width: 100,
        child: Row(
          children: <Widget>[
            IconButton(
              icon: Icon(Icons.edit),
              onPressed: () {
                Navigator.of(context).pushNamed(EditWorkingDogScreen.routeName, arguments: id);
              },
              color: Theme.of(context).primaryColor,
            ),
            IconButton(
              icon: Icon(Icons.delete),
              onPressed: () {
                Provider.of<WorkingDogs>(context, listen: false).deleteDog(id);
              },
              color: Theme.of(context).errorColor,
            ),
          ],
        ),
      ),
    );
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/working_dog_provider.dart';
import '../widgets/Image_item.dart';

class WorkingDogsGrid extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    //Listener of changes.
    final workingdogsData = Provider.of<WorkingDogs>(context);
    final workingdogs = workingdogsData.loadedogs;
    //Building the dog avatars and info.
    return GridView.builder(
      padding: const EdgeInsets.all(10.0),
      itemCount: workingdogs.length,
      itemBuilder: (ctx, i) => ChangeNotifierProvider.value(
        value: workingdogs[i],
        child: ImageItem(
        // workingdogs[i].id,
        //workingdogs[i].name,
        //workingdogs[i].age,
        //workingdogs[i].imageUrl,
          //workingddogs[i].theme,
        ),
      ),
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 6,
        childAspectRatio: 1,
        crossAxisSpacing: 10,
        mainAxisSpacing: 10,
      ),
    );
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../screens/workingdogs_detail_screen.dart';
import '../providers/working_dog.dart';

class ImageItem extends StatelessWidget {
  //final String id;
  //final String name;
  //final String imageUrl;

  //ImageItem(this.id, this.name, this.imageUrl);

  @override
  Widget build(BuildContext context) {
    //Listener for single objects.
    final workingdog = Provider.of<WorkingDog>(context, listen: false);

    return Container(
      decoration: BoxDecoration(
        border: Border.all(width: 1),
        borderRadius: BorderRadius.all(
          Radius.circular(100),
        ),
      ),
      child: ClipRRect(
        borderRadius: BorderRadius.circular(100),
        child: GridTile(
          //Wrapped so we can enable onTap for moving to details.
          child: GestureDetector(
            onTap: () {
              Navigator.of(context).pushNamed(
                WorkingDogsDetailScreen.routeName,
                arguments: workingdog.id,
              );
            },
            child: Image.network(
              workingdog.imageUrl,
              fit: BoxFit.cover,
            ),
          ),
          footer: GridTileBar(
            title: Text(
              workingdog.name,
              style: TextStyle(
                height: 6.5,
                fontSize: 10,
                backgroundColor: Colors.black45,
              ),
              textAlign: TextAlign.center,
              textScaleFactor: 0.85,
            ),
          ),
        ),
      ),
    );

    /*ClipRRect(
      borderRadius: BorderRadius.circular(100),
        child: Image.network(
          imageUrl,
          fit: BoxFit.cover,

        ),



    );*/
  }
}
```

```dart
import 'package:flutter/material.dart';

import '../screens/user_dogs_screen.dart';

class AppDrawer extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: Column(
        children: <Widget>[
          AppBar(
            title: Text('Hi Musher!'),
            automaticallyImplyLeading: false,
          ),
          Divider(),
          ListTile(
            title: Text('Manage Dogs'),
            onTap: () {
              Navigator.of(context)
                  .pushReplacementNamed(UserDogsScreen.routeName);
            },
          ),
          Divider(),
          ListTile(
            title: Text('Back'),
            onTap: () {
              Navigator.of(context)
                  .pushReplacementNamed('/');
            },
          ),
        ],
      ),
    );
  }
}
```