BACHELOROPPGAVE:

Appitude

FORFATTERE:
Bent Holden

DATO:

29.11.2018

Sammendrag av Bacheloroppgaven

Tittel:	**Appitude**
Dato:	29.11.2018
Deltakere:	Bent Holden
Veiledere:	Ivar Farup
Oppdragsgiver:	Appitude Lillehammer
Kontaktperson:	Ida Marie Frøseth, idafroseth@gmail.com, 97609772
Nøkkelord:	Appitude, Bacheloroppgave, Java, Andorid, Skiteknikk, Sensorer
Antall sider:	76
Antall vedlegg:	
Tilgjengelighet:	Åpen

Sammendrag: Denne oppgaven vil ta for seg utviklingen av en smarttelefon applikasjon. Applikasjonen sitt formål er å kunne se nærmere på bruk av eksterne sensorer til å måle og utlede statistiske verdier, fra en spesiell skiteknikk som kalles staking. Oppgaven vil presentere utviklingen både i form av arbeidsmetoder, samt gi innblikk i løsninger knyttet til applikasjonen.

Summary of Graduate Project

Title:	**Appitude**
Date:	29.11.2018
Participants:	Bent Holden
Supervisor:	Ivar Farup
Employer:	Appitude Lillehammer
Contact Person:	Ida Marie Frøseth, idafroseth@gmail.com, 97609772
Keywords:	Appitude, Bachelor thesis, Java, Android, Skiing technique, Sensors
Pages:	76
Attachments:	
Availability:	Open

Abstract: This thesis will take you through the development process for a mobile application. The purpose of the application is to take a closer look at the use of external sensors to measure statistical values from a special skiing technique referred to as “double poling“. The thesis will present both the development process, and give some insight of the solutions used in the application itself.

Forord

Jeg og min partner ble interessert i denne oppgaven veldig tidlig. Det som interesserte oss med oppgaven er at den tilbydde læring innen mange forskjellige temaer som er veldig populære i dagens samfunn. Etter et møte med oppdragsgiver, bestemte vi oss for å gå for denne oppgaven. Vi fikk til vår lykke tildelt oppgaven, men veldig tidlig ble det bestemt at min partner ikke kunne ta del i oppgaven, og jeg endte opp alene på prosjektet.

Jeg vil gjerne takke veilederen min Ivar Farup, som gjennom hele prosjektet har stilt opp med gode tilbakemeldinger på hva som er blitt gjort, samt ledet meg i riktig retning, både for det endelige produktet, samt for denne rapporten.

Jeg vil også takke Gjøvik Skiklubb, og Gjøvik Toten Langrenn for å ha satt meg i kontakt med en skiløper som var villig til å samle inn data for meg, jeg ønsker spesielt også å takke skiløperen som gjorde det mulig.

Sist men ikke minst vil jeg også takke oppdragsgiver, som ga meg muligheten til å ta fatt på denne reisen.

Innhold

Figurer

Tabeller

1 Innledning

1.1 Problemområde

Det har lenge vært mulig å kunne måle ytelsen til personer i ulike aktiviteter som for eksempel jogging, hvor man kan måle blant annet antall skritt, hastighet, distanse, også videre. Innenfor skisporten er staking blitt mer og mer populært, dette er hvor man tar et stavtak med begge staver samtidig.

Staking har en mangel på et verktøy som kan måle ytelsen av staketeknikken til en person, det ønsker Appitude å gjøre noe med. Jeg har derfor fått i oppdrag av Appitude å lage et verktøy som kan måle staketeknikk og ytelse. Verktøyet skal lages slik at en som driver med staking som hobby kan bruke det, like godt som profesjonelle.

1.2 Avgrensning

Grunnet at jeg er en enkelt person på dette prosjektet som egentlig var beregnet på to til tre personer, vil jeg ikke ha mulighet til å levere et godt gjennomført resultat av alt oppdragsgiver ønsker seg. Derfor har jeg blitt enig med oppdragsgiver om at omfanget av oppgaven avgrenses til at oppgaven skal handle om utviklingen av applikasjon til Android operativsystemet, håndtering av sensorer, samt prosessering av sensor data til å utlede nyttige verdier om staketeknikken til brukeren. Det kan også nevnes om at brukergrensesnittet til applikasjonen, vil bruke standard GUI elementer til det valgte programmeringsspråket.

Oppgaven har blitt avgrenset til disse målene fordi dette er noe som er mer realistisk at jeg som en person skal kunne utføre innenfor tidsrammene. Oppdragsgiver finner også disse målene mest matnyttig i oppgaven.

1.3 Oppgavedefinisjon

Denne oppgaven vil gå ut på å utvikle en applikasjon til Android-operativsystemet som skal kunne motta rådata fra eksterne sensorer festet til skistavene, dermed utlede og fremstille nyttige verdier for brukeren om en stakeøkt. Sensorene skal strømme rådata til applikasjonen i sanntid, samt er det også en del av oppgaven å kunne logge data fra sensorene, altså skal man kunne ta opp en økt som lagres på de eksterne sensorene, for så å laste disse inn i applikasjonen senere. Applikasjonen skal ha et brukergrensesnitt hvor brukeren skal kunne:

- Starte og stoppe en økt.
- Se tidligere økter.
- Laste inn loggede økter.
- Fremstille resultatet av en økt på en ryddig måte.

Avledede verdier som vil være av interesse er:

- Antall stavtak
- Frekvensen på stavtakene

- Vinkelen på skistavene da de treffer bakken
- Vinkelen på skistavene da de slipper bakken

Oppdraget kommer med allerede utvalgte tredjeparts sensorer, dette av typen MetaMotionR fra mbientlab, denne typen sensor kommer også med ferdig utviklet API som kan tas i bruk til å programmere sensorene.

1.4 Bagrunn for prosjekt

Bakgrunnen for prosjektet er at oppdragsgiver er aktiv innen ski, og har sett mangelen på et produkt som tilbyr akkurat det å måle ytelse innen staking. Det finnes alt i dag slike typer tilbud for mange idretter som for eksempel løping, hvor man har klokker, telefoner, og dedikerte aparater til å måle ytelsen til utøveren. Men dette er ikke noe som fungerer når det kommer til staking innen skisporten, konseptet er det samme, å kunne gjenkjenne bevegelsen til brukeren for å kartlegge ytelsen, men løsningene som allerede finnes, er ikke designet til å kunne gjenkjenne bevegelser for staking, og det er derfor ingen god løsning.

Med tanke på hvor populært, og konkurranse drevet skisporten er, er det helt garantert at det allerede er blitt gjort private studier og målinger av skistavenes bevegelse både innen vanlig skigåing, og staking. Men dette er ikke studier vi har tilgang til, eller løsninger alle kan ta nytte av, dette er noe de driver med bak kulissene når det gjelder de beste av de beste innen skigåing, for å optimalisere skiteknikker.

1.5 Målgruppe

Oppgaven vil inneholde to målgrupper, én for prosjektet og lærdommen som vil komme frem i denne rapporten, samt vil det være én for brukeren av det endelige produktet.

1.5.1 Prosjektet

Målgruppen for denne rapporten går for de som er interessert i android utvikling, håndtering av de eksterne sensorene som skal brukes, samt bruk av sensorer til å gjenkjenne bevegelser. Rapporten tar utgangspunkt i at leser har forståelse for enkel fysikk, samt er grunnleggende forståelse for programmering nødvendig for å kunne henge med i bitene som omhandler programmering.

1.5.2 Produktet

Det endelige produktet skal ha et formål for skigåere som fokuserer på staking. Produktet sin målgruppe er ikke begrenset av noen form for alder, kjønn eller dyktighet innenfor staking, men skal tilbys som et verktøy for de som ønsker å se statistiske data om stakeøkten sin.

1.6 Min bakgrunn

1.6.1 Personlig

Jeg har alltid vært interessert i data, teknologi og programvare siden jeg kunne gå, og har alltid vært frempå da det gjelder å lære ting på egenhånd, om det skulle være å programmere ulike løsninger for å se om det er noe som fungerer, eller om det skulle

være å lage spesifikke programmer for å kunne se oppførselen til forskjellige funksjoner. Dette vil synes senere i rapporten, da jeg har utviklet en applikasjon på siden, for å kunne forstå oppførselen til de eksterne sensorene som skal tas i bruk, ved hjelp av sanntids strømming inn til grafer som oppdaterer seg i sanntid.

1.6.2 Akademisk

Min akademiske bakgrunn inneholder til stor grad relevans for denne oppgaven. Jeg har selvfølgelig gjennomført alt av grunnskole, hvor jeg gikk videre til å utføre en treårig svakstrøms elektronikk utdannelse ved videregående skole. Etter skolen har jeg jobbet i praksis som telefon reparatør hos én av de største i bransjen i Norge. Her jobbet jeg hovedsaklig med Samsung smarttelefoner, som holder Android som sitt faste operativ-system på nesten alle sine modeller, med unntak av et par som bruker Window's sitt operativsystem. Dette jobbet jeg med i nesten to år, og gikk ut med et fagbrev innen svakstrøms elektronikk. Videre herfra har jeg tatt fatt på min videre utdanning som data-ingeniør her på NTNU Gjøvik, hvor jeg har fått gode kunnskaper innen programmering, og herdet min evne til selvlære, som jeg mener har vært essensielt for å kunne få til denne oppgaven på egenhånd.

1.7 Systemutviklingsmetode

Originalt da jeg gikk inn i planleggingen av dette prosjektet, var jeg ikke fult så opptatt av å ta i bruk en spesifikk systemutviklings metode, jeg er og var alene om prosjektet, som også betyr at jeg hadde ansvar for at alt som skulle bli gjort. Men etter en diskusjon med veileder om dette, bestemte jeg meg for å benytte meg av en systemutviklingsmetode, det kan sies at både jeg og veileder ikke så hele behovet for å benytte en, da dette er mer noe som brukes for en større gruppe arbeidere, for å holde struktur i fremgangen av prosjektet. Men jeg brukte en, for å kunne ta til meg lærdommen av å jobbe med en systemutviklingsmodell i praksis, samt holde styr på en timeplan om hva som skulle bli gjort i hvilken rekkefølge.

Etter diskusjon med veileder, ble jeg fort overtalt til fordelene med en inkrementell systemutviklings modell. Prosjektet går ut på å utvikle programvare, og jeg kan ikke være sikker på at oppdragsgivers krav endres under utviklingsperioden, samt er det lettere å kunne levere programvare som oppdragsgiver vil ha, da man kan vise frem produktet til oppdragsgiver underveis, å få tilbakemeldinger på hva som ønskes.

En inkrementell modell har også én stor fordel fremfor en plandrevet modell. Hvis jeg skulle valgt en plandrevet modell, ville jeg endt med å bruke store deler av start perioden, bare med å planlegge. Dette vil si at hvis jeg skulle komme utfor noe som var uforutsett i denne modellen, vil dette kunne sette meg langt bak skjema, og skape stor risiko for å ikke fullføre prosjektet i tide. Jeg har av disse grunner derfor valgt å implementere en form for Scrum.

Som man også kan se i min planelegging, ønsket jeg også å bruke en test-drevet utvikling (TDD) for å ikke introdusere nye feil i applikasjonen etter oppdateringer, men her måtte jeg se meg selv slått, dette var for mye arbeid å introdusere inn i mitt prosjekt, og jeg så ikke tiden til å gjøre det.

1.8 Implementasjon

Som sagt, skal jeg anvende scrum, jeg velger å gjøre det på en ganske tradisjonell måte, men noen endringer må gjøres. Siden jeg er én person, vil jeg ta rollen som både Scrum-Master og utvikler. Jeg velger å gi rollen som product owner til oppdragsgiver.

Mine sprint perioder er satt til en lengde på to uker, altså 10 arbeidsdager. Jeg vil måle fremgang i sprinten og vurdere hva som bør jobbes på hver dag, dette vil være som en erstatning for daglige Scrum møter. Produkt backlog ble satt opp ved starten av prosjekt perioden, hvor jeg benyttet meg av veileder sin hjelp til å få et bedre estimat på oppgavene. Produkt backlog har hatt en arbeidsmengde på rundt 80 arbeidtimer. Sprint backlog har jeg hovedsaklig styrt selv, men jeg har prøvd å holde oppdragsgiver sine prioriteringer på de forskjellige oppgavene.

1.9 Fremdriftsplan

Under planleggingen til prosjektet, satt jeg opp et gantt diagram for min forventede progresjon. Dette gantt diagrammet var bare en oversikt over sprintene, samt forskjellige milepæler utover utviklingperioden. Hva sprintene skulle inneholde står beskrevet i milepælene ovenfor diagrammet. Dette diagramet kan ses på side 13 i min planleggings rapport (Appendiks B). Milepælene inneholder i stor grad delmål på et høytnivå, som senere kunne deles opp til en sprint backlog jeg kunne bruke å jobbe etter. Vi kan også se at testing har blitt satt mot de to siste sprintene, men i realiteten har testing vært en aktivitet gjennom hele utviklingsprosessen, siden jeg har testet alt jeg har laget, mens jeg har laget det. Vi kan også se at Rapport skriving har tatt del gjennom hele prosjektet, og det har det med at jeg daglig har skrevet logg om hva jeg har gjort, slik at jeg kunne gå tilbake, å se gjennom hele utvikligsprossesen min.

1.10 Roller

Det var i første omgang meningen at vi skulle være to personer på dette prosjektet, men etter noen uheldige omstendigheter endte det med at jeg måtte ta for meg prosjektet på egenhånd. Jeg har derfor veiet mye på veileder og oppdragsgiver for deres meninger i forskjellige beslutninger jeg har måtte foretatt underveis.

1.11 Dokument struktur

1. Introduksjon - I dette kapittelet kan man lese om bagrunnen min, bakgrunnen til prosjektet, hva prosjektet går ut på, og rammeverket som har blitt brukt for dette prosjektet.

2. Spesifikasjon - Dette kapittelet viser brukertilfeller, drøfter og forklarer teknologier og utstyr som har blitt brukt i dette prosjektet.

3. Data analyse - Her i dette kapittelet viser jeg hvordan rådata ser ut, hvordan sensorene oppfører seg, planer for å samle inn test data, samt et analyse av innsamlet data.

4. Utviklingsprosess - I dette kapittelet snakker jeg litt om min planlagte utviklings-

prosess.

5. Design - Dette kapittelet går over designet og strukturen på applikasjonen.
6. Utførelse - Dette kapittelet handler om utførelsen av prosjektet, programmeringen, utviklingen av algoritme, tester, utfordringer og generelt hvordan alt i prosjektet har blitt utført.

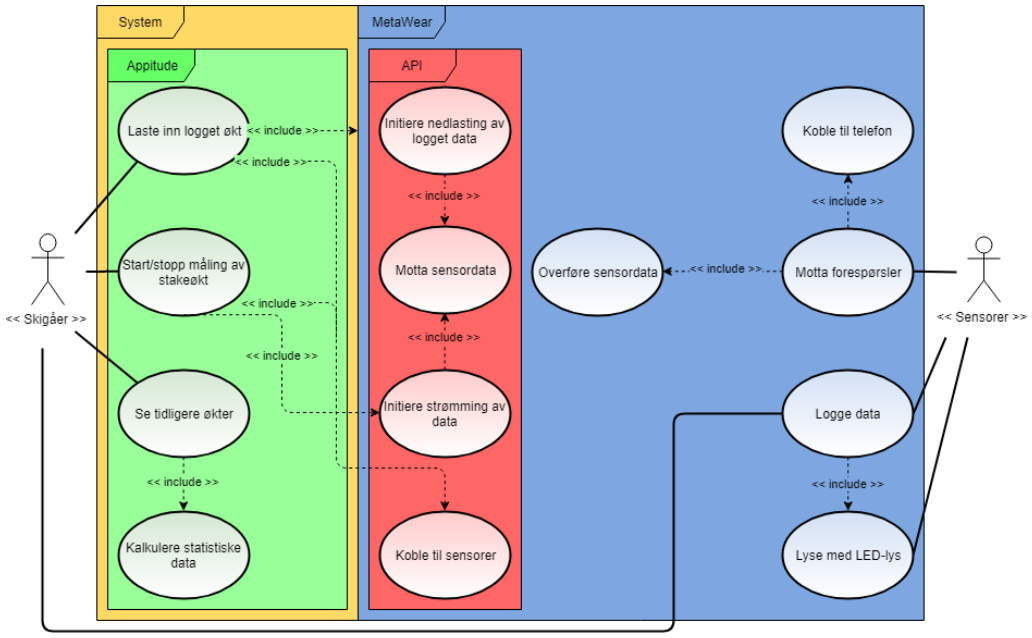
7. Diskusjon - I dette kapittelet vurderer jeg og diskuterer mitt eget arbeid, og snakker litt om hva jeg ville gjort annerledes, samt diskuterer ting som har vært utfordrende.
8. Konklusjon - I dette kapittelet trekker jeg en konklusjon over prosjektet, samt diskuterer jeg litt om fremtiden til mitt arbeid.

9. Appendikser

2 Spesifikasjon

2.1 Use case modell

Applikasjonen skal i hovedsak kunne koble telefonen opp mot sensorene, for mottak av sensordata, dermed i ettertid kunne utlede statistiske data. I denne prosessen, innebærer det en rekke ting som må gjøres, blant annet interaksjon mellom bruker, og kommunikasjon mellom mobiltelefon og sensorer. I use case diagrammet som ses på figur 1, kan man se en oversikt over hva applikasjonen skal kunne gjøre, samt funksjoner som hører til sensorene. Selv om jeg må stille inn innstillingene på sensorene i form av kode for å få dem til å fungere på ønsket måte, anser jeg ikke dette som en del av mitt system, og det er derfor skilt i to forskjellige grupper. Jeg kommer ikke til å dekke use casene som hører til sensorene, siden dette er noe jeg ikke har laget, og jeg vil ikke kunne svare på spørsmål om funksjonaliteten til disse i store detaljer.



Figur 1: Forkjellige use cases

2.2 Høynivå use case

I tabell 1 til tabell 4, vises en høynivå beskrivelse av use casene for figur 1 i mitt system. Selv om jeg har sagt at jeg ikke kommer til å dekke use casene knyttet til sensorene, siden de ikke er del av mitt system, er det et enkelt use case jeg ønsker å dekke herfra, dette er use caset «Logge data». Jeg ønsker å ha med dette spesifikke use caset fordi det står veldig sentralt av hva det endelige produktet skal kunne tilby.

Use case:	Se tidligere økter
Aktør:	Skigåer
Beskrivelse:	1. Skigåer navigerer til «View Sessions». 2. Skigåer velger en økt. 3. Skigåer velger statiske data som vises: • Frekvensen av stavtak. • Vinkel på skistavene da de treffer bakken. • Vinkel på skistavene da de slipper bakken.
Variasjon:	Ingen

Tabell 1: Høynivå use case av «Se tidligere økter»

Use case:	Start/stopp måling av stakeøkt
Aktør:	Skigåer
Beskrivelse:	1. Skigåer navigerer til «Start Session» (start skjerm). 2. Skigåer velger «Start Session» knapp. 3. Skigåer huker av på ønskede sensorer og trykker «Ok» 4. «Start Session» knapp byttes ut med «Stop Session» knapp 5. Skigåer trykker på «Stop Session» knapp, økten avsluttes og knappen går tilbake til «Start Session».
Variasjon:	• Hvis skigåer ikke finner sensorene, kan man søke på nytt. • Hvis skigåer angrer og ikke vil starte, kan man avbryte.

Tabell 2: Høynivå use case av «Start/stopp måling av stakeøkt»

Use case:	Laste inn logget økt
Aktør:	Skigåer
Beskrivelse:	1. Skigåer navigerer til «View Sessions». 2. Skigåer velger «Load Session»-knapp. 3. Skigåer huker av på ønskede sensorer hvor øktene ligger og trykker «Ok».
Variasjon:	Hvis skigåer ikke finner sensorene, kan man søke på nytt. Hvis skigåer angrer og ikke vil laste inn, kan man avbryte.

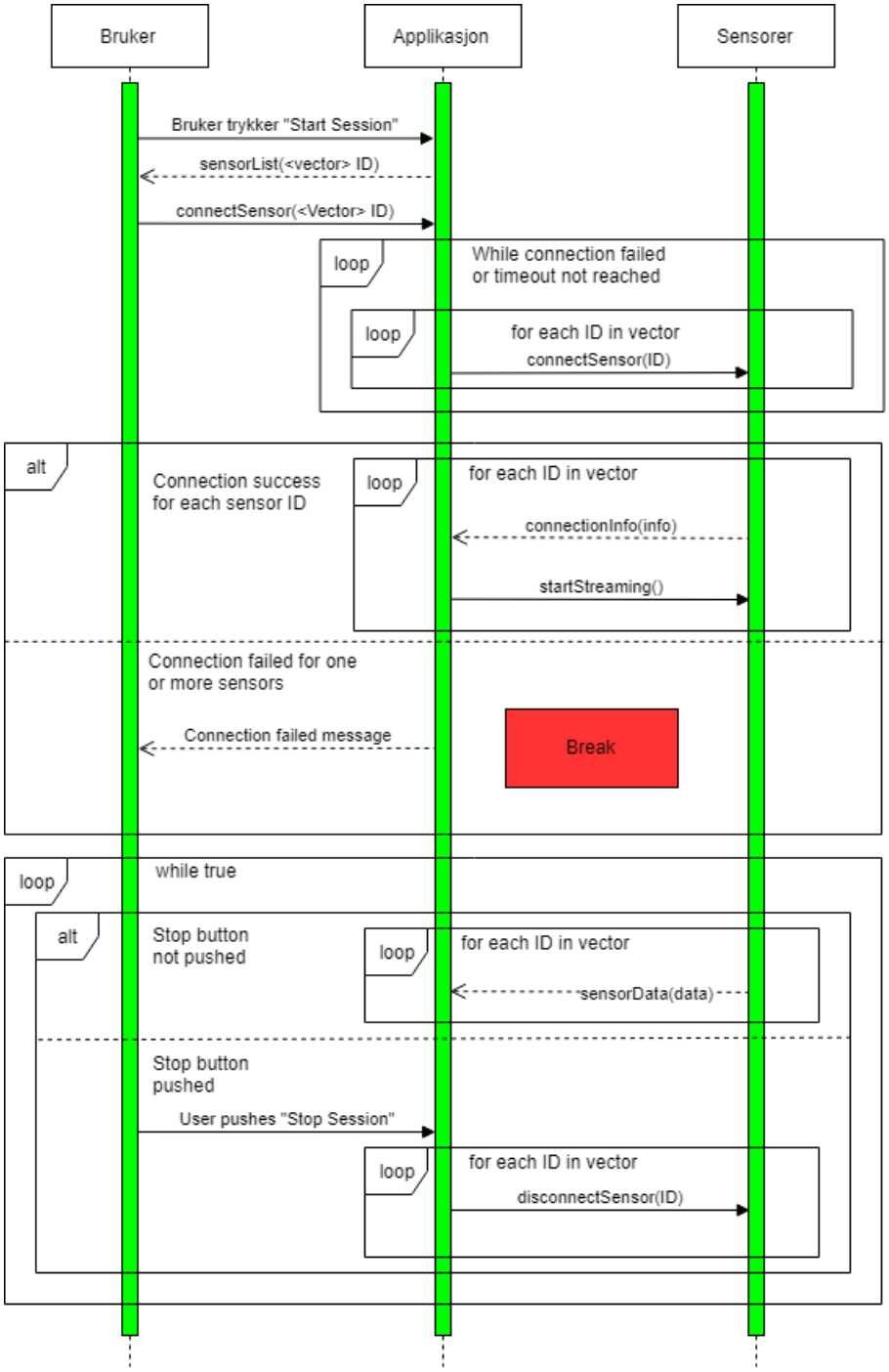
Tabell 3: Høynivå use case av «Laste inn logget økt»

Use case:	Logge data
Aktør:	Skigåer, Sensorer
Beskrivelse:	1. Bruker trykker på knapp på sensor for å starte. 2. LED-lys gir et grønt blink for indikasjon om at logging har startet. 3. Skigåer trykker på samme knapp igjen for å stoppe. 4. LED-lys gir et rødt blink for indikasjon om at logging har stanset.
Variasjon:	Ingen

Tabell 4: Høynivå use case av «Logge data»

2.3 Sekvensdiagram

For å vise litt av hvordan kommunikasjonen i systemet skal fungere, har jeg laget et sekvensdigram som kan ses på figur 2. Dette sekvensdiagrammet viser kommunikasjon mellom bruker, applikasjon og sensorer. Dette sekvensdiagrammet gjelder for da en bruker skal starte økt, frem til brukeren stopper økten.



Figur 2: Sekvensdiagram fra da en bruker starter en økt, til brukeren avslutter økten.

2.4 Sensorene

I oppdraget hadde det allerede blitt valgt sensorer på forhånd. Sensorene som er valgt er to stykker MetaMotionR[1] brett fra mbientlab. Dette er en samling av sensorer med

[1] https://mbientlab.com/product/metamotionr/

allerede installert firmware og tilhørende API, slik at alt som behøves for å ta i bruk sensorene på brettet, er å koble til brettene, samt sende forespørsler til brettene om hva som skal gjøres.

2.4.1 Akselerometer

Akselerometeret har 3-akser, dette vil si at vi får akselerasjonen i form av xyz-komponenter, i både negativ og positiv retning. Akselerasjonen måles i antall G-krefter som er multipler av jordas tyngde akselerasjon(ca 9.81 m/s per G), og kan måles opp til ± 16G.

2.4.2 Gyroskop

Gyroskopet i dette tilfellet har også 3-akser, og vi vil få vinkelhastigheten i form av xyz-komponenter i både negativ og positiv retning. Vinkelhastigheten måles i form av $^\circ/s$. Vinkelhastigheten kan måles i opp til $\pm 2000^\circ/s$

2.4.3 Sensor fusion

Sensor fusion er en funksjon som allerede er laget og lagt inn i sensorene fra produksjon. Sensor fusion er hvor man kombinerer data fra flere sensorer, til å skape mer nøyaktig data. I tilfellet med disse sensorene, kombineres data fra akselerometer, gyroskop, og magnetometer i ulike kombinasjoner avhengig av hvilke modus man bruker på sensorene.

Jeg har valgt å bruke sensor fusion, jeg har valgt å bruke det fordi dette er en funksjon som allerede eksisterer på sensorene og vil gi bedre resultater, samt gir det meg tilgang til andre data som er derivert fra dataene man originalt får ut av sensorene.

Sensor fusion vil gi meg tilgang til en korrigert akselerasjon, som vil fungere på samme måte slik som beskrevet i beskrivelsen av akselerometeret, men vil være mer nøyaktig enn hvis jeg skulle hentet det rett fra akselerometeret.

Sensor fusion gir meg også tilgang til den deriverte datatypen Euler-vinkler, her benyttes det de samme betegnelsene som ved fly [1], altså har vi:

- Roll.
- Pitch.
- Yaw.
- Heading.

Jeg er mest interessert i roll på grunn av orienteringen jeg har valgt å feste sensorene til skistaven. Roll måles i vinkler i området $-90^\circ \leq \texttt{roll} \leq 90^\circ$, hvor 0° er at skistaven er horisontal. Ved min orientering, vil det gi utslag for negative verdier da staven er riktig vei, og positive utslag da staven er oppned, det vil være mellom $0^\circ \pm 90^\circ$ avhengig av hvilke side som er nærmest horisontal stilling, dette illustreres bedre senere i dette kapittelet.

2.4.4 Knapp og LED-lys

Sensorene har en mekanisk knapp, som kan programmeres med API'et som følger med sensorene. Her kan man lage en funksjon som lagres på sensorene, som kjøres da man trykker på knappen. Her er da planen å bruke denne knappen til å starte og stoppe logging av data, da man ikke har en smarttelefon til å sanntidsstrømme til. Sensorene kommer også med et LED-lys. LED-lyset kan fungere som en indikator man kan sette til

å lyse og blinke da spesifikke hendelser oppstår. Jeg har planer om å bruke knappen til å starte og stoppe logging av data, samt LED-lyset som en indikator på at starting og stopping av logging har forekommet.

2.4.5 Bluetooth low energy

Sensorene kommuniserer ved bruk av Bluetooth low energy (BLE) er en avledet versjon av vanlig bluetooth. Denne versjonen er laget med hensikt for applikasjoner innen trening og helse, og er laget for å kunne oppnå samme kommunikasjons distanse som vanlig bluetooth, men med mindre strømforbruk.

Med mindre strømforbruk fører det også med kostnader, her er kostnaden overføringshastighetene, som er blitt redusert veldig mye i forhold til vanlig bluetooth. Overføringshastighetene vil være avhengig av innstillingene på tilkoblings intervallene, som er hvor lenge man er tilkoblet og mottar pakker omgangen.

Det er ulikt hva man har av muligheter på dagens telefoner, for referanse, har IPhone 6 bare mulighet til intervaller på 30 millisekunder, som er veldig dårlig, og Nexus 4 har mulighet for opp til 7.5 millisekunder, som er det beste på markedet. Men alt dette avhenger også på hva sensorene støtter av intervaller, og de støtter heldigvis den beste på 7.5 millisekunder. Det som skjer da man reduserer tilkoblings intervallet, er at sensorene slår sammen 3 data målinger til en BLE pakke, og dermed økes overføringshastighetene enormt.[1]

Ved 30 millisekunder intervaller, kan vi forvente hastigheter rundt 2.6kB/s, som er svært lite, i motsetning til det beste hvor vi har 7.5 millisekunders intervaller på rundt 16kB/s, som er en stor forbedring, men fremdeles ikke imponerende i dette årstallet.

Én av to filer ved testingene jeg har gjort, er på 2089kB, den andre filen er rundt like store, så denne test målingen på 6.5 minutter har en total filstørrelse på rundt 4200kB for hver av sensorene. Teknologien kan oppnå full overføringshastighet på flere tilkoblinger samtidig. Tallene på filstørrelsene og overføringshastighetene begynte å skremme meg tidlig i prosjektet, og dette vil jeg diskutere senere i rapporten.

2.4.6 Andre ting av interesse

Andre ting som er av interesse på disse brettene, er:

- 8MB flash minne.
- Oppladbart batteri.
- Step detection.

Flash minne vil være et absolutt krav, siden jeg vil trenge et sted å lagre den loggede dataen. Man kan jo også tenke seg at siden det er snakk om flash minne, vil man miste all data hvis brettene skulle gå tom for strøm, men her er det snakk om NOR-minne (en form for non-volatile minne), som fungerer slik at den ikke mister data hvis den ikke lenger skulle få strøm.

At en form for batteri er logisk at jeg kom til å trenge på sensorene. I dette tilfellet er det snakk om oppladbart 100mAH li-ion, som skal kunne vare fra 2 dager til 2 uker ifølge produsenten. Batteriet kan lades opp ved hjelp av USB micro B porten som er på sensoren.

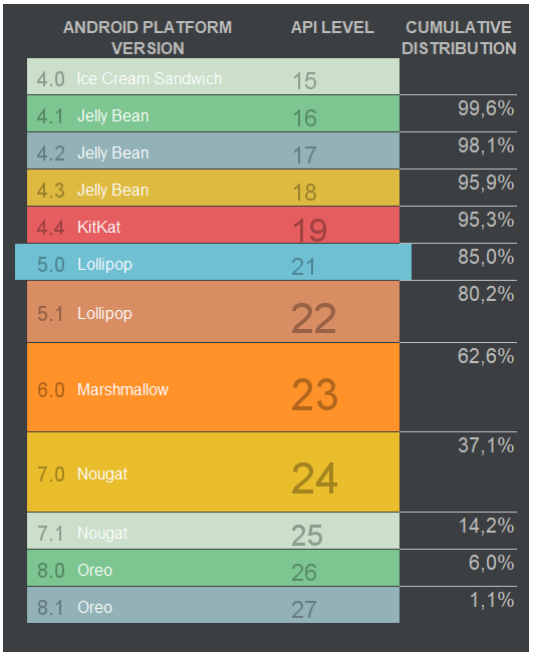
Brettene kommer også med en algoritme for gjenkjennelse av steg. Tanken på at

[1] bluetooth.https://punchthrough.com/pt-blog-post/maximizing-ble-throughput-on-ios-and-android/

denne kan være relevant, er at hvis jeg modifiserer denne algoritmen, vil den kanskje kunne være i stand til å gjenkjenne et stavtak. Senere i rapporten, vil jeg teste disse funksjonene og beskrive hvilke roller de vil få i det endelige produktet.

2.5 Operativsystem

Android er et operativsystem som har vært tilgjengelig de 10 siste årene, og for hver nye oppdatering, slippes nye funksjoner ut, etter utviklingen av smarttelefoner sine behov. I figur 3 vises den kumulativ distribusjonen av de forskjellige Android versjonene opp til Oreo 8.1.



Figur 3: Kumulativ distribusjon av Android versjoner.

MetamotionR sensorene som skal brukes, krever minst en Android versjon på 4.4 Kitkat, dette er fordi det er først i denne versjonen Bluetooth Low Energy introduseres for Android verden. Spørsmålet er om det vil lønne seg å bruke versjon 4.4 Kitkat, eller om jeg burde bruke en senere versjon.

Med så og si alt av store programvare produkter her i verden, så finnes det feil, disse feilene blir fikset og forbedret i nyere oppdateringer, og det samme gjelder for Android. Versjon 4.4 Kitkat har dårlige optimaliserte funksjoner for oppdaging av bluetooth enheter, og derfor ble biblioteket for bluetooth oppdaging i versjon 4.4 Kitkat deprecated ved utgivelse av Android 5.0 Lollipop, som i Android verden vil si at de vil fortsette å støtte dette biblioteket, men det finnes bedre løsninger.

Etter versjon 5.0, vil det ikke komme noen funksjoner som i stor grad er relevante til mitt prosjekt. Men vi ser også i figur 2 at ved hoppet fra 4.4 Kitkat til 5.0 Lollipop, går den kumulative distribusjonen ned 15.3%. Min løsning er at jeg lager til slik at smarttelefoner som bruker 4.4 Kitkat, bruker biblioteket som er deprecated, siden de ikke har tilgang til det nye biblioteket i 5.0 Lollipop, og smarttelefoner som har 5.0 Lollipop eller nyere, får goden av å bruke det nye biblioteket. På denne måten når applikasjonen et så bredt

publikum som mulig, samt vil de som har mulighet få tilgang til nyere og bedre løsninger.

2.6 Programmeringsspråk

Kravet som står til oppgaven, er at applikasjonen skal fungere på Android operativsystemet. Dette kravet gir mange forskjellige valgmuligheter. Men jeg har også noen begrensninger, siden API'et til sensorene som brukes, bare støtter noen programmeringsspråk, dette er følgende språk som støttes av både API og Android:

- Java
- Swift
- C
- C++
- Python
- JavaScript

Java har noen sterke fordeler når det kommer til Android. Java er det eneste av programmeringsspråkene nevnt ovenfor som støttes av Google (utvikler av Android), som vil si at det er dette programmeringsspråket som gis mest støtte i form av dokumentasjon og hjelp, ikke bare av Google, men også av andre Android utviklere. Java har også enda en klar fordel, og det er «Android Studio», dette er et integrert utviklingsmiljø (IDE) laget av Google i sammen med IntelliJ, som er en av de største innen IDE'er for Java for å kunne levere en IDE spesifikt designet for Android utvikling. Samt har Java et stort utvalg av biblioteker som er lisensiert med åpen kilde. Ulemper som følger med Java er at det har påtatt seg et rykte[1] om at det ligger bak de andre programmeringsspråkene ytelsesmessig

men jeg har ikke klart finne en sikker kilde som kan bekrefte dette. Men et poeng verdt å notere seg er at Java kompilerer i en VM (Virtuell maskin) på Android, men det er påvist i flere tilfeller at en VM skal kunne oppnå «native» ytelse [2], selv om dette kan være avhengig av flere faktorer.

Swift bare støtter Android versjon 5.0 Lollipop og høyere, og er derfor uaktuell siden jeg utvikler med forbehold om at applikasjonen skal fungere på 4.4 Kitkat.

C er et mer moderne programmeringsspråk, og har en stor fordel med at det kompilerer på tvers av alle platformer ved hjelp av Xamarin, som er et oversettingsverktøy som støttes av blant annet Android, IOS, og Windows. C kompilerer koden til maskin kode, og utfører koden rett på hardware. Ulemper med å bruke C på Android er at det ikke har like stor støtte på Android platformen som Java har, og har begrenset tilgang til biblioteker lisensiert med «åpen kilde».[2]

C++ har en stor forskjell ovenfor de andre programmeringsspråkene jeg har diskutert her, og det er mengden kontroll dette programmeringsspråket gir deg over ressursene. Med dette språket har man full kontroll over minne, i motsetning til Java og C som benytter seg av Garbage collectorsom er at den rydder i minnet nå og da. Full kontroll over minne kan ses på som både negativt og positivt, det positive med det er at man kan optimalisere applikasjonen bedre, men dette kommer med den negative prisen med at det blir endel ekstra arbeid, og at det er større risiko for å introdusere feil.

Python er ikke et «native» programmeringsspråk til mobiltelefoner, altså er det ikke noe som er laget med mobiltelefoner i tankene. Dette gjør at programmeringsspråket

[1] https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/

[2] https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/

blir sett på som svakt[3] av applikasjonsutviklere innen mobiltelefoner. Siden programmeringsspråket ikke er rettet mot mobiltelefoner, er det heller ikke optimalt å bruke. Dette programmeringsspråket ligger bak på alle punkter, bortsett fra sitt enorme utvalg av biblioteker.

JavaScript er hvis man skulle ønske å lage en web-basert applikasjon til Android. Fordelen her er at det blir en veldig lettvekt applikasjon, som kan bruker overalt hvor JavaScript er implementert, som nesten er overalt nå i dag. Ulempen er at applikasjonen ikke vil yte like godt som ved alternativene, samt vil man ikke ha tilgang til alle funksjoner.

Jeg har i dette prosjektet valgt å bruke Java av flere grunner. Den største grunnen er helt klart mengden dokumentasjon og støtte dette programmeringsspråket tilbyr, som gjør det gunstig for meg siden det er første gang jeg utvikler en Android applikasjon. De andre programmeringsspråkene sine salgstriks er bedre ytelse, og multi-platform muligheter, ytelse er noe alle applikasjoner kan ha bruk for, men Java vil tilby mer enn nok ytelse for mitt prosjekt, og jeg prioriterer ikke multi-platform kompatibilitet siden prosjektet er avgrenset til å bare tilbys på Android. Det er også en stor fordel at jeg har erfaring med Java fra utdanningen min her på NTNU, slik at programmeringsspråket ikke er helt fremmed for meg.

2.7 Verktøy

Til å programmere har jeg valgt å bruke Android Studio, som nevnt ovenfor er dette et utviklingsmiljø (IDE) utviklet av Google og IntelliJ spesifikt til å programmere til android. Jeg velger å bruke denne IDE'en fordi den tilbyr god oversikt og støtte for akkurat det å programmere Java til android, og er rett og slett det beste valget for IDE for mitt bruk.

Jeg har valgt å bruke BitBucket som repository, dette er et valg jeg kan få gratis gjennom NTNU. Det ville ikke vært krise for min del, å ikke bruke repository siden jeg er alene på prosjektet. Jeg velger å bruke et repository slik at eventuelt oppdragsgiver og veileder kan følge progresjonen, samt at det tilbyr meg en backup av prosjektet, og gjør det mulig for meg å jobbe fra andre datamaskiner. En annen ting er at det også gir meg god oversikt over fremgang i prosjektet, ved at jeg kan se alle oppdateringer som forekommer, samt vil jeg kunne gå tilbake til tidligere versjoner, for eventuell debugging.

Trello er integrert i BitBucket, og jeg har valgt å bruke dette verktøyet av den grunn til å holde styr på produkt backlog, og sprint backlog.

Jeg velger å bruke Latex til å skrive bachelorrapporten, dette fordi Latex er dominerende når det gjelder muligheter for hva man kan gjøre, samt at veileder og andre lærere på NTNU, anbefaler sterkt å bruke Latex. Jeg velger å bruke Latex i form av «Overleaf», som er en online versjon av latex, slik at jeg får en sikkerhetskopi, samt muligheten for å jobbe fra andre datamaskiner.

2.8 Dokumentasjon og internasjonalisering

Jeg vil bruke JavaDoc, s støttes fult ut i Android Studio, så jeg velger å bruke JavaDoc samt kommentering for å dokumentere kilde koden. Javadoc er en svært gunstig måte å dokumentere kilde kode på, ved at man lett har tilang til godt oversiktlig dokumentasjo-

[3] https://www.netguru.co/blog/python-pros-and-cons-what-are-the-benefits-and-downsides-of-the-programming-language

nen for den koden man jobber på. De fleste andre IDE'er for Java støtter også JavaDoc fult ut som for eksempel Eclipse og IntelliJ, samt kan man få tilgang til dokumentet ved hjelp av en nettleser.

JavaDoc og kommentering av koden har skjedd mens koden har blitt skrevet, slik at det ikke har samlet seg opp mye udokumentert kode, som jeg nødvendigvis ikke har kunnet huske hva gjør i ettertid.

Applikasjonen skal bli internasjonalisert på norsk og engelsk, jeg velger å ta med dette fordi det er veldig lite som skal til for å få med denne biten, og sparer tid hvis prosjektet skulle videre utvikles, siden de ikke trenger å gå gjennom hele koden og endre til at det blir internasjonalisert. Det gir også muligheten for å simpelt kunne legge inn nye språk, og alt som trengs er oversetting.

2.9 Lisenser

Når det gjelder lisens på selve applikasjonen, så er det helt opp til oppdragsgiver. Jeg har hatt problemer med å få et svar på hvordan type lisensiering som ønskes på applikasjonen, som har avgrenset meg i hvilke biblioteker jeg har kunnet bruke. Siden jeg ikke har fått noe definitivt svar, har jeg valgt å bare inkludere biblioteker med åpen kildekode lisensiering, slik at oppdragsgiver står fritt til å velge hvilke type lisens som ønskes på applikasjonen i fremtiden, uten å få noen problemer med lovens lange arm.

Når det kommer til gratis biblioteker, er det to kjente kategorier [3], den ene går under navnet «Copyleft», som innen programvare vil si at man kan bruke, modifisere og distribuere fritt, på den ene betingelsen at det distribueres med samme betingelsene. Et godt eksempel på dette er GPL (General Public License), som var den første Copyleft lisensen som så mye bruk, og er fremdeles dominant når det kommer til copyleft. Et eksempel på bruk av denne type lisensiering, er kjernen til Linux operativsystemet, som bruker GPL lisens. Jeg vil helst unngå å bruke lisenser som går innenfor kategorien Copyleft, fordi det kan tvinge oppdragsgiver opp i et hjørne da det kommer til lisensiering av applikasjonen som lages i dette prosjektet.

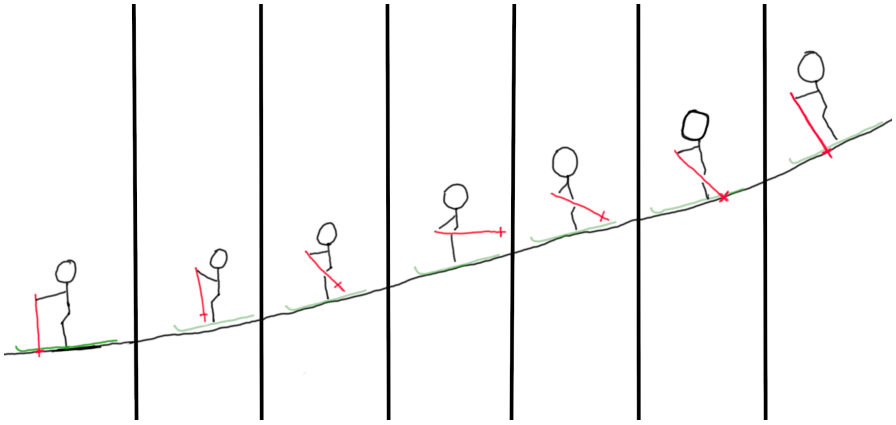
Den andre store kategorien som også finnes, går under BSD-lignende (Berkeley Software Distribution) lisenser, som hovedsaklig vil si at man kan gjøre akkurat det man vil når det kommer til bruk, modifisering og distribusjon, men man kan ikke holde personen som har laget det ansvarlig på noen måte. Et eksempel her er MIT (Massachusetts Institute of Technology) lisensiering, som er den største lisensen innen denne kategorien. Et eksempel på bruk av denne type lisensiering er programmeringspråket LUA, som bruker MIT lisens.

Det finnes utrolig mange typer når det kommer til åpen kildekode lisensiering, samt går det å kombinere ulike typer lisenser. Jeg kommer ikke til å gå gjennom alle typer lisenser og kombinasjoner av lisenser her, men jeg vil opplyse om at dette er noe jeg har hatt i tankene da jeg har lett etter biblioteker jeg skal bruke. Jeg kommer til å gå mer over lisensen til bibliotekene jeg har brukt, da jeg introduserer bibliotekene senere i rapporten.

3 Dataanalyse

3.1 Definisjon av et staketak

Et staketak er hvor man bruker begge staver samtidig[1] til å bevege seg fremover på ski. I figur 5 kan man se bevegelsen til skistaven i et staketak.



Figur 4: Bevegelse av skistav i et staketak

3.2 Innsamling av data

For innsamling av data trengte jeg en skiløper, mine kriterier for skiløper var følgende:

- Skal kunne beherske god staketeknikk.
- være i alderen 16+.

Jeg tok derfor kontakt med Gjøvik-toten langrenn, som videreførte meg til Gjøvik skiklubb, som satte meg i kontakt med en skiløper som var villig til å ta oppdraget. Grunnen til mine kriterier er at jeg ønsker å jobbe ut fra test data som jeg vet er pålitelige, slik at jeg har gode data å jobbe ut fra i resten av prosjektet.

For skiterreng, trengte jeg et sted med variert terreng, flatmark, oppoverbakker, og nedoverbakker. Siden jeg ikke er så kjent i området, foreslo test skiløperen Vind idrettsplass, som var det perfekte stedet med godt variert terreng. Mine kriterier for terreng er fordi jeg må vite om det er forskjell i data i forskjellig terreng.

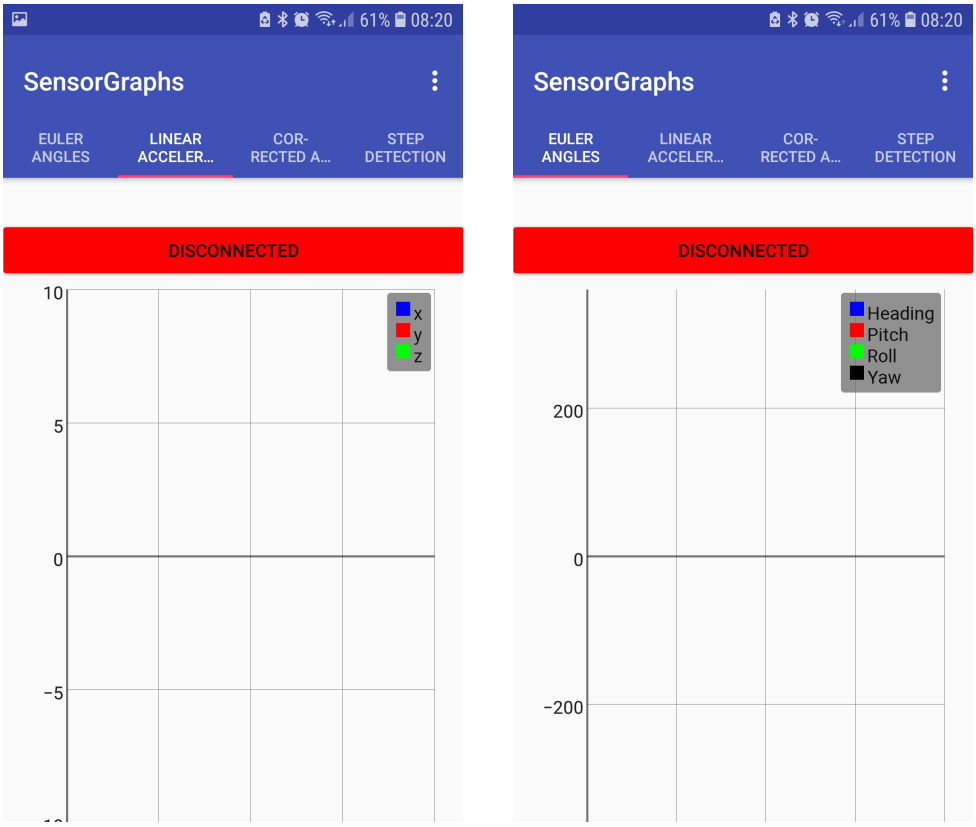
I forkant av innsamlingen, hadde jeg kommet såpass langt i utviklingen at jeg kunne bruke min egen applikasjon til innsamling av data. Denne applikasjonen lagret da dataene i en tekst fil, oppsatt slik at jeg lett kunne lime det inn i et plotte program for å se på dataene. Applikasjonen kunne lett bytte mellom frekvensen på målingene, altså målinger per sekund, og jeg foretok målinger i 10hz, 20hz, 30hz, og 100hz. Grunnen til at jeg valgte å måle i flere frekvenser, var fordi jeg vil ikke håndtere mer data enn jeg trenger, og måtte se om lavere frekvenser kunne levere et fyldig nok resultat til at jeg

[1] https://www.youtube.com/watch?v=K9Vk8AHTP9Q

kunne jobbe med det.

3.3 Tilleggs applikasjon

For å bedre forstå dataene jeg har anskaffet, trengte jeg en applikasjon som kunne vise meg sensor dataene i sanntid. Denne typen applikasjon er noe som allerede finnes til andre type sensorer som for eksempel mobiltelefonen sine sensorer, men det fantes ikke noe rettet til sensorene jeg skulle bruke (men produsenten ga ut dette noen uker etter jeg hadde laget denne). Jeg har jeg laget en tilleggs applikasjon, denne applikasjonen kommer jeg ikke til å dekke i denne rapporten, bortsett fra hvordan den har hjulpet meg, og hva den gjør. Denne tilleggs applikasjonen er laget slik at jeg kan se hvordan sensorene oppfører seg i sanntid. Dette gjøres ved at jeg oppretter en strømming av data fra sensorene til smarttelefonen, som deretter plottes direkte inn i en graf som oppdaterer seg når den får nye data. Dette gjør at jeg kan se for eksempel akselerasjonen til sensoren i sanntid. Figur 5 gir et kjapt innblikk i tilleggs applikasjonen. Det er verdt å merke seg at ved akselerasjonen, er det X-aksen som er av interesse, og ved vinklene er det roll.



(a) Akselerasjonen.

(b) Euler vinkler.

Figur 5: Tilleggs applikasjon

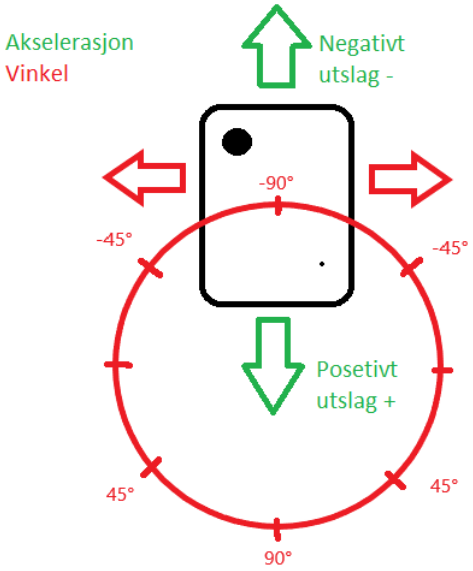
For grafene her, har jeg benyttet meg av biblioteket MPAndroidChart[1], dette bibloteket gir høyest konfigurerbare grafer, som jeg konfigurerte til å oppdatere seg hver gang det

[1] https://github.com/PhilJay/MPAndroidChart

ble plottet inn nye data. Dette biblioteket er lisensiert under Apache versjon 2.0 lisens [4], som er en BSD-lignende lisens som ble diskutert tidligere.

Ellers er det benyttet standard Java biblioteker, og API'et[2] for sensorene som er lisensiert med Copyright MbientLab Inc, her er det snakk om copyright, men de lar deg modifisere, kopiere, og distribuere gitt at det brukes i sammenheng med et av deres produkter.

Med tilleggs applikasjonen har jeg kunnet finne ut oppførselen til sensorene, som vist i figur 6. Her ser vi retningene på sensorene, og hvilke type utslag vi er ute etter. Vinkelen vil gå fra $0°$ til $-90°$ avhengig av hvilke side som er nærmest bakken, vi forventer derimot ikke at en skiløper skal holde stavene oppned og skape positive utfall, derfor er det bare de negative utfallene for vinklene vi er ute etter. Ved $0°$, vil sensoren ligge horisontalt rotert enten til venstre, eller høyre.



Figur 6: Retning på aksene i forhold til orientering.

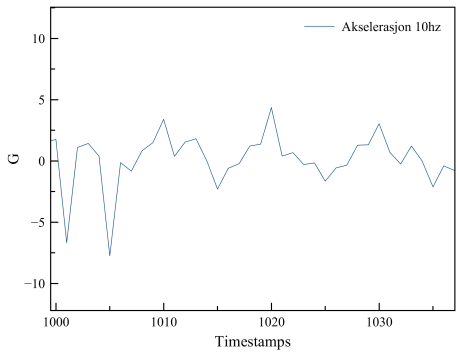
Det er også kanskje oppdaget i figur 5 at det er en side som kalles for step detection, dette snakket jeg litt om tidligere ved å si at sensorene hadde innebygd skritteller som jeg kanskje kunne få bruk for. Etter å ha testet denne skrittelleren, kom jeg til konklusjonen at den ikke fungerte så veldig bra til det den var laget til å gjøre, og valgte å avise tanken om at denne kanskje kunne komme til nytte. Jeg har derfor valgt å ikke dekke mer av denne funksjonen i rapporten.

3.4 Analyse av data

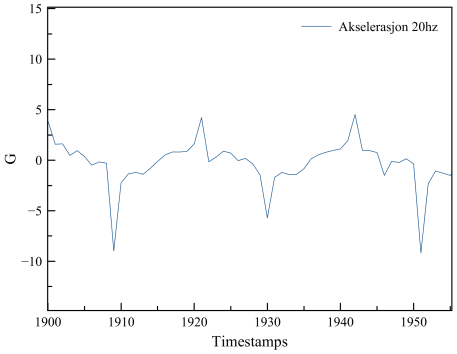
I figur 7, ser vi målingene i de forskjellige frekvensene over 2 sekunder.

Som vi ser i figur 7, hvor vi kan bruke 100hz målingene som et utgangspunkt, gir 10hz målingene en veldig uklar graf i forhold til de andre, det kan være umulig å si hva som skjer, da det ikke er noe spesielt mønster som gjentar seg. Først i 20hz målingene, begynner vi se et mønster, til gjentakelse av samme bevegelse, men det er mulig mye data

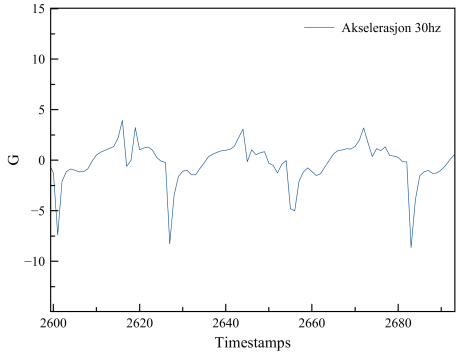
[2] https://github.com/mbientlab/MetaWear-SDK-Android



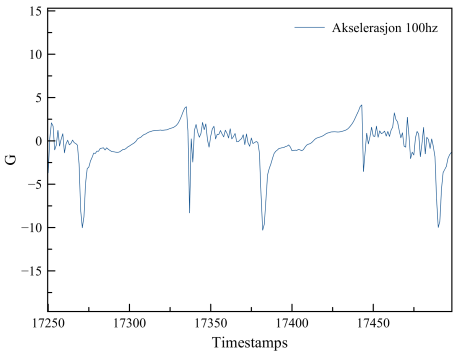
(a) Målinger over 2 sekunder i 10hz.



(b) Målinger over 2 sekunder i 20hz.



(c) Målinger over 2 sekunder i 30hz.



(d) Målinger over 2 sekunder i 100hz.

Figur 7: Test målinger i ulike frekvenser.

kan ha gått tapt, siden det ikke er nok målinger per sekund. I 30hz målingene begynner vi å se antydning til støy i grafen, dette er naturlig, siden staven aldri vil være i helt i ro under skiøkten. Ved 100hz målingene, ser vi klart å tydelig all akselerasjon som skjer med staven.

Jeg har valgt å gå med 100hz målinger, fordi jeg har veldig begrenset tilgang til å kunne anskaffe test data, samt teste ut forskjellige frekvenser ytterligere. Ved 100hz får jeg det beste sensorene klarer å gi, men det hadde vært gunstig å senke frekvensen, for å redusere filstørrelser som diskuteres senere i rapporten.

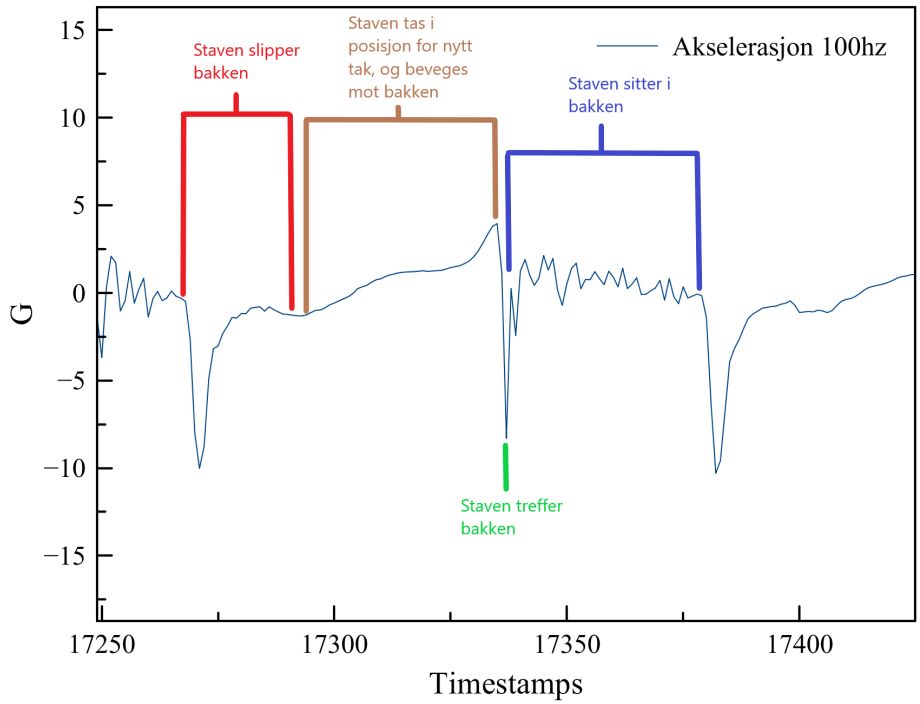
Før vi begynner å analysere test data, må vi tenke oss til hva vi faktisk leter etter for å kunne gjenkjenne de følgende kritiske bevegelsene vi er ute etter:

- Når skistaven settes i bakken.
- Når skistaven slipper bakken.

For å kunne finne disse punktene, er vi nødt til å vite hvordan sensoren fungerer, og hvordan den er festet til skistaven. Orienteringen på sensorene er bestemt til å være på yttersiden av skistavene, med knappen på sensorene mot toppen. Figur 6 som er vist i kapittelet om sensorenes virkemåte, viser orienteringen sensorene er festet på skistaven, samt hvilke type utslag vi kan forvente, da sensoren beveger seg i gitt retning. Vi vet at da skistaven slipper bakken, kan vi forvente et hopp i negativ retning, siden den vil da få

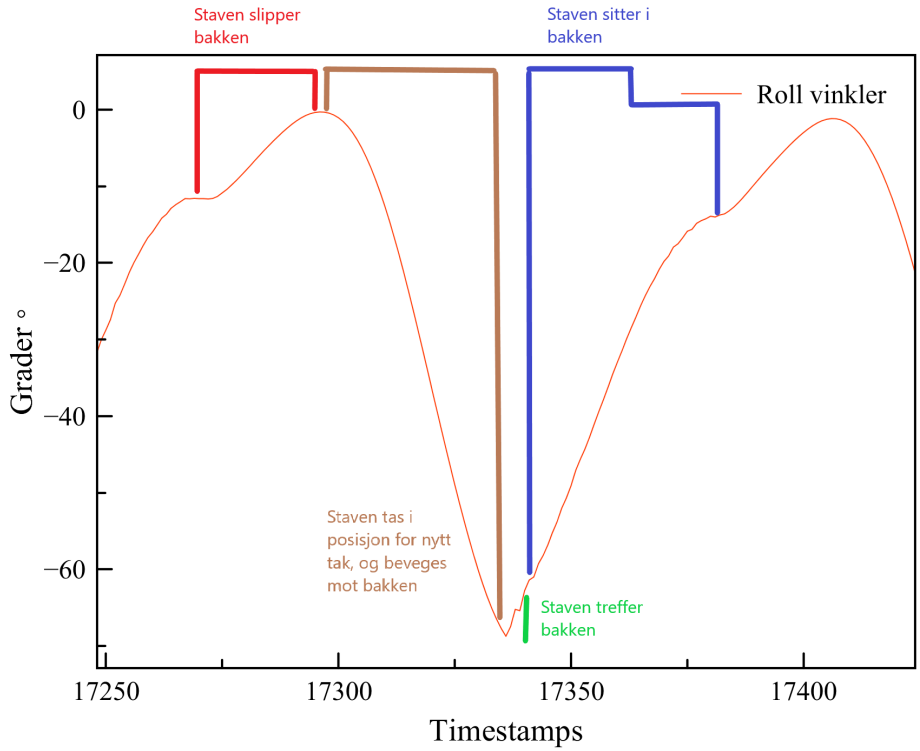
akselerasjon oppover, vi kan da videre forvente at akselerasjonen vil få en stopp i retning oppover, for da er skiløperen klar til å sette skistaven i bakken igjen.

Dermed vil det forekomme en akselerasjon som gir positivt utslag, da skiløperen beveger skistaven mot bakken igjen forså at den settes i bakken, da kan vi forvente en brå stopp og ifølge Newtons tredje lov, vil vi få en like stor motsatt akselerasjon, dette fungerer ikke helt slik i virkeligheten, da det er snakk om andre faktorer som spiller inn på kreftene, som for eksempel underlaget, som vil være snø. I figur 8, har jeg blåst opp et segment av mønsteret som gjentar seg under stakingen, og delt denne opp i sektorer som viser de forskjellige tegnene jeg har snakket om her.



Figur 8: Oppdelt akselerasjon fra skiøkt.

Måten vi kan bekrefte at jeg har funnet riktig punkter for figur 8, er ved at vi ser på vinklene for samme tidsrom i skiøkten. I figur 9 har jeg delt opp likt som i figur 8, men for vinklene.



Figur 9: Oppdelte vinkler fra skiøkt.

Vi må tenke oss tilbake til figur 6 fra kapittelet om sensorenes virkemåte. Vi vet at når skistaven er horisontal, skal det gi et utslag på 0°, og da staven er vertikal skal det gi et utslag på −90°.

På figur 9, kan vi se at det er en liten ujevnhet der staven slipper bakken, forså at den fortsetter mot 0°, dette forventes, siden staven vil bevege seg mot horisontal stilling etter den har sluppet bakken. Vi kan også se når skiløperen begynner å bevege staven mot et nytt staketak, da vinkelen begynner å bygge seg opp mot negativ retning, og at staven settes i bakken på rundt −69°, her ser vi igjen en ujevnhet i vinkelene. Dermed vil staven sitte i bakken, og vendes mot en horisontal stilling igjen før den slipper bakken igjen.

4 Utviklingsprosess

4.1 Planlegging av prosess

Jeg satte som mål å ta for meg 80 timer med oppgaver per sprint. Arbeidstiden skulle være fra 08:00 til 16:00 hver ukedag, altså 40 timer per uke, som skulle gå opp med arbeidsmengden per sprint. For å måle fremgangen i prosjektet, benyttet jeg meg av Trello, hvor jeg kan holde styr på alle oppgaver. Jeg har også laget noen generelle rutiner for utviklingen:

- Det skal ikke pushes opp kode som ikke kompilerer til repositoriet.
- Alt som pushes til repositoriet, skal ha en passende kort melding om hva som er blitt gjort.
- Det skal skrives logg hver arbeidsdag om hva som er blitt gjort.

Disse runtinene er for at repositoriet alltid skal ha fungerende kode, samt at hvis det skulle være en feil, har jeg dokumentert hvor feilen oppstod ved hjelp av at det er en melding som indikerer hva som skjedde for gitt push til repositoriet. Loggen som skrives hver dag, er for at jeg skal kunne se tilbake på når jeg har gjort hva, samt når jeg stod ovenfor valg. Loggen vil komme til nytte nå som jeg skriver denne oppgaven.

Det kan også nevnes at møter med veileder blir holdt hver uke, for å diskutere oppgaven og for å få tilbakemeldinger på fremgangen av prosjektet. Møter med oppdragsgiver er ikke planlagt, og holdes etter som de er nødvendige, samt at det passer for begge parter.

4.2 Prioriteringer av oppgaver

Prioriteringen av oppgaver bestemmes ut fra hva jeg og oppdragsgiver synes er mest matnyttig å få ut av prosjektet. Jeg for min del har lyst å fokusere på å lære meg Android utvikling, så for meg er oppgaver knyttet til dette viktig. Oppdragsgiver derimot synes det er mest matnyttig med en algoritme for gjenkjenning av kritiske punkter under et stavtak. Oppgavene i prosjektet kan samles under tre kategorier:

1. Lage og implementere et brukergrensesnitt.
2. Implementere metoder for generering av data fra sensorer.
3. Lage og implementere metode for gjenkjenning av stavtak.

Dette er også rekkefølgen utviklingen skal følge. Grunnen til at brukergrensesnittet kommer først, er fordi dette skaper en struktur i prosjektet, slik at jeg kan implementere og teste de senere funksjonene som har med sensorer og algoritmen å gjøre. Alt som har med sensorene å gjøre, kommer da etter brukergrensesnittet. Jeg bestemte meg for at det er mer gunstig at dette kommer før algoritmen, slik at jeg kan samle inn data for å teste algoritmen da denne skal utvikles.

5 Design

5.1 Optimalisering av kode

Det er viktig å ha optimalisert kode. Dette på grunn av oppnåelse av bedre ytelse, slik at det blir en bedre brukeropplevelse, samt for å opprettholde ryddig kode. Jeg har tatt et steg mot å nå disse målene, ved å benytte meg av Android sine egne tips[1] om optimalisering av kode. Jeg har også prøvd å være konsekvent ved oppsett av klassene. klassenes oppsett er som følger:

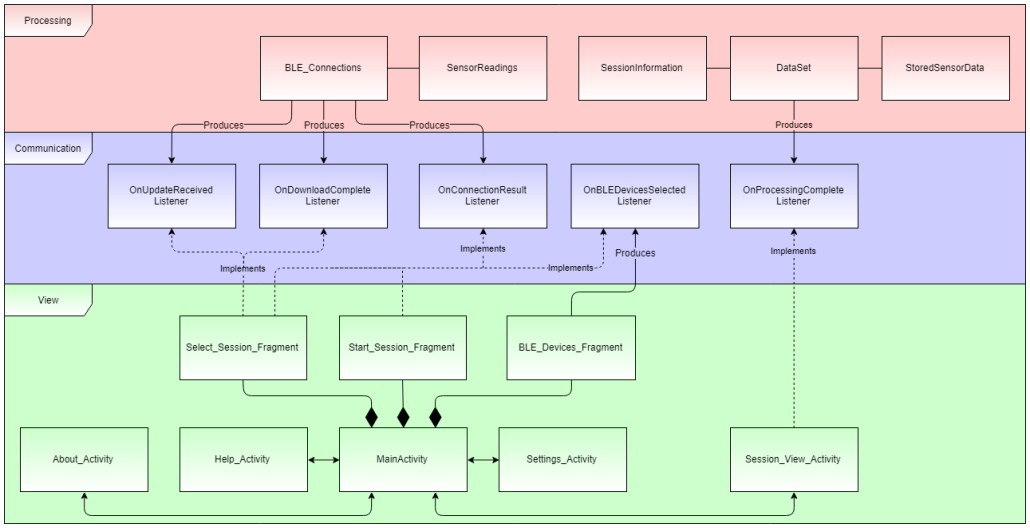
1. Importering av biblioteker.
2. Deklarering av globale statiske variable.
3. Deklarering av globale private variable.
4. Deklarering av globale offentlig variable.
5. Deklarering av interfaces.
6. Klasse konstruktor.
7. Offentlige funksjoner.
8. Private funksjoner.
9. Inner-klasser.

Grunnen til at jeg har valgt å ha et konsekvent oppsett av klassene, er fordi dette danner mer ryddig kode, samt gir det bedre oversikt over koden.

5.2 Struktur på høynivå

Stukturen i Appitude kan ses på figur 10. Dette er applikasjonen sin struktur på et veldig høyt nivå, som ikke inneholder relasjoner til tredjeparts klasser og biblioteker. Vi kan se at jeg har delt opp strukturen i tre lag. View-laget er ansvarlig for å stå for grafikk og interaksjon med bruker. Processing-laget står for tyngre oppgaver bak kulissene, som brukergrensesnittet ikke har noe med å gjøre. Til slutt er Communication-laget, som utfører kommunikasjon mellom Processing-laget, og View-laget.

[1] https://developer.android.com/training/articles/perf-tips



Figur 10: Simplifisert høynivå struktur og assosiasjoner i applikasjonen.

Vi kan se at Communication-laget består av «Listenere», disse har minst én «Producer» og minst én «Listener», poenget her er at en producer kan produsere en melding da noe spesielt skjer, og alle som er satt til å høre på, altså listenere, får da denne meldingen og kan handle ut det. Meldingen som sendes kan inneholde alt fra ingenting, altså bare at varsel om at noe har skjedd, eller data, slik at man kan handle ut fra data. Figur 10 er bare en simpel versjon for å vise assosiasjoner mellom noen veldig sentrale klasser i applikasjonen. Den fulle versjonen av klassediagrammet kan ses i Appendiks C

5.3 Gjenbruk av kode

Når det kommer til gjenbruk av kode, har jeg benyttet meg av to eksterne biblioteker. Dette er de samme to bibliotekene jeg benyttet meg av i tilleggs applikasjonen, som ble nevnt tidligere i rapporten.

Det første biblioteket er Mbientlab sitt API[1] for bruk av sensorene. Dette biblioteket lar meg sende forespørsler, og endre innstillinger på sensorene. Den eneste måten jeg kunne ha kommet meg rundt å bruke dette API'et, var å lage egen firmware til sensorene, dette fordi koden til firmwaren holdes bak lukkede dører, samt måtte jeg h laget et API til å operere med firmwaret. Dette ville vært et prosjekt i seg selv, og jeg fant ikke dette hensiktsmessig å gjøre, siden det er noe som allerede finnes, og selv om jeg synes at APIet kan være litt tungvindt å bruke, så kan det utføre det jeg trenger at det skal kunne utføre. Dette biblioteket er som sagt utgitt med lisensieringen Mbient inc, som er en variant av Copyright, men som vi kan lese i deres lisensiering, gir de rettighetene til å modifisere, kopiere, og distribuere, gitt at det brukes i sammenheng med et av deres produkter. Derfor, med lisensieringen som brukes her, er ikke oppdragsgiver begrenset på noen måte ved bruk av dette biblioteket.

Når det kommer til det andre biblioteket, stod jeg ovenfor et valg, det finnes mange biblioteker for å plotte grafer i Java, her er de bibliotekene som var til vurdering:

- AChartEngine

[1] https://mbientlab.com/documents/metawear/android/latest/

- Androidplot
- GraphView
- MPAndroidChart

AChartEngine hadde ikke blitt oppdatert på over et år fra da jeg så etter biblioteker, og har liggende feil som må rettes ifølge tilbakemeldinger på biblioteket. Dette gjør at jeg ikke har lyst å bruke dette biblioteket, fordi hvis jeg skulle møte på feil nå, eller i fremtiden, er det store sjanser for at man ender opp med å fikse disse selv.

AndroidPlot er stabilt, tilbyr god funksjonalitet, og har fått gode tilbakemeldinger. Biblioteket skal være enkelt å bruke, og har god brukerstøtte, både fra andre utviklere som har brukt biblioteket, samt utviklerne selv. Dette biblioteket blir oppdatert til den dag i dag, og fremstår som en veldig god kandidat til mitt prosjekt.

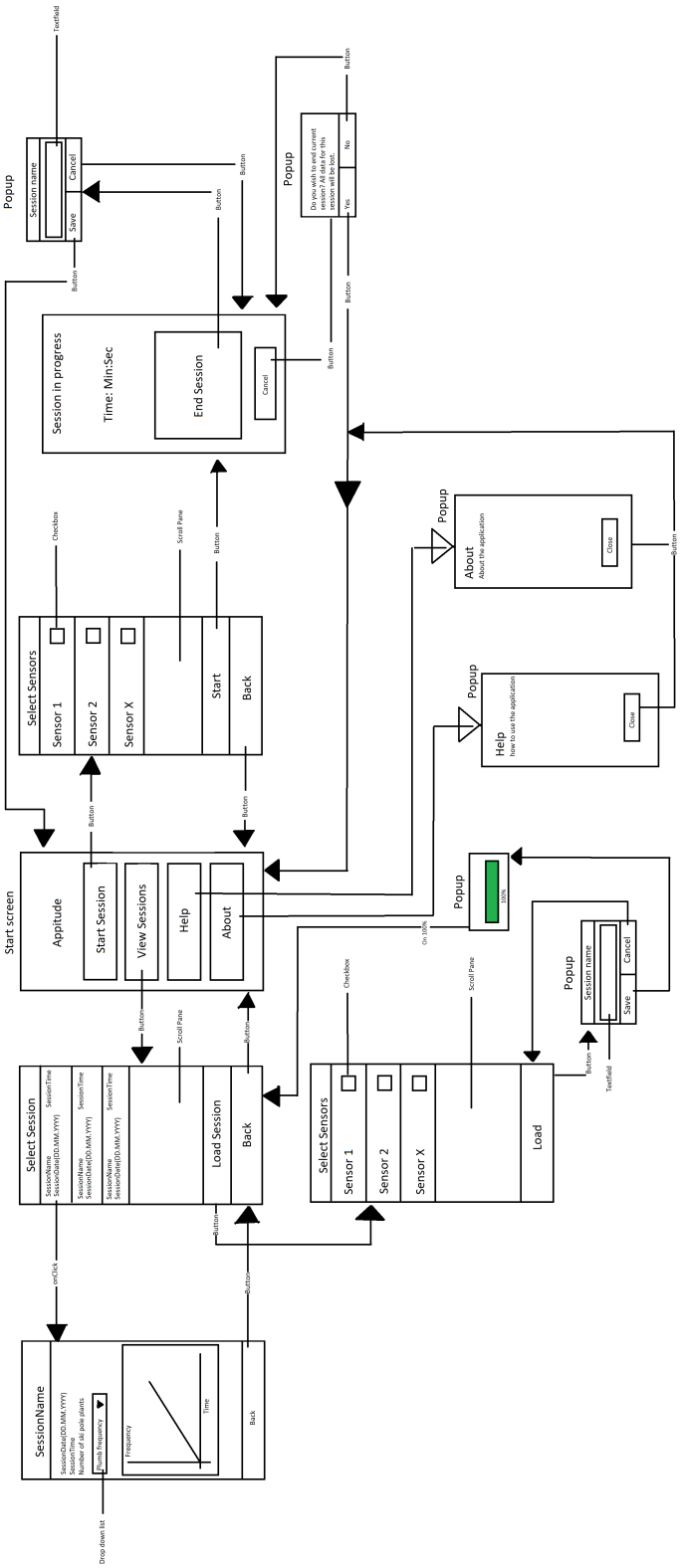
GraphView er en kandidat jeg selv prøvde ut, dette biblioteket var ekstremt lett å håndtere, veldig stabilt og man kan endre utseende på grafene i en middels høy grad. Dette biblioteket kom med en ulempe da jeg prøvde det, og det er at det er begrenset på hvor mye datapunkter man kan ha i grafene. Da jeg prøvde det ut lå dette på 16000 punkter, og de lovet ingenting når det kom til over dette antallet. Av den grunn har jeg valgt å ikke bruke dette biblioteket, selv om det er tvilsomt at det noen gang behøves over 16000 punkter, vil jeg helst ikke begrense brukeren, eller skape problemer for fremtidig utvikling av prosjektet, da det finnes andre og bedre valg som ikke har dette problemet.

MPAndroidChart er desidert det mest brukte biblioteket av de jeg har nevnt her. Biblioteket tilbyr alt man kan ønske seg, og eneste begrensende faktor oppgitt fra utvikler, er at det ikke offisielt støtter sanntids grafer, men som man kan få til hvis man ønsker det. Jeg brukte dette biblioteket i min tilleggs applikasjon og fikk til å lage sanntids grafer. Biblioteket blir oppdatert til den dag i dag, og tilbyr tilpasning av grafene i høyeste grad. Jeg har valgt å bruke dette biblioteket fremfor de andre grunnet at det tilbyr all funksjonalt jeg ønsker meg, og mer til. Fordi det er så populært, dette gjør at hjelp og dokumentasjon er mer tilgjengelig hvis jeg skulle trenge det.

MPAndroidChart er som sagt lisensiert under Apache V2 lisensen [4], som er en BSD-lignende lisensiering. Dette vil si at applikasjonen ikke holdes tilbake på noen måte på grunn av lisensen. Eneste kravet som stilles til applikasjonen, er at den er nødt til å inkludere en kopi av lisensen. I min forståelse, er det nok at biblioteket som blir brukt holder denne kopien, men det er pent gjort å gi utvikleren av biblioteket noen fine ord et sted i applikasjonen. Det skulle også være et krav å ha med «NOTICE» tekst-filen hvis biblioteket hadde det, men i dette tilfellet har ikke biblioteket noen slike filer.

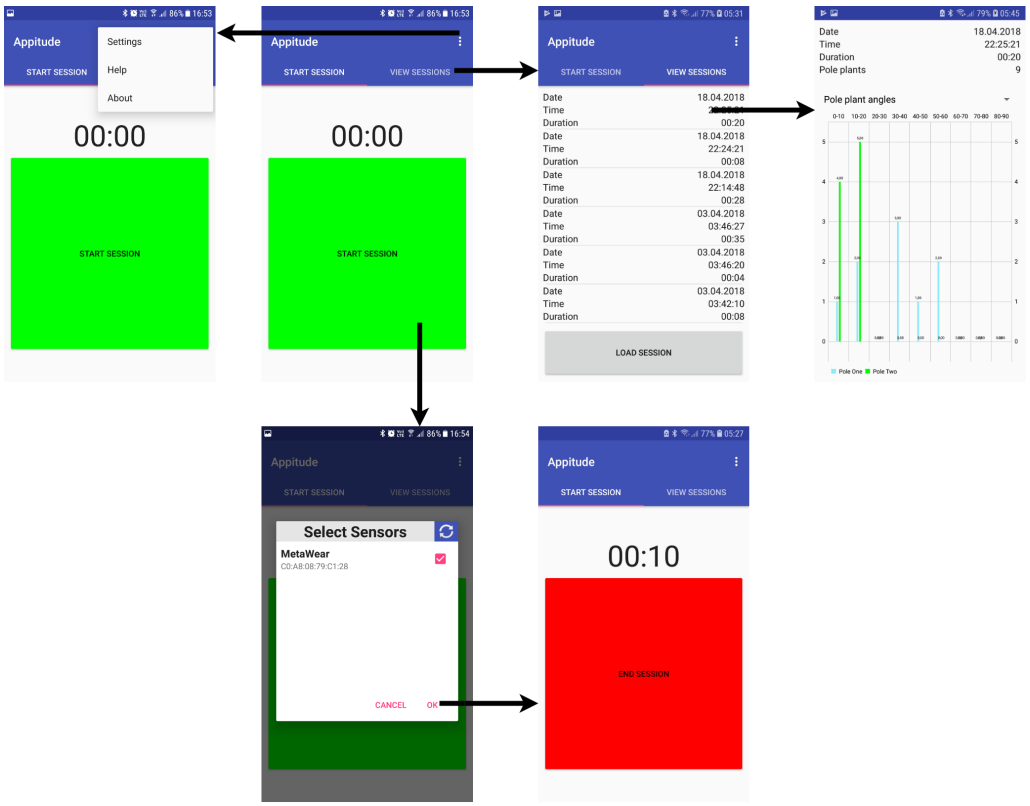
5.4 Brukergrensesnitt

Brukergrensesnittet er det brukeren av applikasjonen kan se, og står for interaksjon mellom bruker og applikasjon. Jeg har ikke hatt brukergrensesnitt i så stort fokus, siden dette ikke er helt mitt fagfelt, og fordi det ikke er essensen i prosjektet. Derfor har jeg i mitt brukergrensesnitt, brukt standard Java elementer for brukergrensesnitt. I figur 11 kan man se en tidlig planlegging av brukergrensesnittet.



Figur 11: Tidlig versjon av brukergrensesnitt

Jeg sendte versjonen av brukergrensesnittet som er vist på figur 11 til oppdragsgiver, for å få tilbakemeldinger på hvordan oppdragsgiver ønsket at det skulle se ut. I tilbakemelding opplyste oppdragsgiver meg om at knapper for å gå tilbake ikke var nødvendig, siden det er snakk om android er det tilbakeknapper på telefonen som kan benyttes. Jeg fikk også vite at knapper for «hjelp» og «om» burde bli satt i en appbar meny, dette er en liten horisontal i toppen av brukergrensesnittet, som kan inneholde både tekst, knapper og menyer. Videre fant jeg også ut at at jeg kunne omplassere og endre på ting, for å skape en bedre brukeropplevelse. Det endelige brukergrensesnittet, kan ses på figur 12.



Figur 12: Endelig versjon av brukergrensesnitt

Vi kan klart se på figur 12 at det endelige brukergrensesnittet har trekk fra den tidlige versjonen. Jeg fant det mer hensiktsmessig, å bytte ut elementene på skjermen da det startes en økt. Måten jeg har gjort dette på er at når det trykkes på «Ok» og man får koblet opp mot sensorene, så endres fargen på knappen, samt endres teksten. Det er også satt inn en tidtaker, denne er alltid en del av brukergrensesnittet, men gjøres usynlig da den ikke er i bruk. Som vi også ser, har jeg tatt tilbakemeldingene jeg fikk fra oppdragsgiver på alvor, jeg har fjernet alle tilbakeknapper, samt satt inn en appbar med «dropdown»-meny, som deretter gir tilgang til «Innstillinger», «Om», og «Hjelp», disse leder foreløpig bare til tomme sider, siden det ikke finnes noen hensiktsmessige innstillinger enda, og jeg synes «Om» og «Hjelp» burde skrives av oppdragsgiver, siden det er hun som holdes ansvarlig for applikasjonen når prosjektet er avsluttet. Man kan også se at jeg har endret layout på hjem skjermen, man kan nå «swipe» for å komme seg til «View Sessions», dette er mer ryddig og gir et generelt bedre inntrykk for brukeren.

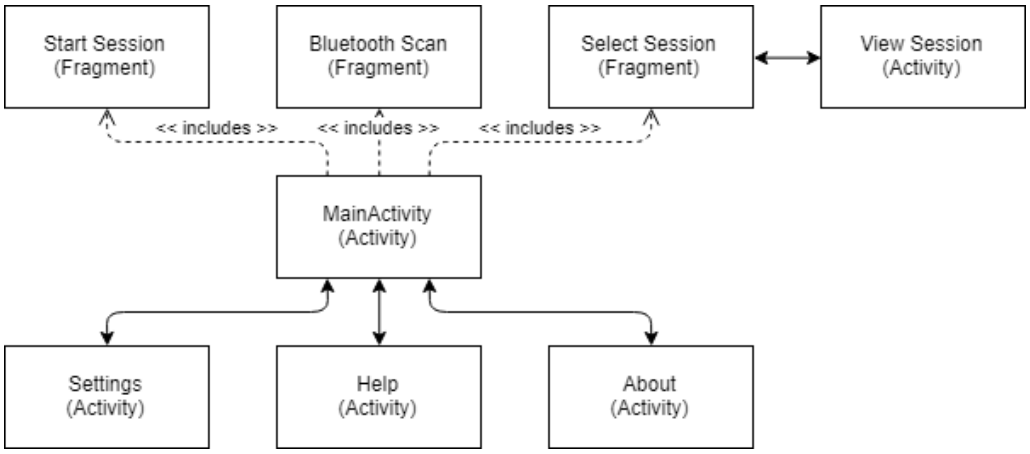
Søking av sensorer er nå et popup-vindu, som igjen er mer ryddig, og tar mindre ressurser enn å bytte ut innholdet på hele skjermen. Det er også et element som mangler her som jeg dessverre ikke får inkludert bilde av, siden jeg ikke lenger har tilgang til sensorene. Det som mangler bilde av er at når man trykker på «Load Session»-knappen, får man samme popup-vindu som vist da man skal starte en økt, og knappen byttes ut med en progresjonsbar, som viser progresjonen i nedlastningen av logget data.

5.5 Activity og Fragment

Når man skal lage brukergrensesnitt for Android, står man ovenfor et valg, dette valget gjelder mellom noe som kalles «Activity» og «Fragment» klasser. Disse klassene kan ses på som bokser, som inneholder kode for alt som foregår med brukergrensesnittet. Et eksempel hva disse klassene gjør, er at de inneholder kode for hva som skal skje når siden åpnes og lukkes.

Activity og Fragment klassene, kan ses på som veldig like ved første glimt, men det er noen vesentlige forskjeller. Activity klassen er hoved boksen, hvor man kan sette inn Fragment objekter, altså er det Activity objektet som ligger i grunnen. Man kan lage brukergrensesnittet helt basert på Activity objekter, dette er en feil jeg gjorde da jeg var ny i dette prosjektet. Jeg fant senere ut at å bytte mellom Activity objekter, spiser vesentlig mer ressurser enn å bytte mellom Fragment objekter. Dette gjorde at jeg endret mye av koden, og prøvde minimalisere bruken av Activity objekter, og heller benytte meg av Fragments.

Ombyttet til Fragments gjorde det mulig å for eksempel lage swiping mellom skjermer, som man ser at jeg har gjort i figur 12. Det kan være vanskelig å se hvilket objekt man skal benytte seg av når det kommer til Activity og Fragment klassene, dette er nok en av de store utfordringene med å kunne lage et godt optimalisert brukergrensesnitt til Android. Jeg har valgt objekt basert på hvor mye siden skal gjøre, og om det har noen sammenheng med noen andre sider. I figur 13 kan vi se det endelige strukturen av Activity og Fragment objekter i applikasjonen.



Figur 13: Oppsett av Fragments og Activity

6 Implementasjon

6.1 Implementasjon av brukergrensesnitt

Bak kulissene av brukergrensesnittet, gjemmer det seg kode som skaper grafikken, og gir elementene en funksjon, det hadde vært kjedelig om ingenting skjedde da man trykket på en knapp. Grafikken skapes ved bruk av XML-kode, hvor jeg har valgt å bruke standard Java brukergrensesnitt elementer. I grunnen av XML-koden, ligger en layout type som man bygger opp med elementer. I kode eksempel 6.1 kan vi se hvordan en slik layout bygger seg opp i form av XML-kode.

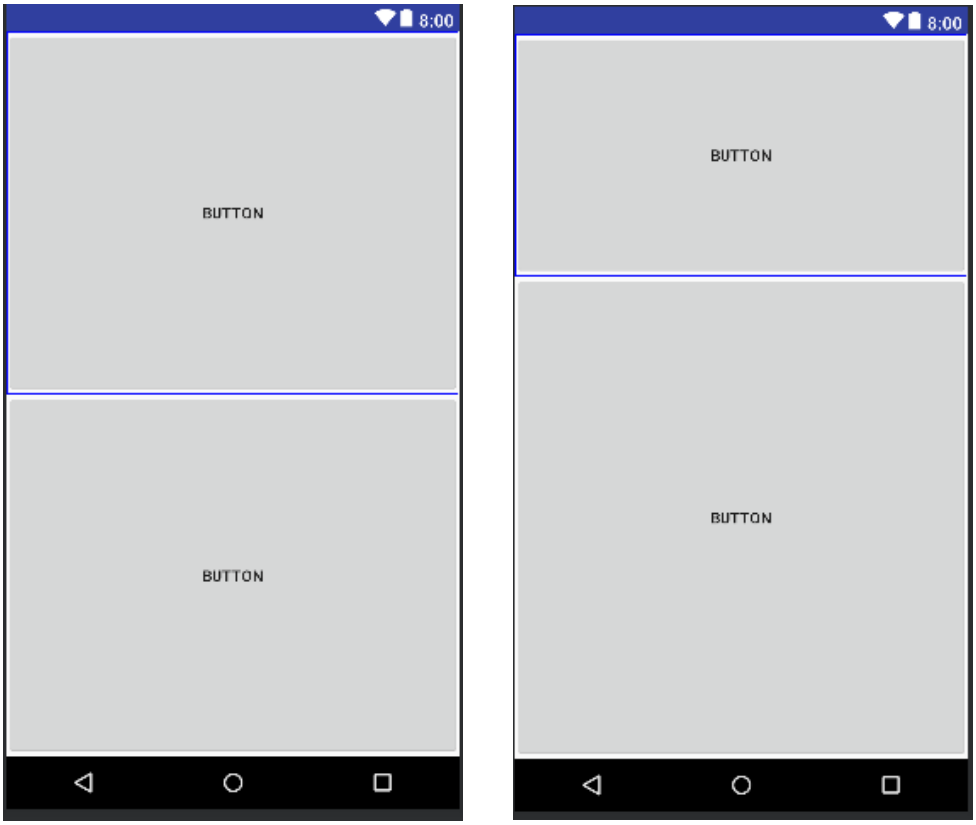
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/sessionLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true"
    android:focusable="true"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical">
```

Kode eksempel 6.1: Bruk av XML til bygging av grafisk grensenitt

Dette er starten av XML-koden til siden der man kan se på en økt. Som vi ser har jeg brukt «LinearLayout» i grunnen, som jeg bygger opp. Her kan vi se interessante verdier man kan stille inn. «layout width» og «layout height» er to veldig viktige verdier, disse sier oss hvordan siden skal bruke skjermen, for eksempel har jeg «match parent» på layout width, dette vil si at den bruker samme størrelse i vidden, som layoutet som ligger under denne igjen. Da match parent er brukt på det underste elementet i siden, tar den størrelsen fra Android operativsystemet og benytter seg av hele skjermen. Det er viktig å ikke bruke statiske verdier da man bestemmer vidden og høyden på et layout, dette er fordi applikasjonen skal brukes på flere typer smarttelefoner, med forskjellig skjermoppløsning, så for å bevare skjermforholdet til gitt telefon, må man bruke størrelser hentet fra operativsystemet, og heller bruke «vekter» for å bestemme størrelsen på forskjellige elementer i brukergrensesnittet. I figur 14 kan man se et eksempel på hvordan vekter fungerer.

I figur 14a, er det samme vekt på begge knappene, dette gjør at de deler plassen i skjermen likt, siden dette er de to eneste elementene i skjermen. I figur 14b, er det ulik fordeling av vekter, her har øverste knappen vekten 1, og nederste knappen vekten 0.5. I sammenligning med figur 14a, gjør dette at den nederste knappen, tar opp halve plassen



(a) Lik vekt på knapper, begge vekt 1.

(b) Ulik vekter, øverste knapp 1, nederste 0.5.

Figur 14: Eksempel for bruk av vekter

til den øverste knappen. Jeg har valgt å sette maksimums vekt på et element til 1, og minimum til 0. Dette er i stor grad noe man kan velge rekkevidden på selv, men som jeg forstår er rekkevidden jeg har valgt, en veldig vanlig rekkevidde å bruke.

Så langt, har jeg forklart hvordan man lager utseende på brukergrensesnittet, men ingenting av det jeg har nevnt så langt forklarer hvordan et element, som for eksempel en knapp, får en funksjon og faktisk gjør noe da den trykkes på. Dette er noe jeg gjør i Java kode. Jeg kunne ha referert til en funksjon som skal kjøres i spesifikke situasjoner, som da en knapp trykkes på, i selve XML-koden, men her har jeg vært i kontakt med andre utviklere, og fått til respons at dette ikke er noe utviklere ser positivt på, siden det skaper rotete og ustrukturert kode. I kode eksempel 6.2 kan man se hvordan jeg har løst det funksjonelle bak «Start Session»-knappen fra figur 12. Jeg gjør oppmerksom på at kommentering i kode eksemplene har blitt redusert for å skape mer leselighet, .PDF er ikke optimalt når det kommer til å lese kode.

```
1 private Button startSessionButton; // Declaration of button
2
3       /*Find button in the view by it's ID, then set its color to be green and
      apply correct On click listener*/
4       startSessionButton = view.findViewById(R.id.start_session_Button);
5       startSessionButton.getBackground().setColorFilter(Color.GREEN,
```

```
    PorterDuff.Mode.MULTIPLY);
        startSessionButton.setOnClickListener(this);

public void onClick(View view) {
    switch (view.getId()){
        case R.id.start_session_Button:
            if(activeSession){
                    sessionEnd();
               }
            else{
                FragmentManager fragmentManager = getActivity().
    getSupportFragmentManager();
                BLE_Devices_Fragment newFragment = new BLE_Devices_Fragment();
                newFragment.setOnBLEDevicesSelectedListener(this);
                newFragment.show(fragmentManager, "dialog");
                if(stopClock.getVisibility() == View.VISIBLE)
                    stopClock.setVisibility(View.GONE);
                }
            break;
    }
}
```

Kode eksempel 6.2: Kode som gir «Start Session»-knappen en funksjon.

Først av alt i kode eksempel 6.2 kan vi se at jeg deklarerer en knapp, jeg finner dermed knappen fra XML-filen og initialiserer knappen jeg deklarerte, til å være denne. Vi kan også se at jeg definerer en «OnClickListener» for knappen, dette er da funksjonen som kjøres hver gang knappen trykkes på. I OnclickListeneren, må jeg først finne ut hvilke knapp som trykkes på, dette gjør jeg ved at jeg bruker en switch på elementet sin ID. Det er unødvendig i dette tilfellet, siden det bare er en knapp i denne delen av brukergrensesnittet, men skaper i min mening mer tydelig kode, samt muligheten for å lett legge på flere elementer som bruker denne funksjonen i fremtiden.

6.2 Søking etter bluetooth enheter

Søking etter sensorene foregår i BLE_Devices_Fragment-klassen. Denne klassen søker etter sensorene ved bruk av bluetooth, og legger dem til i en liste, som deretter legges inn i brukergrensesnittet slik at brukeren kan velge sensorene som skal brukes. Som sagt tidligere, skal jeg benytte meg av to metoder for å utføre et bluetooth søk. Jeg må bruke to forskjellige, fordi Android 4.4 bruker en gammel utdatert metode, mens Android 5.0 og senere versjoner har tilgang til en bedre metode for å gjøre dette. Måten jeg velger mellom hvilke metode som brukes, kan ses i kode eksempel 6.3.

```
private void startLeScan(final boolean enable){
    if(Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP){
        baleScanDeprecated(enable);
    }
    else{
```

```
6           bleScan(enable);
7       }
8   }
```

Kode eksempel 6.3: Eksempel på kode valg basert på Android versjon

Prosessene for å gjøre søke opp bluetooth-enheter er veldig like når det kommer til de to metodene, men det benyttes forskjellige bibloteker. Det første jeg gjør da jeg skal søke opp enheter, er å sjekke at applikasjonen har fått rettigheten til å utføre et bluetooth søk (For Android versjon 6.0 og senere kreves det også rettigheter til lokasjon), hvis telefonen tidligere ikke har gitt rettighetene til dette, spør applikasjonen om rettighetene.

Da rettighetene for å utføre et søk er i orden, lages det et «ScanCallBack», dette er for når bluetooth-enheter blir funnet. Da en bluetooth-enhet blir funnet, kalles det på «ScanCallBack» som videre lagrer informasjonen om enheten i minnet, samt legger enheten til i listen over enheter man kan velge mellom i brukergrensesnittet.

Videre lages det en oppgave som skal kjøres senere, denne oppgaven sin funksjon, er å stoppe søket etter bluetooth-enheter etter en periode har gått. Dette er fordi det å søke opp bluetooth-enheter bruker veldig mye strøm, så man vil ikke at telefonen skal konstant gjøre dette. Jeg har satt min periode til å være på 10 sekunder, samt har jeg lagt inn muligheten til å stoppe et pågående bluetooth søk, for hvis brukeren skulle gå tilbake, eller at enhetene brukeren leter etter alt er funnet.

Når alle forberedelser til å utføre et søk er uført, kan jeg endelig starte selve søket. Ved å utføre et bluetooth søk, vil man få opp alle enheter som bruker bluetooth, dette er noe jeg ikke ønsker. Derfor har jeg spesifisert at det er MbientLab sine sensorer jeg leter etter ved bruk av UUID (Universally unique identifier), som er unik til disse type sensorer.

6.3 Tilkobling og frakobling av sensorer

Etter at man har funnet sensorenes informasjon med hjelp av bluetooh søket, er det tid for å koble seg opp mot sensorene. En liste med sensorer som det ønskes å opprette en tilkobling med, sendes med som en parameter til et nytt BLE_Conntections objekt, dette er klasses som er ansvarlig for alle bluetooth tilkoblinger, samt sending og mottak av data over bluetooh. BLE_Connections implementerer «ServiceConnection» som er et interface for å kunne overvåke en ekstern tilkobling, og binder denne opp med en klasse fra API'et til Mbientlab som er laget for å kunne kommunisere med sensorene.

Etter bindingen er ferdig, vil applikasjonen initialisere alle sensorene, med dette mener jeg at jeg henter all nødvendig informasjon om selve sensoren over bluetooth, for så å lage et «MetaWearBoard» objekt, dette er en representasjon av sensoren i Java kode. Dermed, da all initiasjon og informasjon er på plass, vil applikasjonen prøve koble seg opp mot alle sensorene. Kode eksempel 6.4 viser hvordan jeg har løst selve oppkoblingen av sensorer.

```
1 public boolean connectToBoards() {
2     ExecutorService executor = Executors.newCachedThreadPool();
3     for(MetaWearBoard board : boards){
4         executor.execute(()->
5             connectAsync(board)
```

```
6           );
7       }
8       try {
9           executor.shutdown();
10          executor.awaitTermination(CONNECTION_TIMEOUT, TimeUnit.SECONDS);
11      } catch (InterruptedException e) {
12          e.printStackTrace();
13      }
14      for (MetaWearBoard board : boards){
15          if(!board.isConnected()){
16              return false;
17          }
18      }
19      return true;
20  }
21
22  public void connectAsync(MetaWearBoard board){
23      try {
24          board.connectAsync().continueWith((task)-> {
25                  if (task.isFaulted()) {
26                      connectAsync(board);
27                  }
28                  return null;
29          }).waitForCompletion();
30      } catch (InterruptedException e) {
31          e.printStackTrace();
32      }
33  }
```

Kode eksempel 6.4: Oppkobling av sensorer

Som vi kan se i kode eksempel 6.4 starter jeg det hele av med å lage en tråd for hver av sensorene som skal kobles opp, dette gjør jeg for å skape fortgang i oppkoblingen, ved at flere sensorer kan kobles opp samtidig. Hoved tråden som fødte trådene som tar seg av oppkoblingene, vil dermed skru av konteineren av tråder, slik at ingen nye tråder kan startes i denne konteineren, forså at den tvinger avslutning av alle gjenværende tråder etter en periode har gått. Jeg har valgt å gi trådene 30 sekunder på å fullføre tilkoblingen, før de termineres. Hoved tråden vil vente her til enten de 30 sekundene har gått, eller til trådene som tar seg av oppkoblingene blir ferdige.

Etter at trådene som skal foreta oppkoblingene er laget, kjører de «connectAsync» metoden for sensoren den gitte tråden har ansvar for å koble seg til. Denne metoden vil prøve koble seg til brettet, og hvis tilkoblingen feiler, vil den fortsette å prøve på nytt til den lykkes. Det er her grunnlaget for terminering som jeg nevnte kommer inn, dette gir tråden muligheten for flere forsøk på å koble seg til, før den til slutt termineres fordi det har gått for lang tid. Lykkes derimot tråden med å koble til sensoren, er den ferdig for dagen og avslutter seg selv.

Da alle trådene er ferdig, eller 30 sekunder har gått, vil hoved tråden gå videre. Hoved

tråden vil da sjekke om alle sensorene ble tilkoblet eller ikke. Hvis en sensor har feilet oppkoblingen sin, anses oppkoblingen som feilet, og funksjonen vil returnere «false» for å melde ifra til kallende funksjon om at oppkoblingen feilet. Hvis derimot oppkoblingen var en suksess, returnerer funksjonen «true».

Det er også viktig å koble seg fra sensorene på riktig måte etter man er ferdig med å bruke dem. Dette fordi hvis man ikke kobler seg fra på riktig måte, vil sensoren fremdeles tro at den er koblet opp mot en telefon, og man vil ikke lenger kunne finne den ved et bluetooth søk, før den forstår at den ikke lenger har en tilkobling, som kan ta veldig lang tid. Frakoblig av sensorene er en kortvarig prosess, og er tilsvarende lik meldingen som trådene sender til brettene ved tilkobling.

6.4 Strømming av data

Før jeg kan foreta strømming, har sensorene innstillinger som må stilles inn for at de skal fungere på måten jeg ønsker. Jeg må selv velge hvilke funksjoner som skal brukes, og manuelt programmere inn dette. Kode eksempel 6.5 viser hvordan jeg endrer på innstillingene, og starter strømmingen.

```
SensorFusionBosch sfb = board.getModule(SensorFusionBosch.class);
sfb.configure()
    .mode(SensorFusionBosch.Mode.IMU_PLUS)
    .accRange(SensorFusionBosch.AccRange.AR_16G)
    .gyroRange(SensorFusionBosch.GyroRange.GR_2000DPS)
    .commit();
setSettings(board);

sfb.linearAcceleration().addRouteAsync((source)->
    source.stream((data, env)-> {
        readings.addAccelerationData(new AccelerationReading(
        data.value(Acceleration.class).x(),
        data.value(Acceleration.class).y(),
        data.value(Acceleration.class).z(),
        data.timestamp()));
    })
).continueWith((task)-> {
    sfb.linearAcceleration().start();
    sfb.eulerAngles().start();
    sfb.start();

    return null;
});
```

Kode eksempel 6.5: Starting og endring av innstillinger på sensorer

Som vi kan se i kode eksempel 6.5, så henter jeg ut modulen jeg skal bruke fra sensoren, som i dette tilfellet er «SensorFusion» modulen. Dermed så stiller jeg inn modus, og måle rekkevidden på både akselerometer og gyroskop. Med dette gjort, er innstillingene på sensoren i orden, men jeg må fremdeles velge hva som skal skje med dataen

som måles. Dette gjøres ved bruk av «ruter», altså jeg definerer en rute for gitt funksjon, og velger hva som skjer med dataen. I mitt tilfelle velger jeg at dataen skal legges inn i et «SensorReadings»-objekt, dette er klassen som er ansvarlig for å lagre målet data til fil, jeg kommer til å snakke mer om filsystem og lagring senere. I kode eksempel 6.5 kan man se at jeg har laget en rute for akselerasjonen, og tilsvarende gjøres for Euler-vinklene. Til slutt velger jeg alle funksjoner på sensoren som skal brukes, og starter dem. Kode eksempel 6.5 er et eksempel fra strømming av data, og etter man har gjort som jeg har beskrevet ovenfor, vil strømmen av data fra sensorene starte.

For å stoppe strømmingen, gjør man tilnærmet det samme som gjøres ved slutten av kode eksempel 6.5, men istedet for at man bruker kommandoen «start», bruker man kommandoen «stop» på alle funksjonene.

6.5 Logging av data

Når det gjelder logging av data, er dette litt annerledes, men ikke helt fremmed fra strømming. Jeg må fremdeles endre innstillingene på sensorene, og lage ruter. Eneste forskjell her er at rutene blir satt til å logge dataen som måles. Men det er noe vesentlig annerledes med loggingen, her må jeg lage et såkalt «Macro», som er et opptak av kode. Macroet lagres og på sensoren, slik at den ikke trenger en telefon for å kjøre koden som er i macroet. I kode eksempel 6.6 viser jeg hvordan et macro opprettes.

```
1           macro.startRecord(true);
2           onBoardSwitch.state().addRouteAsync((source)->
3
4                   source.count().map(Function2.MODULUS, 2).filter(Comparison.
    EQ, 1).
5                           accumulate()
6                           .map(Function2.MODULUS, 2)
7                           .multicast()
8                           .to()
9                           .filter(Comparison.EQ, 1)
10                          .react((token)-> {
11                                  startLogging(board);
12                                  led.editPattern(Led.Color.GREEN, Led.
    PatternPreset.SOLID)
13                                  .repeatCount((byte) 2)
14                                  .commit();
15                                  led.play();
16                          })
17                          .to()
18                          .filter(Comparison.EQ, 0)
19                          .react((token)-> {
20                                  stopLogging(board);
21                                  led.editPattern(Led.Color.RED, Led.
    PatternPreset.SOLID)
22                                          .repeatCount((byte) 2)
23                                          .commit();
```

```
24                                  led.play();
25                              })
26              );
27              macro.endRecordAsync()
```

Kode eksempel 6.6: Oppretting av macro på sensor

Som vi kan se i kode eksempel 6.6, starter jeg et opptak, alt som skjer i form av kode etter opptaket startes, lagres i macroet, og opptaket stopper som vi ser på siste linjen. I macroet, lager jeg en rute for knappen som er på sensorene, altså jeg gir knappen en funksjon. Mitt ønske her var at når knappen trykkes på engang, så starter loggingen, og da knappen trykkes på for en andre gang, stopper loggingen. For å få til denne funksjonen måtte jeg operere med funksjoner som var tilgjengelig for ruten til knappen, og som man kan se, ender det opp med en veldig avansert måte å skrive en if/else setning.

Hovedsaklig det alle funksjonene gjør, er at sensoren holder lagret hvor mange ganger knappen har blitt trykket på. Dette bruker jeg til min fordel ved at jeg bruker (antall ganger knappen har blitt trykket på) % 2, da vil jeg ut fra svaret enten få 0, eller 1 og ut fra det kan jeg finne ut av om det var første, eller andre gang brukeren trykket på knappen.

Koden som kjøres er avhengig av om det den skal starte eller stoppe loggingen. Hvis det er første gang brukeren trykker, kjøres kode som setter innstillinger på sensorene, og lager ruter, likt som i strømming av data, bortsett fra at dataen settes til å bli logget. Etter sensorene har startet, blir LED-lyset på sensoren bedt om å lyse grønt for å indikere at loggingen har startet.

Hvis det er andre gangen brukeren trykker på knappen, skjer det motsatte og loggingen stopper. her lyser også LED-lyset men denne gangen med fargen rød. Stopping av loggingen skjer på identisk måte som ved strømming, men det legges på en enkelt linje som sier at loggingen skal stoppe.

Nedlasting av logget data er heller ikke helt fremmed. Dette skjer også ved bruk av ruter, nesten på samme måte som ved strømming, men her startes ikke alle komponentene som ved strømming siden all data allerede eksisterer. Det som skjer er at jeg lager en «Anonym rute», dette er en spesiell rute man må bruke, for når telefonen ikke var tilkoblet sensorene mens den produserte data. Dermed så ber jeg sensorene sende meg all dataen de har logget, forså at jeg behandler dataene på samme måte som ved strømming. Jeg får også oppdateringer på hvor langt på vei nedlastingen er, og dette sender jeg videre til listenere for å oppdatere en nedlastings bar i brukergrensesnittet.

Når det kommer til nedlasting av logget data, støtte jeg på et problem hvor nedlastingen henger seg opp. Etter jeg hadde brukt et stort antall timer på å feilsøke dette, bestemte jeg meg for å kontakte produsenten av sensorene. Etter å ha snakket med produsenten av sensorene, skjønte de heller ikke hvorfor nedlastingen hengte seg opp, og det eneste svaret de kunne gi meg var å bytte telefon. Dette er ikke noe jeg har ressurser til å gjøre, så derfor kan det hende at nedlasting av logget data ikke fungerer helt som det skal. Jeg må dessverre legge videre testing og utarbeiding av dette bak meg for dette prosjektet, og håpe at produsenten har rett i at det er min telefon det er feil med.

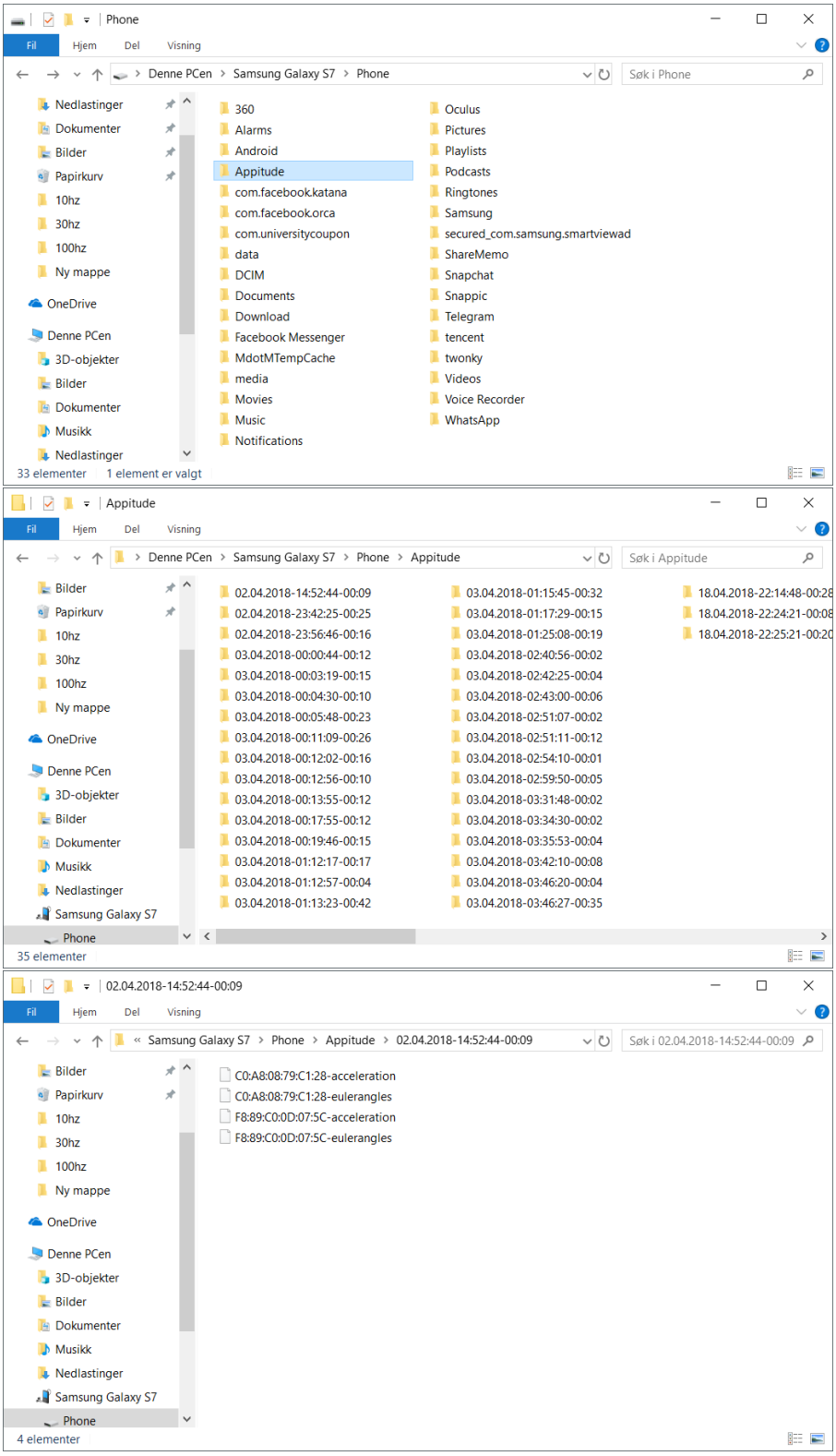
6.6 Filhåndtering og lagring av data

For å kunne se tidligere økter på et senere tidspunkt, er man nødt til å lagre dataen et permanent sted. Android har to steder man kan lagre data permanent, dette er intern lagring, og ekstern, lagring. Intern lagring er lagring på selve telefonen, her er det bare applikasjonen som har skrevet filene, som har tilgang til dem, og brukeren av telefonen har ikke tilgang til disse filene. I intern lagringen er det vanlig å lagre midlertidige filer, samt filer man ikke ønsker at andre applikasjoner, eller brukeren skal ha tilgang til.

Android sin ekstern lagring er noe missvisende, ved ordet ekstern vil man anta at dette er en plass som ikke er på telefonen, men det er bare halveis riktig. Ekstern lagring når det gjelder Android, er hovedsaklig alt man kan se av lagringsplass da man kobler telefonen til en datamaskin. Denne lagringsplassen inneholder både lagring på selve mobiltelefonen, samt lagring på SD-kort. En stor forskjell ved intern lagring og ekstern lagring er at alle filene som tilhører applikasjonen slettes fra intern lagringen da applikasjonen avinstalleres. Ved den eksterne lagringen, kan filene beholdes.

Jeg har valgt å benytte meg av den eksterne lagringsplassen, som ligger på selve telefonen. Grunnen til at jeg velger å bruke denne er fordi dataene som lagres, er ikke noe som må hemmeligholdes. Samt er det gunstig for min del å lagre dataene her, slik at jeg beholder test filer under utviklingen, og at jeg ikke må legge inn test filer for hver gang jeg gjør en endring i applikasjonen. Det kan også hende at en utvikler kan utarbeide et annet kult prosjekt med bruk av de samme dataene som ligger lagret.

SensorReadings er ansvarlig for å skrive all data til filer. Skriving av filer skjer mens data mottas fra telefonen, dette er fordi det er begrenset av hvor mye minne man kan benytte seg av med en applikasjon, så all data som mottas skrives direkte til fil, og derreter kan Java sin «garbage collector» ta seg av jobben med å rydde opp i minnet. I figur 15 vises filsystemet til applikasjonen.



Figur 15: Filsystemet for Appitude

Som vi kan se i figur 15, dannes en mappe for hver økt. Mappen får navn basert på dato, tidspunktet målingen startet, og lengden på økten. Deretter har vi 2 filer for hver av sensorene inni disse mappene som inneholder målet data. Filene som holder målet data er er oppkalt etter MAC-adressen til sensoren som målte, samt hvilke type data filen inneholder.

Jeg har valgt å benytte meg av binære filer for å gjøre ting lettere. Bruk av binære filer gjør slik at jeg kan skrive hele objekter til fil, og lese hele objekter fra fil. Dette eliminerer nødvendigheten med å skape struktur i filen. Den negative siden med å benytte binære filer er at disse tar større plass, objektene som skrives til fil har en satt størrelse, som benyttes uansett om objektet har behov for så stor plass eller ikke. For å redusere størrelsen på filene, har jeg implementert en filkomprimerings metode under navnet «Gzip», denne bruker en algoritme som er basert på «DEFLATE»-algoritmen, som er en kombinasjon av «LZ77»-algoritmen og den mer kjente «Huffman Coding» algoritmen. Det Gzip gjør i praksis, er at den lager seg referanser for gjentatte mønstre i filen. Dette er en metode for komprimering som ikke innebærer risiko for tap av data.

Java kommer med et Gzip biblotek som standard, og alt man trenger å gjøre er å filtrere strømmen som skriver til fil, med Gzip bibloteket. Denne algoritmen er ikke kjent for å gi de beste resultatene når det kommer til komprimering, men mer at det gir gode resultater fra begge verdner da det kommer til komprimeringen, og hastigheten på komprimeringen.

Til slutt har jeg også lagt på en mulighet for å lagre data i tekst format, denne muligheten er lagt inn med utvikling og testing i tankene, og kan bare stilles inn i rent kode format, altså er ikke dette en innstilling som er ment for brukeren. Tekst formatet som lages, er en vanlig .txt fil som kan limes rett inn i programmer som excel slik at dataene kan studeres.

6.7 Implementasjon av algoritme

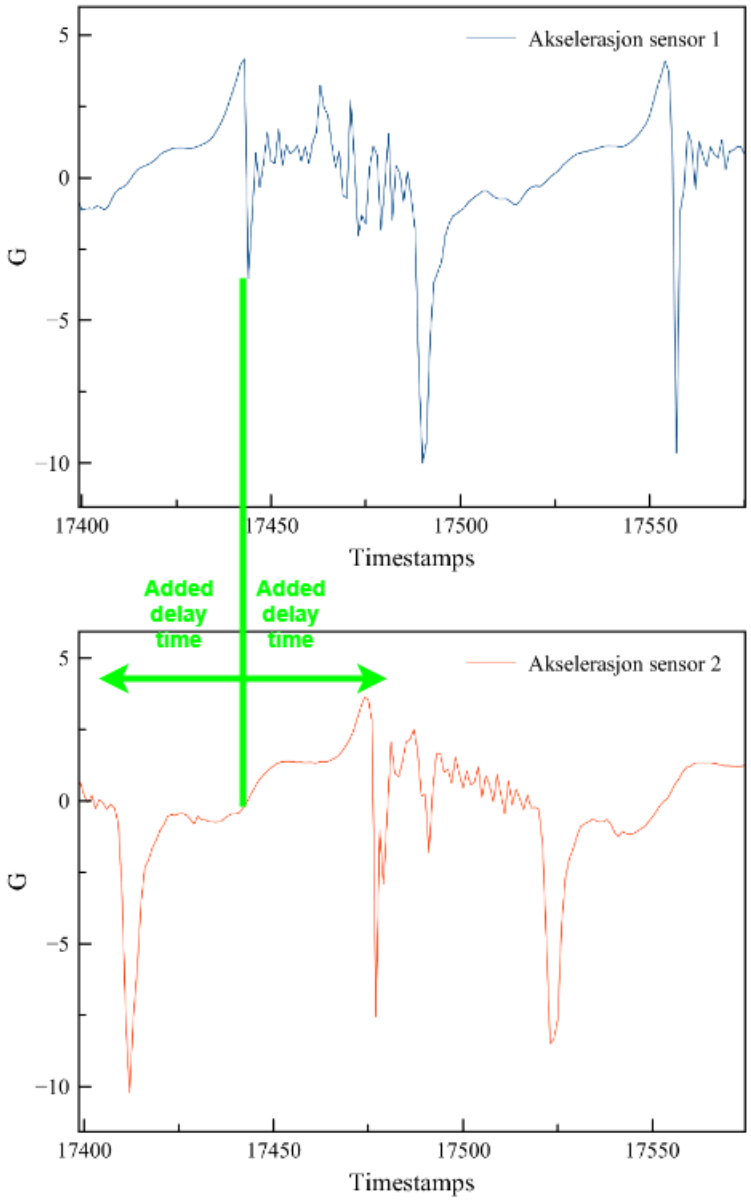
Gjenkjenning av et stavtak, er noe vi så på i kapittel 3.4. Her fant vi ut hvordan akselerasjonen så ut under de ulike fasene av stavtaket. Som vi vet fra dette kapittelet, får akselerasjonen et hopp i positiv retning kun når staven har sluppet bakken, og er på vei inn i et nytt stavtak. Staven får hopp i negativ retning i to tilfeller, da den treffer bakken, og da den slipper bakken.

Med denne informasjonen, må jeg lete etter et hopp i positiv retning for å kunne identifisere starten på et stavtak, siden hoppet med høye positive verider, er noe som bare skjer én gang i løpet av et stavtak. Måten jeg starter gjenkjenning av et stavtak på, er at jeg leter etter en akselerasjons verdi som er på 3G eller høyere. Hvis en akselerasjons verdi på 3G eller høyere er funnet, vil jeg gå ut ifra at jeg har funnet fasen hvor staven er på vei inn i et stavtak.

Da jeg har har funnet et punkt jeg kan anta er del av fasen hvor staven tas inn i et nytt stavtak, søker jeg meg opp til den følgende høyeste punktet i akselerasjonen, forså den laveste verdien, som da antas å være da staven treffer bakken.

Alt dette søkes opp på én av senorene sine akselerasjons data, hvis det ble funnet et punkt som antas å være da staven treffer bakken på denne sensoren sine data, vil det foretas et søk på den andre sensoren sine akselerasjons data i samme tidspunkt som da første sensoren registrerte et antatt punkt for da staven traff bakken. Her gis det litt

margin på tidspunktet, siden sensorene ikke er synkronisert til punkt og prikke, samt kan det ikke antas at brukeren er synkronisert med at begge stavene treffer bakken helt samtidig. I figur 16 kan vi se at stavtakene fra begge sensorene ikke er helt synkronisert.



Figur 16: Akselerasjons data fra begge sensorer under samme tidsperiode

Som vi ser, er det rundt 40 timestamps forskjell mellom målingene i sensorene. Med 100 hz målinger, altså 100 målinger i sekundet, vil dette si rundt 0.4 sekunders forskjell. Her har jeg ingen måte å vite hvor mye forskjell det er fra da brukeren faktisk setter stavene i bakken. Men jeg har foretatt tester, hvor sensorene holdes i en hånd, og ved sammenligning av data kan vi regne med at det er mindre enn 100 millisekunders avvik mellom synkroniseringen av sensorene.

Hvis stavtak blir funnet for begge stavene i rundt det samme tidspunktet, vil algoritmen gå videre til å finne ut når staven slipper bakken. Hvis derimot det ikke ble funnet et stavtak for den andre sensoren, vil stavtaket bli forkastet. For å finne ut når staven slipper bakken, leter jeg meg frem til et hopp i negativ retning fra etter tidspunktet staven traff bakken. Jeg har lagt inn at punktet ikke kan være rett etter staven har truffet bakken, fordi det kan fremdeles forekomme negative verdier for samme kurve som da staven traff bakken. Hvis ikke sensoren finner punktet hvor staven slipper bakken en liten periode etter staven traff bakken, vil stavtaket forkastes. Jeg har satt denne verdien til å være 1.5 sekunder.

Hvis det blir funnet et punkt hvor staven slipper bakken på det første data settet som undersøkes, vil det foretas et søk på den andre staven for å se om det også er å finne her. Dette foretas på samme måte som da staven treffer bakken med samme avviks verdier.

Da algoritmen finner punktet da staven slipper bakken på begge staver, vil dette telles som et gyldig staketak, og tidspunktet for de kritiske punktene lagres unna.

Da algoritmen har gått gjennom hele akselerasjons filen, vil den ta fatt på å finne vinklene på staven for de kritiske punktene, for hvert av de gyldige staketakene. Dette gjør den ganske enkelt ved å åpne Euler-vinkel målingne, og søker opp målingen av vinkel ved hjelp av tidspunktet for det kritiske punktet. Det kan ikke antas at Euler-vinklene har tilsvarende tidspunkter på alle målingene som akselerasjonen, derfor tar den det nærmeste tidspunktet den finner. Da den finner den Euler-vinkelen nærmest det kritiske punktet, lagres Euler-vinkelen unna. Da dette er gjort for alle kritiske punkter for hvert av staketakene, er utledning av data ferdig, og utledet data gjøres tilgjengelig i et «sessionInformation» objekt, slik at utledet data kan hentes ut og plottes i grafer.

Jeg kan som sagt tidligere ikke ha alt av data i minnet samtidig. Av den grunn har jeg lagt inn et maksimum antall målinger jeg kan ha i minnet samtidig. Jeg opererer også ved noe jeg har kalt for «Frames». Dette er en liten bit av data målingene. Størrelsen på ett frame, er avhengig av andre konstanter i algoritmen, og kan regnes ut på følgende måte:

$$\text{FRAME\_SIZE} = \frac{1000\text{ms}}{\text{MAX\_POLE\_PLANT\_FREQUENCY}} - \text{TOTAL\_DELAY}$$

Med denne utregningen, vil det maksimalt være ett stavtak i en frame. Altså må det settes en maksimal frekvens man kan ha på stavtakene, også må det trekkes fra alle synkroniserings avvik, slik at dette blir gyldig for data fra begge sensorer.

Denne algoritmen bruker litt tid på å fullføre, og siden jeg har valgt å lagre rådataene fra sensorene, må algoritmen kjøres for hver gang man skal se på en økt. Dette er helt klart ikke optimalt, og man skulle heller ha kjørt algoritmen mens dataene mottas fra sensorene, og lagret den utledete dataen. Dette ville både redusert filstørrelsene, samt ville man ikke måtte ha ventet på at algoritmen skal gjøre seg ferdig for hver gang man skulle se en økt. Jeg valgte å beholde denne metoden, ved at man må kjøre algoritmen for hver gang, siden det gir bedre muligheter for videre utvikling, for når man eventuelt endrer på algoritmen kan man teste algoritmen på allerede innsamlet data med engang. Bruken av frames gjør det lett å endre til at algoritmen kjøres samtidig som man samler inn data.

6.8 Utledning av frekvens

Utledning av frekvensen på staketakene, skjer da man skal hente ut grafen for dette. Denne utledningen har tilgang til den totale lengden på økten, samt alle staketakene og tidspunktene for disse. Det den da gjør er at den lager seg punkter for hvert minutt i økten, og teller opp antall staketak for hvert minutt. I 6.7 kan man se funksjonen som utleder frekvensen.

```
public List<Entry> getFrequencyEntries(long duration){
    List<Entry> frequencyEntries = new ArrayList<>();

    int numberOfMins = ((int)(duration/FREQUENCY_INTERVAL))+1;

        for(int i = 0; i<=numberOfMins; i++){
            frequencyEntries.add(new Entry(i, 0));
        }
        Entry temp;
        for(Long polePlant : polePlants){
            int plantMinute = (int)(polePlant/FREQUENCY_INTERVAL);
            temp = frequencyEntries.get(plantMinute);
            temp.setY(temp.getY()+1);
        }

    return frequencyEntries;
}
```

Kode eksempel 6.7: Utledning av frekvens

Det er litt missvisende å kalle det frekvens, siden dette betyr «Svigning per sekund» eller i denne sammenhengen «Hendelse per sekund», men jeg prøvde å fremvise frekvensen per sekund, istedet for hvert minutt, og dette førte til data som bare byttet mellom 1 og 0, og var veldig lite givende.

6.9 Feilhåndtering

Feilhåndtering fikk jeg dessverre ikke så mye tid til å gjøre. Dette er noe jeg begynte å fikle med mot slutten av prosjektet, og andre ting som hastet mer kom i veien for å kunne fullføre noe her. Selv om jeg ikke fikk lagt inn noen komplett feilhåndtering, hadde jeg en plan om hvordan jeg skulle gjøre det, samt fikk jeg tiden til å prøve ut planen på ett av punktene det kan oppstå feil.

```
final static public String UNEXPECTED_DISCONNECT = "A1";
final static public String UNABLE_TO_CREATE_RAW_FILE = "B1";
final static public String UNABLE_TO_WRITE_RAW_FILE = "B2";

public void onErrorOccurred(String errorCode) {
    runOnUiThread(()->
        new AlertDialog.Builder(this)
                .setMessage("Feilkode:"+errorCode)
```

```
9                   .setPositiveButton("Ok", (dialogInterface, i)-> {
10                      // Required to have in order to have a basic negative button
11                      // that closes dialog upon pressed.
12                  })
13                  .create()
14                  .show()
15      );
16  }
```

Kode eksempel 6.8: Plan om feilhåndtering

Min plan for feilhåndtering var å benytte meg av listenere i alle klasser det kunne oppstå feil. Dermed, hvis det skulle oppstå en feil, ville det blitt sendt et varsel til alle som hørte på, med informasjon om hva som gikk feil i form av feilkoder. I kode eksempel 6.8 ser vi jeg fikk laget til noen feilkoder, samt et popup-vindu til brukerern, med beskjed om at noe gikk feil, dette er i en av klassene som hører etter feilmeldinger. Jeg implementerte dette inn for da sensoren mistet kontakt ufrivillig, og testet dette ut. Det fungerte som planlagt, men dette er dessverre så langt som jeg kom innen feilhåndteringen.

6.10 Testing

Stabiliteten til applikasjonen har blitt testet i et mangfold av timer under utviklingen. Jeg har etter hver eneste oppdatering, installert applikasjonen på nytt og testet ut at alt fungerer som det skal. Å utgi en beta versjon av applikasjonen, har forsåvidt vært uaktuelt, siden det kreves at man har sensorer fra Mbientlab for å kunne bruke applikajsonen. Derfor har jeg vært ene og alene om å teste ut stabiliteten til applikasjonen, og har bare hatt mulighet til å kunne teste på én type telefon.

Applikasjonen har ikke hatt noen tydelige problemer ved min testing, foruten det jeg har nevnt tidligere med logging, men dette er ikke noe jeg får utbredt eller gjort noe med i løpet av dette prosjektet. I mine øyne fremstår applikasjonen som stabil.

Sensorene har jeg bare hatt ett problem med under hele prosjektet. Problemet jeg har støtt på er at det kan være vanskelig å få til en oppkobling iblant. Dette løste jeg, som nevnt tidligere, ved at applikasjonen prøver å koble seg opp gjentatte ganger til en periode har gått. Bortsett fra dette, har ikke sensorene presentert noen problemer.

Jeg har hatt store begrensninger når det kommer til testing av algoritmen. Jeg eier ikke skiutstyr eller ferdigheter, så å teste selv har ikke vært aktuelt. Man kunne jo tro at jeg kunne anskaffe en person til å teste ut algoritmen for meg, men man kan også tenke seg til at jeg brukte flere uker på å få tak i en som var villig til å samle inn test data som jeg kunne studere. Samt var all snøen borte innen jeg fikk algoritmen ferdig, som også var en faktor som gjorde det litt vanskelig.

Jeg hadde et siste møte med oppdragsgiver 19. April, hvor vi snakket litt om hvordan applikasjonen hadde kommet seg. Oppdragsgiver stilte da også opp med rulleski, slik at vi fikk testet algoritmen. Ved de første test rundene hadde algoritmen liten suksess, da den ikke klarte å oppdage alle staketak. Jeg tenkte jeg skulle prøve å endre noen av terskel og avviks verdiene, og vi endte opp med en algoritme som klarte å oppdage alle staketakene som ble utført.

Det er nok mye mer testing som må til når det gjelder algoritmen. Her gjelder det å

finne riktige terskel verdier, og prøve seg frem. Her kunne det nok også lønnet seg å ha flere skiløpere for å teste.

7 Diskusjon

7.1 Resultat

I planleggingen av prosjektet, laget jeg meg mål for prosjektet. Noen av disse målene, er ikke blitt nådd i det endelige resultatet. I resultatmålet ble det satt at applikasjonen skal kunne måle akselerasjonen og farten på skiløperen, denne oppgaven satte jeg som laveste prioritet, siden det allerede er noe som finnes ved bruk av GPS måling. Siden dette var laveste prioritet, endte jeg med å ikke få tilstrekkelig med tid til å implementere denne funksjonen. Men som sagt er dette noe som allerede finnes, og det skal ikke mye til for å få implementert en løsning for å få til dette målet.

Oppdragsgiver ønsket seg en prototype, for å se hvordan et slikt produkt ville fungert i praksis. Jeg føler jeg har levert en fullverdig prototype som resultat av dette prosjektet. Det endelige produktet som jeg har utviklet, er et godt stykke unna å kunne kommersialiseres, men fremstår mer enn bra nok for å kunne presenteres som en prototype. Både jeg og oppdragsgiver er fornøyde da det kommer til resultatet av produktet som er utviklet.

Som resultat av dette prosjektet, har jeg tatt i bruk allerede kjente kunnskaper, men viktigst av alt har jeg tatt til meg en hel del nye kunnskaper. Prosjektet har basert seg mye på å kode i Java, dette er noe jeg allerede har erfaring med, men ikke rettet til Android. Det er veldig mye likheter med å programmere til Android, som ved Windows som jeg tidligere hadde erfaring med, men det er også et helt annet operativsystem, ment for en helt annen enhet. Dette gjør at det også er en haug av forskjeller mellom å kode til de forskjellige operativsystemene. Og ved slutten av prosjektet, føler jeg selfølgelig at jeg har utviklet min evne til å kode i Java, men jeg føler også at jeg har lært meg en helt ny kunnskap, som er å lage kode til mobiltelefon, uavhengig av programmeringsspråk.

Når det gjelder bruk av sensorene, føler jeg ikke at jeg har lært meg en generell kunnskap. Sensorene kom med et API som var veldig spesifikt rettet mot disse sensorene, så eneste kunnskapen jeg fikk ut av sensorene var å benytte meg av tilhørende API. Kunnskapen av å bruke dette spesifikke APIet er nok ikke en kunnskap jeg kommer til å få bruk for videre i livet.

7.2 Utviklingsprosess

Som beskrevet tidligere, har jeg benyttet meg av Scrum som systemutviklingsmodell. Den største delen av Scrum modellen er vel det at man jobber i sprints. For min del som har vært alene på prosjektet, har sprints og oppgavene i sprinten fungert bra som en måte å se progresjonen i prosjektet, samt holdt styr på hva jeg har å gjøre.

Jeg hadde nok litt for store forventninger til meg selv, da jeg planla at sprintene skulle inneholde 80 timer med oppgaver. Disse 80 timene inneholdt ikke tid til å drive med bakgrunns arbeid, som for eksempel å lese meg opp på stoffet jeg skulle gjøre, samt hadde jeg også et annet fag vedsiden av bacheloroppgaven. Dette har ført til at jeg nesten aldri klarte å fullføre alle oppgavene som lå i en sprint, og derfor følte det ikke helt som at jeg jobbet i sprinter. For min del føltes det mer ut som at hele bachelor perioden var én sprint, hvor det bare strømmet inn flere oppgaver.

Jeg tror nok at det er lettere å kjenne effekten av å ha delt opp oppgaven i sprinter og oppgaver, da man er flere på prosjektet. Mye av tanken bak dette er å kunne øke kommunikasjon og forbedre oppgavefordelingen på en gruppe [5], dette er noe jeg ikke har fått erfare, siden det er jeg som har måtte gjort alt.

Jeg har prøvd å logge alle dagene jeg har jobbet. Det er en del bakgrunnsarbeid som ikke er logget, hvor jeg har lest meg opp på hvordan jeg gjør ting. Jeg har også prøvd å lage noen «Sprint Burndown Charts» for sprint periodene, som kan ses i loggen. Det er ikke laget burndown charts til alle sprintene, siden ting begynte å gå litt skeis, men jeg har inkludert de jeg har laget, for å vise at det er noe jeg har hatt i tankene.

Det lå også i planene at jeg skulle ha fast arbeidstid hver ukedag. Dette er noe som ikke helt fungerte, og jeg benyttet meg mye av å velge mine egne arbeidstider, siden jeg var alene, hadde jeg friheten til dette. Sykdommen jeg har hatt under prosjektet har nok vært en medspillende faktor her, og har ført til at jeg har jobbet da jeg har klart det, og det har kunnet gått dager mellom når jeg har fått jobbet.

Utviklingsprossesen har for meg vært veldig hard, dette mye på grunn av sykdom, som har gjort det vanskelig for meg å holde rutiner, samt holde meg til planen. Det at jeg har vært alene på prosjektet har både vært en fordel og en ulempe. Jeg føler at det for meg har pekt mer i den negative retningen, fordi jeg får mer motivasjon til å jobbe da jeg har et ansvar for andre mennesker. Nå som jeg har vært alene om prosjektet, er det bare meg resultatet går utover.

Når alt kommer til alt, ville jeg foretrukket å ha en gruppe, og tror jeg kunne ha fått større utbytte av oppgaven når det kommer til utviklingsprosess og systemutviklingsmetode hvis jeg hadde det. Men jeg må også si at sykdommen har ødelagt mye for meg, og jeg er godt fornøyd med resultatet av oppgaven gitt de uforutsette hendelsene som har skjedd i løpet av prosjektet.

8 Konklusjon og videre arbeid

8.1 Konklusjon

Da jeg først påtok meg dette oppdraget, visste jeg ikke hvor mye arbeidsmengde jeg hadde fremfor meg. Det var ikke planlagt at jeg skulle være alene på prosjektet, men en uforutsett hendelse som oppstod tidlig. Personlig tror jeg at jeg ville fått mer ut av prosjektet, hvis jeg hadde hatt en gruppe. På denne måten ville jeg kanskje fått kjent virkningen av å ha en systemutviklingsmodell i større grad enn jeg har gjort.

Appitude har vist seg til å være et utfordrende prosjekt, både i form av programmering, design, og algoritmiske løsninger. Det mest utfordrende aspektet med prosjektet, har vært å tilvende meg å kunne programmere til Android, og her føler jeg at det har gått med en del tid.

Jeg hadde nok litt for store krav til meg selv under planleggingen av oppgaven, og jeg kan ikke si at jeg oppfylte mine egne forventninger til det endelige produktet. Men, jeg har fått inntrykk av at oppdragsgiver var fornøyd med resultatet av prosjektet, og det er nok det viktigste for meg.

Den endelige konklusjonen av dette prosjektet, må være at det har vært en lang og hard reise, men også det mest givende jeg har gjort i løpet av studie tiden min. Oppgaven har virkelig utfordret min evne til selvstendig jobbing, samt vært utrolig lærerik, og ikke minst interessant.

8.2 Videre utvikling

Applikasjonen har som sagt et stykke å gå. For videre utvikling ville jeg prioritert å få ferdig feilhåndteringen, samt få på plass GPS-måling av skiløperen sin hastighet og akselerasjon. Det kreves også mye videre testing av algoritmen, og det må nok ytterligere justeres verdier for detektering av kritiske punkter. En annen smart videre utvikling, ville vært at applikasjonen husket sensorene den skal bruke, slik at man slipper velge sensorer for hver gang man skal ha en økt. Dette er altså ting jeg ikke hadde tilstrekkelig med tid til å gjennomføre, men som er fult mulig å få til, og som jeg gjerne skulle hatt med i det endelige produktet.

Siden jeg endte opp som alene på prosjektet, avgrenset jeg bort store deler av den originale oppgaven slik at det skulle være en mer realistisk arbeidsmengde for én person. Et nytt prosjekt kunne vært å fullføre resten av det oppgaven originalt spørte om, samt muligens oversette slik at applikasjonen fungerer på tvers av operativsystemer.

Brukergrensesnittet benytter seg fullstendig av standard Java elementer. Dette er noe som godt synes, og applikasjonen kunne ha hatt godt av en grafisk designer til å oppdatere elementene, og gitt brukergrensesnittet et grafisk løft. Det hadde også vært kult med tilbakemeldinger på ytelsen til brukeren. Dette ville krevd en analyse av staking i sin helhet, for å finne de optimale vinklene for staven sin bevegelse. Vi vet også at det inngår endel kroppsbevegelser når det kommer til staking [6] og det kunne vært et fint videre prosjekt å få implementert dette samt, kraft inn i applikasjonen.

Bibliografi

[1] Wikipedia contributors. 2018. Aircraft principal axes — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Aircraft_principal_axes&oldid=868124208. [Online; accessed 30-November-2018].

[2] Sugerman, J., Venkitachalam, G., & Lim, B.-H. 2001. Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor. In *USENIX Annual Technical Conference, General Track*, 1–14.

[3] Wikipedia contributors. 2018. Copyleft — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Copyleft&oldid=867678706. [Online; accessed 12-November-2018].

[4] Apache Software Foundation. 2004. Apache License Version 2.0. https://www.apache.org/licenses/LICENSE-2.0. [Online; accessed 15-November-2018].

[5] Wikipedia. 2017. Scrum — wikipedia,. [På internett; besøkt 5-november-2018]. URL: https://no.wikipedia.org/w/index.php?title=Scrum&oldid=18026299.

[6] Danielsen, J., Sandbakk, Ø., Holmberg, H.-C., & Ettema, G. 2015. Mechanical energy and propulsion in ergometer double poling by cross-country skiers. *Medicine and science in sports and exercise*, 47(12), 2586–2594.

A Arbeidslogg

Loggbok:

Torsdag 01.02.2018
Grunnet at veileder ikke hadde tid denne dagen til å hjelpe til med estimering, så har jeg funnet alternative oppgaver som var å lage en skisse av GUI og sette opp alle verktøy jeg skal bruke, samt bli kjent med verktøyene. Timer jobbet 4 timer.

Fredag 02.02.2018
Møte med veileder, estimert oppgaver for første sprint. Laget ca 60% av GUI og gjort klart for oversetting til flere språk i programmet. Timer jobbet totalt inkludert møte 5timer.

Torsdag 08.02.2018
Fortsatt på GUI, nærmer seg slutten for GUI og har påtatt ny oppgave vedsiden av GUI som er å lage funksjoner til GUI'et. Morgendagen vil antageligvis GUI bli ferdig og funksjoner til GUI vil bli gjort ferdig så langt det kan før jeg får inn apiet for sensorene. Timer jobbet: 8timer.

VALG: Metode for deklarering av OnClickListener bestemt, valgt å definere listenere i java kode fremfor å referere til metode fra XML, dette grunnet at å referere til metode fra XML kan skape problemer med refactor i fremtiden og er mer uoversiktlig.

Lørdag 10.02.2018
Sagt med ferdig med GUI for nå. Sammenlignet ulike bibloteker for grafer. Gir et forsøk på å bruke graphview. Implementert og testet ut bibloteket. Implementert API for sensorene. Timer jobbet: 6timer.

VALG: Valget av biblotek til grafer var ikke lett, ettersom det finnes så mange bibloteker for dette. Måten jeg valgte på var å søke opp de mest populære biblotekene og sammenligne disse ved hjelp av hva jeg tror jeg kommer til å trenge, samt ved hjelp av condix.io for å se en sammenligning av biblotekene. Mine krav til bibloteket var at det skulle være gratis, passende lisens for bruk i dette prosjektet, og ha funksjonaliteten jeg absolutt trenger. funksjonaliteten jeg trenger er å kunne fremvise endringer i verdier over tid.

Bibloteker som sammenlignes:
- achartengine
- androidplot
- graphview
- MPAndroidChart

AChartEngine har ikke blitt oppdatert på ett år, og har liggende open bugs i android, så hvis det skulle bli problemer med dette bibloteket i fremtiden må man rette opp i dette selv.

androidplot har best utfall på condix.io, bibloteket skal være enkelt å bruke, ha god brukerstøtte og stabilitet. Bibloteket blir oppdatert av utviklerne til den dag idag og er en god kandidat. androidplot tilbyr all funksjonalitet som kreves og mer til.

GraphView skal være veldig enkelt å bruke, men har begrenset funksjonalitet. Dokumentasjonen for bruk av bibliteket er på topp, men ytelsen til dette bibloteket ligger litt under de andre. Bibloteket blir aktivt vedlikeholdt.

MPAndroidChart dette er uten tvil det mest populære bibloteket som blir brukt til graf håndtering. Bibloteket tilbyr all funksjonalitet man kan ønske seg, og ligger over gjennomsnittet på condix.io.

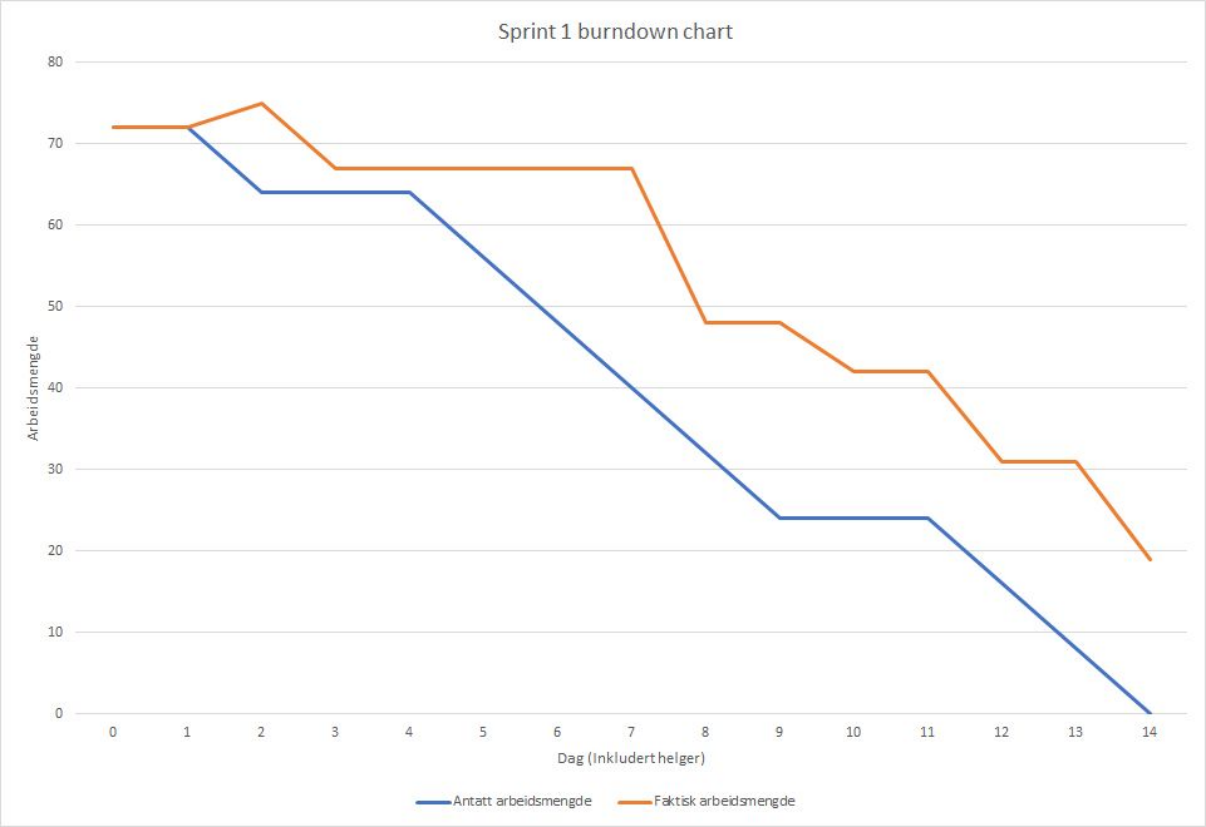
AChartEngine vil jeg ikke bruke grunnet at de sluttet å oppdatere bibloteket. GraphView virker som det er veldig enkelt og rettfrem, men er også veldig begrenset på hva man kan gjøre. Mitt valg vil stå mellom MPAndroidChart og androidplot, her er det vanskelig å velge men jeg går for MPAndroidChart fordi det har flere brukere som igjen fører til mer dokumentasjon og støtte, bibloteket er også tilgjenglig på andre platformer som IOS, som gjør at dette bibloteket kan bli brukt hvis prosjektet skal videreutvikles etter endt prosjekt. MPAndroidChart er også det høyeste rated verktøyet på android arsenal. https://android-arsenal.com/tag/40?sort=rating
https://stackoverflow.com/questions/26467376/android-charting-libraries

Mandag 12.02.2018 Vært på møte med veileder(1 time). Jobbet med utvikling i 7 timer. Omgjort hele GUI'et

Tirsdag 13.02.2018 Jobbet med å søke etter bluetooth enheter, men har ikke kommet noen vei. Arbeidet 6 timer, så fikk jeg migrene.

Onsdag 14.02.2018 Jobbet videre med å søke etter bluetooth enheter, fikk til det meste og litt småplukk igjen på denne delen imorgen. 12 timer arbeid.

VALG: Funksjoner for bluetooth scanning for android SDK 4.4(API 19) er depricated, og eneste andre alternativ er et biblotek som ikke blir introdusert før SDK 5.0(API 21). Jeg velger å lage en løsning for begge her, slik at appen fungerer på android 4.4(API 19), men de som har SDK 5.0 eller nyere vil kunne ta nytte av bedre ytelse som ble introdusert med 5.0 (API 21) bibloteket.



Kommentarer til sprint: Mange av oppgavene var dårlig estimert og tok lengre tid enn forventet. Gamle oppgaver som ikke ble fullført ble flyttet over til neste sprint. Oppgaven “Gjøre klart til visualisering av data” ble satt på hylla til neste sprint grunnet at jeg trengte info fra arbeidsgiver, men arbeidsgiver har vært utilgjenglig denne sprinten, Oppgaven ble byttet ut med “Gjøre om GUI” siden jeg ikke ble fornøyd med første versjon av GUI’et, denne oppgaven hadde et tids estimat på 8T, gjenstående arbeid på “visualisering av data” var estimert til 12T. Det har også gått bort mye tid til faget jeg har vedsiden av bacheloren denne sprinten, og jeg har derfor ikke fått jobbet med bacheloroppgaven så mye som jeg hadde ønsket.

Torsdag 15.02.2018 Jobbet med å sette opp ny sprint og jobbet videre med å søke opp bluetooth enheter, implementert metode for API level 21 og høyere, og fikset alt gjenstående arbeid her. Arbeid 8t

Fredag 16.02.2018 Jobbet med å koble til sensorene, fikk til å koble meg opp og starte en oppgave, nå gjenstår det å få til at enhetene strømmer og laster ned logget data. Arbeid 8t

Mandag 19.02.2018 Lest meg opp på hvordan jeg skal strømme og logge data. Arbeid 4t
Tirsdag 20.02.2018 Lest meg opp på hvordan jeg skal strømme og logge data, samt hvordan jeg skal programmere knappen til å starte/stoppe logging av data. arbeid 4t

Onsdag 21.02.2018 Møte med Ivar, blitt enige om at jeg skal lage en applikasjon på siden for å se hvordan de forskjellige sensor funksjonene oppfører seg. Arbeidet med side applikasjon. arbeid 8t.

Torsdag 22.02.2018 Laget ferdig side applikasjon som viser hvordan SensorFusion funksjonen til sensor brettet sin data oppfører seg. Presentert applikasjon til Ivar. Arbeid 8t.

Fredag 23.02.2018 Jobbet med faget jeg har vedsiden av.

Lørdag 24.02.2018 Jobbet videre med å opprette streaming 4t arbeid.

Søndag 25.02.2018 Jobbet videre med å opprette streaming, så og si ferdig, har også laget et reconnect vindu for når oppkobling feiler. 7t arbeid

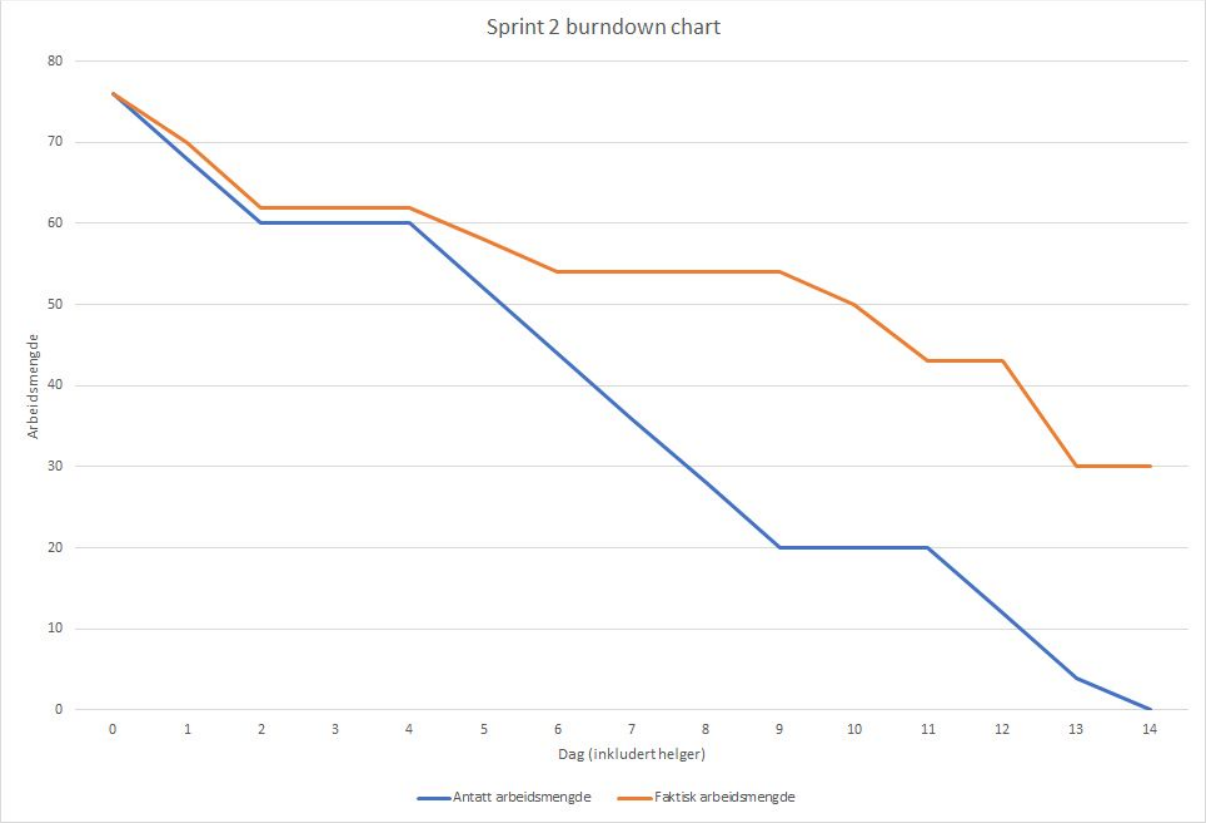
Mandag 26.02.2018 Jobbet med faget jeg har vedsiden av.

Tirsdag 27.02.2018 Jobbet med å starte/stoppe logging av data ved at knappen på sensorene trykkes på, fikk til dette og nedlastning av logget data. Jobbet 13t arbeid.

Onsdag 28.02.2018 Vært på møte med Ivar, hovedsaklig status oppdatering. Fikk migrene resten av dagen etter møtet.

Torsdag 01.03.2018 Ny sprint skulle være ferdig oppsatt til idag, men grunnet migrene dagen før ble det lite gjort, sprint planlegging skjer derfor idag. Idag har jeg også

Oppgaver som ikke ble ferdig på forrgie sprint blir flyttet over til den nye sprinten.



Det som ikke ble ferdig er visualisering av data, denne oppgaven ble aldri påbegynt ettersom jeg avventet svar fra arbeidsgiver, og for øyeblikket lurer jeg på om jeg skal ta denne oppgaven litt som jeg utleder dataene og ser hva som kan passe best.

Testing av applikasjonen er også uferdig, dette fordi applikasjonen ikke er ferdig, men den blir testet underveis, så dette er mer en oppgave som går med under alle sprinter ettersom mer blir gjort.

Synkronisering av sensor data har jeg rett og slett ikke hatt tid til å gjennomføre.

Det kom også en uforutsett oppgave under denne sprinten som var å lage en side applikasjon for å se hvordan sensorene oppfører seg ved bruk av sanntids grafer og streaming, denne oppgaven gikk det 2 dager på (16t).

Fredag 02.03.2018
Satt opp datastruktur i minnet for innsamlet data og testet meg litt rundt å samle inn data. Jobbet 4T

VALG: Det er tid for å bestemme datastrukturen til innsamlet data. Egenskapene til datastrukturen er at den skal kunne holde mye data, og det må kunne søkes i dataene.

Mandag 05.03.2018
Oppstått komplikasjoner med logging. nedlastning av loggede filer tar altfor lang tid, sett på muligheter for å bringe denne tiden ned. 8T arbeid.

Tirsdag 06.03.2018
Fortsatt arbeidet fra mandag og jobbet med å finpusse applikasjon. Arbeid 8T

VALG:
Nedlastning av filer ved 100HZ logging tar fryktelig lang tid, eneste mulige løsning jeg fant var å skru ned HZ av loggingen, hvilken HZ jeg skal bruke vet jeg ikke enda, ettersom jeg må samle inn litt data på forskjellige frekvenser for å se hva som er brukanes.

Onsdag 07.03.2018
Vært på skolen store deler av dagen pga møte med veileder og kurs i bacheloroppgave skriving. Jobbet litt med å finpusse applikasjonen mellom timer. 2T arbeid på oppgaven.

Torsdag 08.03.2018
Finpusset applikasjonen. arbeid 4T

Fredag 09.03.2018
Gjort forberedelser for å måle test data. Applikasjonen som produsenten har laget er veldig ustabil og det ønsker jeg derfor ikke bruke da jeg skal måle mine test data. Har derfor sett på å lage til slik at jeg kan bruke min applikasjon til denne data innsamlingen. Arbeid 8T

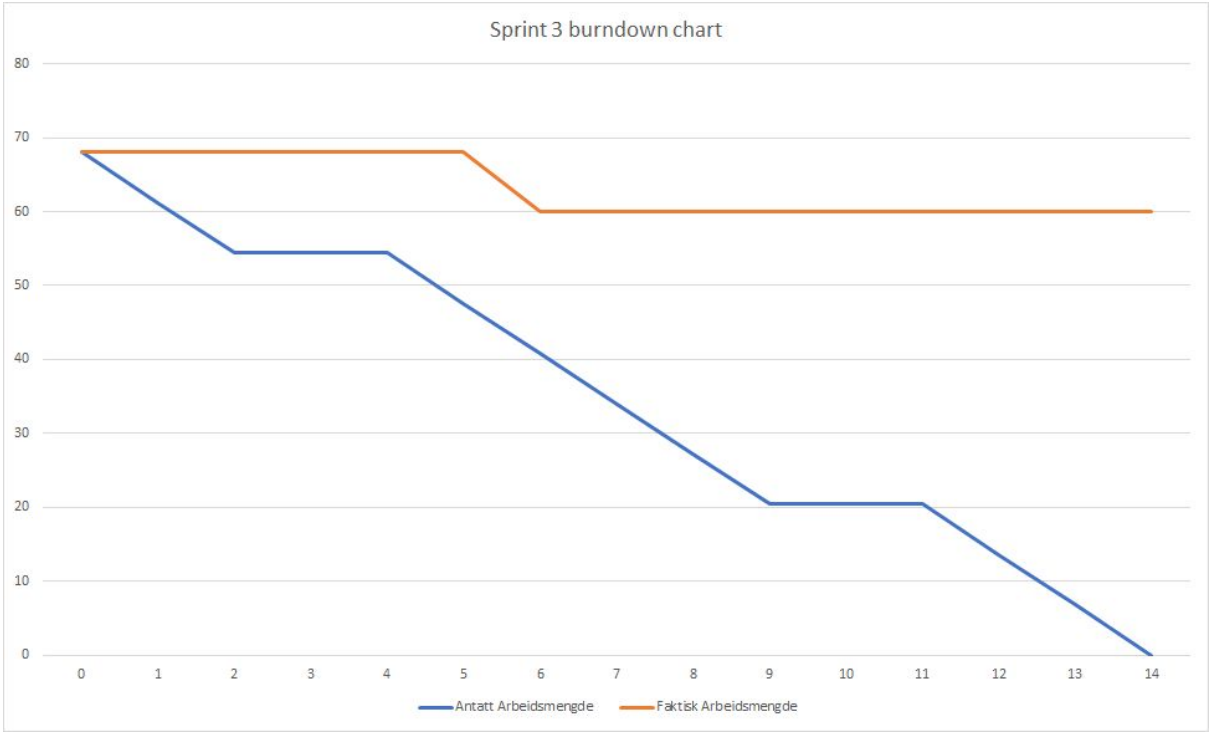
Mandag 12.03.2018

Fant ut at det er begrenset til 16-40MB man kan bruke av minnet på engang på en andorid telefon. Dette løser jeg ved å lage en cache fil som holder på måledata. Har laget slik at applikasjonen lagrer all sensor data til cache filer. Har også satt opp slik at sensor data skrives til tekst fil slik at jeg kan bruke min applikasjon til å samle inn test data. 12T

Tirsdag 13.03.2018
finpusset applikasjonen. arbeid 4T

Onsdag 14.03.2018
Vært på møte med Ivar. Avdekket 2 feil i applikasjonen, den ene er en memory leak og den andre er at nedlastning av loggede data stopper ut av det blå uten feilkoder. Løst memory leak feilen, denne lå i objectoutputstreamen som holder referanser til alle objektene den skriver til fil, fjernet det at den holder på referansene ettersom det ikke er noe jeg har bruk for. Arbeid 12T

Torsdag 15.03.2018
Jobbet med feilen hvor logging stopper ut av det blå, har ikke funnet løsning og begynner lure på om det kan være en firmware feil, tatt kontakt med produsentene av API/Sensorer og avventer svar. Har gått over til å lage Unexpected disconnect handler slik at streaming er komplett til testing imorgen. Arbeid 10T



Denne sprinten gikk ikke som planlagt grunnet mange uforventede oppgaver, og det at jeg ikke fikk tak i løper under sprinten som ødela endel. Men jeg har vært veldig produktiv med å løse andre ting og finne andre oppgaver å gjøre mens jeg hadde mangel på løper som gjorde slik at jeg ikke fikk utført de planlagte oppgaver.

Fredag 16.03.2018

Vært på Vind med frivillig fra Gjøvik-Toten langrenn som samlet inn data for meg ved bruk av sensorene, data ble samlet inn på 10, 20, 30 og 100 hz. Jobbet med å få plottet dem. 4T

Mandag 19.03.2018
Jobbet med å plotte og studere data fra fredagen. 8T

Tirsdag 20.03.2018
Jobbet med å synkronisere sensor data. 4T

Onsdag 21.03.2018
Vært på møte med Ivar og snakket om mulighetene for å detektere stavtak. Går for en metode med frames som beskrives nærmere i bachelorrapporten. Arbeidet videre med synkronisering av sensor data, og bestemt meg for sammen med ivar at avviket er så lite at det ikke er nødvendig. 4T

Torsdag 22.03.2018
Jobbet med å fikse bug meg logging etter svar fra produsentene. Deres fix funket ikke og avventer nytt svar. 4T

Fredag 23.03.2018
Implementert metode for detektering av stavtak. Metoden er noe uferdig men fungerer!. 8T arbeid.

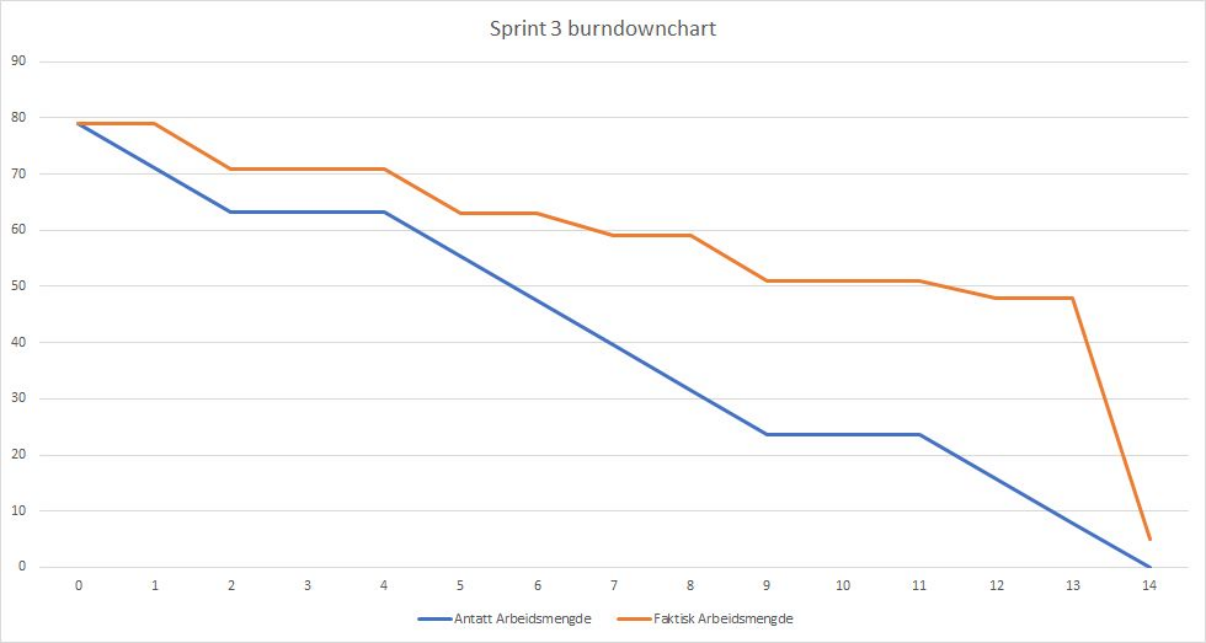
Mandag 26.03.2018
Optimalisert implementasjon. Funnet feil med å skrive til fil, jobbet med denne. Har også funnet fin metode for frekvens, implementert denne. 8T.

Tirsdag 27.03.2018
Fikset noen bugs og laget filstruktur til programmet. 8T.

Valg: Bestemt meg for å lagre rå sensor data til filer å behandle dem på nytt hver gang de skal ses. Tiden det tar å behandle dem er litt lang, så skal derfor se om jeg kan optimalisere her etterhvert.

Onsdag 28.03.2018
Omgjort datastruktur litt og fikset med følgene bugs som skjedde pga dette. har også funnet metode for å finne vinker og implementert denne. 8T

Torsdag 29.03.2018
Jobbet med å visualisere en sesjon. 8T.



Denne sprinten gikk litt sakte i starten men i slutten så tok det av. Jeg fikk gjort mer eller mindre alt jeg hadde planlagt og mer til. Var mye av estimasjonene som var for høye. Det som gjenstår er å fikse en bug i algoritmen for å gjenkjenne et staketak.

02.04.2018 - 06.04.2018
Ikke fått jobbet grunnet sykdom.

Mandag 09.04.2018
Jobbet med rapporten. 4T
Tirsdag 10.04.2018
Jobbet med rapporten. 4T

Onsdag 11.04.2018
Vært på møte med Ivar samt vært hos lege grunnet sykdom.

12.04.2018 - 19.04.2018
Sykdom. Anskaffet legeerklæring på sykdom fra 18.03.2018 til 18.04.2018 som gjelder at jeg ikke har kunnet jobbe med bacheloroppgaven av medisinske grunner, har dog jobbet det jeg klarte i denne perioden.

Onsdag 18.04.2018
Møte med ivar.

Torsdag 19.04.2018
Møte med oppdragsgiver. Testet den ferdige applikasjonen i en faktisk stakeøkt (rulle ski) som var en suksess.

Fredag 20.04.2018
Jobbet med rapport 4T

Søndag 22.04.2018
Jobbet med rapport 2T.

Fra dette punktet ble det bestemt at jeg ikke kunne fullføre oppgaven dette semesteret på grunn av medisinske grunner.

Mandag 05.11.2018
Jobbet med rapport 3T

Tirsdag 06.11.2018
Jobbet med rapport 2T

Onsdag 07.11.2018
Jobbet med rapport 2T

Torsdag 08.11.2018
Jobbet med rapport 3T

Fredag 09.11.2018
Jobbet med rapport 1T

Lørdag 10.11.2018
Jobbet med rapport 4T

Mandag 12.11.2018
Jobbet med rapport 1T.

Tirsdag 13.11.2018
Jobbet med rapport 3T.

Onsdag 14.11.2018
Jobbet med rapport 2T

Fredag 16.11.2018
Jobbet med rapport 3T

Lørdag 17.11.2018
Jobbet med rapport 8T.

Søndag 18.11.2018
Jobbet med rapport 15T.

Tirsdag 20.11.2018
Jobbet med rapport 8T.

Onsdag 21.11.2018

Jobbet med rapport 2T.

Torsdag 22.11.2018
Jobbet med rapport 1T.

Fredag 23.11.2018
Jobbet med rapport 8T.

Lørdag 24.11.2018
Jobbet med rapport 15T.

Søndag 25.11.2018
Jobbet med rapport 20T.

Mandag 26.11.2018
Jobbet med rapport 5T.

Torsdag 29.11.2018
Fikset litt manglende Javadoc og oversetting i applikasjonen. 1T
Jobbet med rapport 15T

B Planlegging

Planlegging

Bent Holden

Innhold

1 Mål og rammer

1.1 Bakgrunn

Det har lenge vært mulig å kunne måle ytelsen til personer i ulike aktiviteter som for eksempel jogging, hvor man kan måle blant annet antall skritt, hastighet, distanse også videre. Innenfor skisporten er staking blitt mer og mer populært, dette er hvor man tar et stavtak med begge staver samtidig.

Staking har en mangel på et verktøy som kan måle ytelsen til en person, det ønsker Appitude å gjøre noe med. Jeg har derfor fått i oppdrag av Appitude å lage et verktøy som kan måle staketeknikk og ytelse. Verktøyet skal lages slik at en som driver med staking som hobby kan bruke det, like godt som profesjonelle.

Oppdraget beskriver at det skal festes sensorer på skistavene som sender rådata til en mobiltelefon, videre til en server som analyserer rådataene fra sensorene og gjør om dette til lesbare data, som deretter sendes tilbake til mobiltelefonen slik at bruker kan se.

Det ønskes også fra oppdragsgiver at man skal kunne bruke både mobiltelefoner med IOS og android, samt ønskes det en webapplikasjon som kan vise frem de lesbare dataene fra skiøkten.

Sensorene som tildeles er fra mbientlab og er av typen MetaMotionR, denne kommer med akselerometer, gyroskop, barometer, magnetometer og termostat.

1.2 Oversikt over oppdraget

Oppdragsgiver ønsker å nå følgende delmål og undermål under dette prosjektet:

1. Prossesere data
   (a) Antall staketak
   (b) Vinkel på skistavene i ulike faser av staketaket
   (c) Akselerasjon på skistavene i ulke faser av staketaket
   (d) Akselerasjon på skiløper
   (e) Kraft fra da bruker skyver seg fremover
   (f) Frekvensen på staketakene
   (g) Hastighet på skiløper
2. Lage mobilapplikasjon

(a) Kommunikasjon mellom mobil og sensorer

(b) Kommunikasjon mellom mobil og server

(c) Visualisering av prosessert data

3. Lage server

(a) Kommunikasjon mellom mobil og server

(b) Kommunikasjon mellom server og webapplikasjon

(c) Implementere prosessering av rådata

4. Lage webapplikasjon

(a) Kommunikasjon mellom server og webapplikasjon

(b) Visualisering av prosessert data

1.3 Prosjektmål

1.3.1 Resultatmål

Mine resultatmål i dette prosjektet er å utvikle en mobilapplikasjon som kan strømme rådata fra sensorene og vise frem behandlet data til bruker. Jeg ønsker også å behandle rådata fra sensorene. Altså er mine resultatmål delmål 1 og delmål 2.

1.3.2 Læringsmål

En bacheloroppgave er en oppgave man skal ta i bruk kjente kunnskaper, men samtidig lære nye ut fra hvilken retning man tar oppgaven. Jeg ønsker å få mer erfaring med å planlegge og utføre større prosjekter, jeg har også lyst å lære mer om utvikling av programvare til mobiltelefoner, særlig til android platformen.

1.3.3 Effektmål

Med dette prosjektet ønsker oppdragsgiver å oppnå mer motivasjon for skiløpere som driver med staking samt gi brukere et bedre verktøy for å forbedre staketeknikken sin. Motivasjonen skal komme fra at brukeren skal kunne se sin egen forbedring over tid.

Det er planer om å videreutvikle prosjektet etter endt prosjekt periode og implementere det slik at en bruker kan sammenligne seg selv med andre, dermed kan brukeren sammenligne seg selv med for eksempel profesjonelle og se hva de kan forbedre seg på, men dette er ikke inkludert i dette prosjektet.

2 Omfang

2.1 Fagområde

Som ingeniør, skal man ha innsikt i mange fagområder. Denne oppgaven utfordrer mange kjente og ukjente kunnskaper, som er innenfor applikasjonsutvikling, matte, fysikk, og muligens maskinlæring ettersom hvilke retning man tar oppgaven, som alt er innenfor fagområdet til en dataingeniør.

2.2 Avgrensing

Grunnet at jeg er en enkelt person på dette prosjektet som egentlig var beregnet på to til tre personer, vil jeg ikke ha mulighet til å levere et godt gjennomført resultat av alt oppdragsgiver ønsker seg. Derfor har jeg blitt enig med oppdragsgiver om at oppgaven avgrenses til delmål 1 og delmål 2.

Grunnen for at delmål 1 er blitt valgt er at oppdragsgiver finner dette mest matnyttig, samt er det veldig sentralt i oppdraget. Delmål 2 er blitt valgt på grunn av min personlige interesse om hva jeg ønsker lære ut fra prosjektet.

2.2.1 Avgrensing innenfor delmål 1

Jeg går nå utfra at all data som ønskes kan bli funnet ved bruk av de sensorene jeg er gitt tilgang til, men hvis jeg finner ut at dataene ikke gjør seg mulig å bli funnet, eller gir et utroverdig resultat, vil jeg eventuelt benytte funksjoner på mobiltelefonen hvis det er hensiktsmessig.

Det har også alt blitt enighet med oppdragsgiver om at kraft fra da bruker skyver seg fremover ikke vil være mulig uten en kraftmåler, som ikke er inkludert i sensorer som gis tilgang til, derfor utelukkes denne målingen fra oppgaven.

2.2.2 Avgrensing innenfor delmål 2

Jeg velger å avgrense mobilapplikasjonen til å holde fokus på android plattformen. Oppdragsgiver ønsker en applikasjon som fungerer på kryss av plattformer, men jeg kommer til å bruke det programmeringsspråket jeg finner mest hensiktsmessig uavhengig om det vil fungere på kryss av plattformer eller ikke.

Grunnet at en dedikert server for å analysere data ikke lenger er en del av oppdraget, vil jeg implementere analysen av data fra sensorene i mobilapplikasjonen.

2.3 Oppgavebeskrivelse

Dette prosjektet skal gå nærmere inn på rådata fra sensorer festet til skistavene. Ut fra rådataene som kommer fra sensorene skal det utledes en analyse av staketeknikken til bruker. Data som skal utledes er:

- Antall staketak
- Vinkel på skistaven i ulike faser av stavtaket
- Akselerasjon på skistavene i ulke faser av staketaket
- Akselerasjon på skiløper
- Frekvensen på staketakene
- Hastighet på skiløper

Rådataene skal strømmes fra sensorene som er festet til skistavene, til mobiltelefonen. På mobiltelefonen skal det lages en applikasjon som tar imot rådataene fra sensorene og utleder informasjonen nevnt ovenfor, deretter skal applikasjonen kunne vise frem de utledede data til bruker på en oversiktlig måte.

3 Planlegging, oppfølging og rapportering

3.1 Ansvarsforhold og rutiner

Jeg er alene på dette prosjektet og er derfor ansvarlig for å levere et godt produkt til oppdragsgiver, samt skrive en god bacheloroppgave for min egen del.

Uansett om jeg er alene på gruppen, er det greit å lage noen rutiner for meg selv. Jeg har kommet frem til følgende:

- Arbeidstiden er fra 08:00 til 16:00 mandag til fredag.
- Det skal ikke pushes opp kode som ikke kompilerer til repositoriet.
- Alt som pushes opp til repository, skal ha en passende kort melding om hva som er blitt gjort.

3.2 Valg av programmeringsspråk

Som det er kommet frem tidligere, skal jeg utvikle applikasjon til android, og ikke ta hensyn til at applikasjonen skal fungere på tvers av platformer. Android har sine offesielle språk, dette er Java og Kotlin, det at de er offesielle vil si at google, som eier android støtter disse språkene. Det finnes også andre programmeringsspråk som fungerer på android, noen av disse er C/C++, C#, BASIC og HTML.

Jeg kunne her ha valgt HTML som språk og oppfylt oppdragsgiver sitt ønske om at applikasjonen skal kunne brukes på tvers av platformer, dette ville også medført at applikasjonen hadde vært lettere å vedlikeholde, siden man bare trenger å oppdatere en kode som går på tvers av flere platformer, istedet for å oppdatere en kode per platform man skal ha applikasjonen.

Android benytter en virtuell maskin for å kjøre mye av koden sin, C/C++ bruker ikke denne virtuelle maskinen, og klarer derfor å utnytte maskinvaren litt mer enn de andre språkene.

Jeg har valgt å bruke Java for dette prosjektet. Grunnen til at jeg velger Java er fordi java er det mest utbredte programmeringsspråket for android, jeg har litt erfaring med java fra før av og jeg skal jobbe mot et API for sensorene, hvorav det er godt dokumentert med eksempler for java. Det er også en fordell for oppdragsgiver at jeg bruker java, siden oppdragsgiver har planer om å videreutvikle prosjektet, og implementere server med nettopp java, hvorav prossesering av rådata skal flyttes over til serveren.

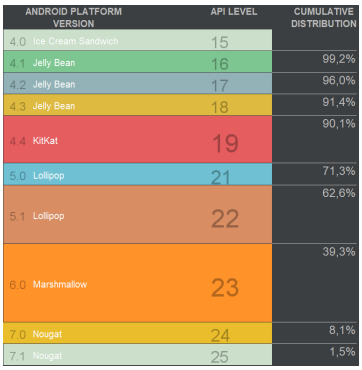
Den eneste grunnen til at jeg skulle ha valgt en løsning i HTML, er at det skulle fungert på kryss av platformer, noe som jeg ikke prioriterer. Å kode i HTML vil også påvirke applikasjonens ytelse, dette vil føre til at brukere blir irritert, og i værste fall vil det ikke være en god løsning for kryss platform heller siden Apple stiller strenge krav til ytelse for applikasjoner som kommer i Apple store.

C/C++ har også sine negative sider, men det som virkelig trekker denne valgmuligheten ned er at det legger på mye kompleksitet til prosjektet, som skaper større risiko for å bli hengende etter tids skjemaet. Det er også diskuterbart om det er noen ytelses forbedringer å ikke benytte en virtuell maskin.

3.3 Android versjon

Da man utvikler til android, må man velge hvilke versjon man utvikler til. Alle har ikke mobiltelefon med den nyeste versjonen av android, men programvare til de eldre versjonene, vil fungere på de nyere.

Hver versjon av android introduserer nye funksjoner man kan ta nytte av i sin programvare, derfor er det viktig å velge riktig versjon, slik at man har tilgang til de funksjonene man trenger i applikasjonen, men samtidig at flest mulig brukere har tilgang til applikasjonen. I figur 1 fremvises den kumulative distribusjonen av de forskjellige android versjonene



Figur 1: Android versjoner

Et krav som stilles av sensorene som benyttes er Bluetooth Low Energy (BLE). BLE teknologien introduseres først i versjon 4.3 Jelly Bean, så jeg er derfor nødt til å velge denne versjonen, eller nyere. Hoppet fra versjon 4.4, 5.0 og videre ser man at distribusjonen virkelig begynner å falle, da kommer valget til hvilke funksjoner som vil være av nytte i dette prosjektet.

Funksjoner som vil være av nytte i dette prosjektet er BLE og sensor relaterte funksjoner. Jeg velger å utvikle til versjon 4.4 KitKat, her introduseres tilgang en rekke sensor funksjoner som kan komme til nytte i dette prosjektet, og nedfallet i distribusjonen er ganske lav fra miste kravet som er versjon 4.3 Jelly Bean. Fra og med versjon 5.0 introduseres det ikke flere funksjoner som er av spesiell interesse for dette prosjektet.

3.4 Hovedinndeling av prosjektet

3.4.1 Valg av SU-modell

Valg av systemutviklingsmodell vil velges ut fra følgende kriterier:

- Krav til programvaren.
- Kompleksiteten til løsningen.
- Tilbakemelding på arbeid som er blitt gjort.
- Hyppige endringer til kravene og/eller løsningen til prosjektet.
- Kostnaden for forsinkelser.
- Min erfaring på prosjekter.

Jeg velger å bruke en agil fremfor en plandrevet systemutviklingsmodell, dette fordi det ikke er noen klar løsning på kravene, dette vil si at det finnes flere

måter å løse utledningen til de forskjellige målingene som ønskes. Så det er muligheter for at jeg må gå tilbake for å eksempel endre på en algoritme som er mer gunstig i en agil modell.

Under de agile modellene er det mangt å velge mellom. Xtreme programming er en, men denne metoden baserer seg mye på par-programmering, noe som blir vanskelig når jeg er én person, derfor utelukker jeg denne metoden.

Oppgaven har klare krav om hva som ønskes gjort, og det er lite sannsynlighet for endring i kravene samt veies tilbakemeldinger på produktet underveis både fra oppdragsgiver og veileder høyt.

En variasjon av Scrum eller kanban er to mulige kandidater, scrum vektlegger mer på å holde et tidsskjema enn kanban, som ikke har noen tids avgrensning på oppgavene.

Jeg velger å anvende en variasjon av scrum grunnet at sprinter passer prosjektet med tanke på at det er behov på tilbakemeldinger på arbeidet som er blitt gjort, og fordi konstnaden for forsikelser er stor, derfor vil tidsskjema aspektet av scrum være til nytte. Det er også en bonus at jeg tidligere har erfaring med scrum.

Sammen med scrum velger jeg også å bruke test-drevet utvikling (TDD) for å ikke introdusere nye feil i applikasjonen etter den er blitt oppdatert.

3.4.2 Anvendelse av SU-modell

I dette prosjektet vil jeg anvende scrum på en ganske tradisjonell måte, men noen endringer må gjøres. Siden jeg er én person på prosjektet vil jeg ta rollen som både ScrumMaster og utvikler. Oppdragsgiver vil ta rollen som product owner. Jeg velger også å ha korte sprint perioder på to uker, altså 10 arbeidsdager.

Jeg vil måle fremgang i sprinten og vurdere hva jeg må jobbe på hver dag, som erstatning for daglige scrum møter.

Produkt backlog vil bli satt opp i starten av prosjekt perioden, hvor jeg skal se om jeg kan benytte veileder sin hjelp til å få et bedre estimat på oppgavene. produkt backlog vil ha en arbeidsmenge på rundt 80 arbeidtimer, men det vil bli tatt unntak hvis det viser seg å bli knapt med tid.

Sprint backlog kommer jeg til å styre litt selv, men jeg vil prøve å få oppdragsgiver sin prioritet på de forskjellige oppgavene.

Test-drevet utvikling vil bli benyttet ved at jeg lager tester for applikasjonen først, dermed lager jeg selve applikasjonen til å passe inn i testene.

3.5 Valg av verktøy

Til å programmere velger jeg å bruke Android Studio, som er utviklet av google spesifikt for å utvikle til android platformen. Android Studio kommer med alle verktøyene man trenger for å utvikle i java til android, som for eksempel er det innebygd virtuell android platform, slik at man kan lett teste programmet mens man programmerer.

Latex er valget for å skrive bacheloroppgave, jeg bruker latex fordi man kan få til alt man vil, og jeg er ganske vandt med latex.

Det vil bli brukt et bitbucket repository til å lagre koden, jeg vil bruke dette fordi det minsker risiko for tap av data, samt gir det oppdragsgiver muligheten til å se koden under prosjektet.

Jeg vil benytte meg av Trello for å holde styr på produkt backlog og sprint backlog.

3.6 Plan for statusmøter og beslutningspunkter i perioden

Det er satt opp ukentlige møter med veileder hver onsdag kl. 09:00, flere møter kan arrangeres ved behov. Møter med oppdragsgiver er ikke bestemt, men tas når det er behov, og passer for begge parter.

Torsdag 01.02.2018 vil innholdet til den første sprinten bestemmes, og hver andre uke etter dette.

4 Organisering av kvalitetssikring

4.1 Dokumentasjon, standardbruk og kildekode

Jeg vil bruke JavaDoc og kommentarer for å dokumentere kilde koden. Grunnen til at jeg velger JavaDoc er fordi det støttes i Android studio, de fleste populære java verktøy som Eclipse og IntelliJ, det gir lett tilgang til dokumentasjonen du trenger der og da, samt en god oversikt over hele programmet.

JavaDoc og kommentering av koden skal skje mens koden blir skrevet, slik at det ikke samler seg opp mye udokumentert kode, som jeg nødvendigvis ikke husker hva gjør i ettertid.

4.2 Konfigurasjonsstyring

Konfigurasjonsstyring omhandler styringen av innhold, endringer eller status på delt informasjon i et prosjekt. I dette prosjektet er det bare jeg som skal gjøre endringer og oppdateringer av all kilde kode og rapport skriving, derfor er konfigurasjonsstyring unødvendig for min del, men som tidligere nevnt, vil jeg benytte meg av bitbucket for kilde koden som er en form for konfigurasjonsstyring.

4.3 Risikoanalyse

Risiko blir vurdert av et produkt av sansynlighet og konsekvens. I Tabell 1 har jeg laget en oversikt over problemer jeg anser som en risiko.

Beskrivelse	**Sansynlighet (1-10)**	**Konsekvens(1-10)**	**Risikovurdering**
Langvarig sykdom	4	3	12%
Mangel av kompetanse	7	3	21%
Dårlig tid estimater	6	5	30%
Tap av data/utstyr	2	8	16%
Endringer av krav	1	6	6%
For dårlig utstyr	2	9	18%

Tabell 1: Risikovurdering

Et prosjekt som dette vil det alltid følge med en risiko, men det er mulig å gjøre noen tiltak for å redusere enten sansynligheten eller konsekvensen for at problemet inntreffer. Mangel av kompetanse/kunnskap er noe som antageligvis vil skje, men jeg vil ha tilgang til hjelp, enten om det er fra internett, veileder, og forhåpentligvis forelesere på NTNU som kan hjelpe meg hvis jeg skulle stå fast.

Tap av data kan inntreffe, og her vil jeg bruke et repository på bitbucket slik at koden til programmet blir lagret på to steder og jeg vil ha en backup, jeg må også laste opp endringer mens jeg gjør dem slik at ikke noe kode blir tapt.

Når det gjelder dårlig tids estimater på oppgaver, er dette noe som fort kan skje, siden jeg har lite erfaring med prosjekter, men som sagt tidligere vil jeg benytte veileder til hjelp for å estimere oppgaver.

Hvis det skulle forekomme endringer av krav fra oppdragsgiver kan det være katastrofalt, eller det kan gå bra. Hvis endrigene av kravene kommer sent i prosjekt perioden vil jeg antageligvis ikke ha tid til å endre det, derimot hvis endringene kommer tidlig, vil det kunne gå problemfritt.

Det kan forekomme at sensorene ikke er bra nok til å levere stabil data, men dette anses som svært usannsynlig, og hvis det skulle skje vil jeg bare måtte simulere data eller jobbe med det jeg har.

5 Plan for gjennomføring

5.1 Liste over aktiviteter

Oppgaven kan deles opp i følgende hoved oppgaver og under oppgaver:

1. Lage applikasjon til android
   (a) Lage GUI
   (b) Implementere funksjoner til GUI
   (c) Gjøre klart til visualisering av data
   (d) Opprette strømming over BLE med sensorer
   (e) Testing av applikasjonen
2. Prosessere data
   (a) Finne metode for utledning av gitt data
   (b) Implementere metoden i android applikasjonen
   (c) Teste metoden

Hvorav oppgavene under prosessering av data gjentas for hver type data som skal utledes.

5.2 Milepæler og beslutningspunkter

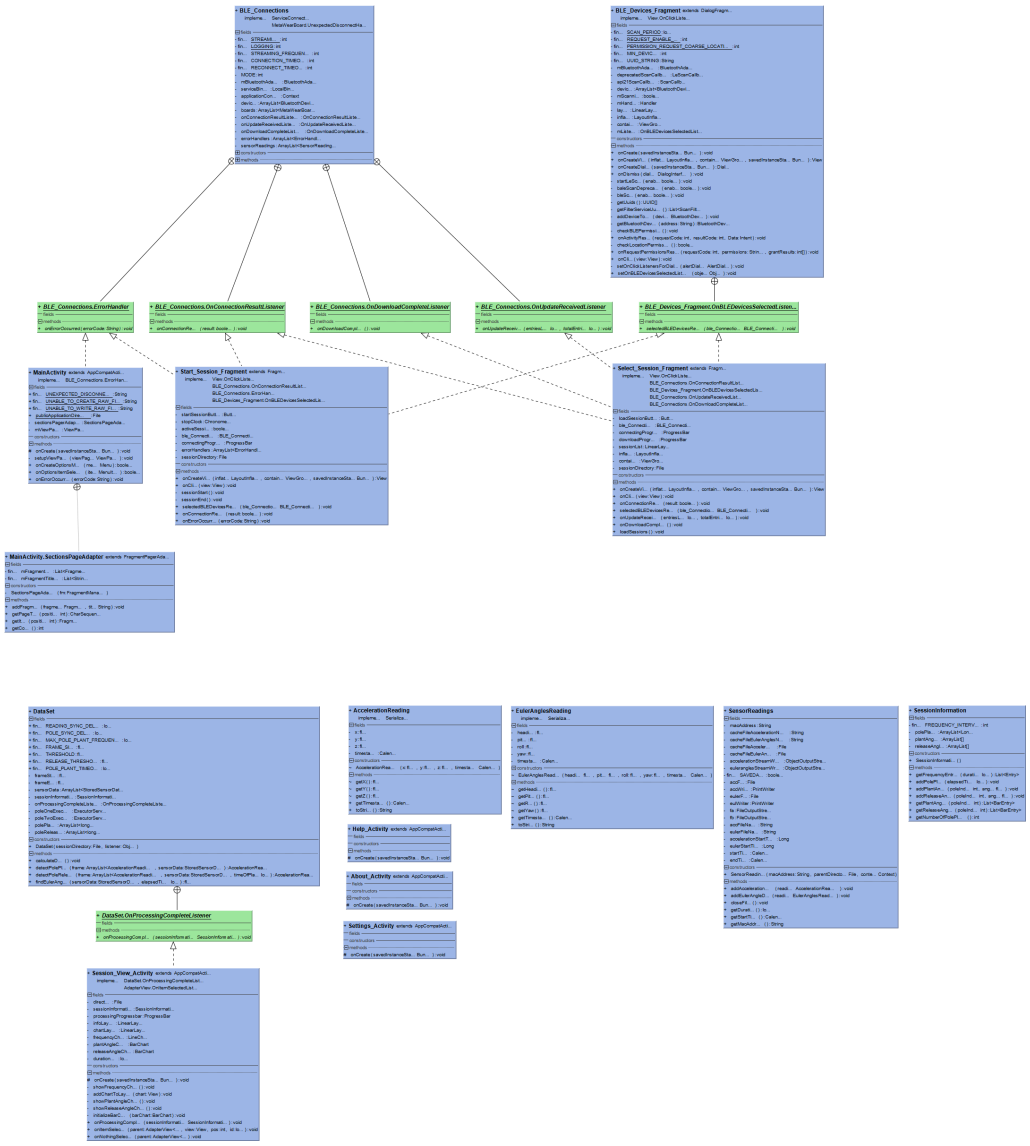
Prosjektet gir mye frihet til å velge i hvilken ende jeg vil starte, siden få av oppgavene er avhengig av hverandre. Jeg velger etter samtale med oppdragsgiver å starte med å utvikle mobilapplikasjonen. Jeg starter med å utvikle mobilapplikasjonen grunnet at jeg da får lettere tilgang til å teste analysen av rådata.

- Milepæl 1
  - Utviklet android applikasjon som kan/har:
    * strømme rådata fra sensorer
    * fremvise behandlet data
    * brukervennlig GUI
- Milepæl 2
  - Utledet ønsket data fra rådata.
  - Implementert metode for å utlede data i android applikasjonen

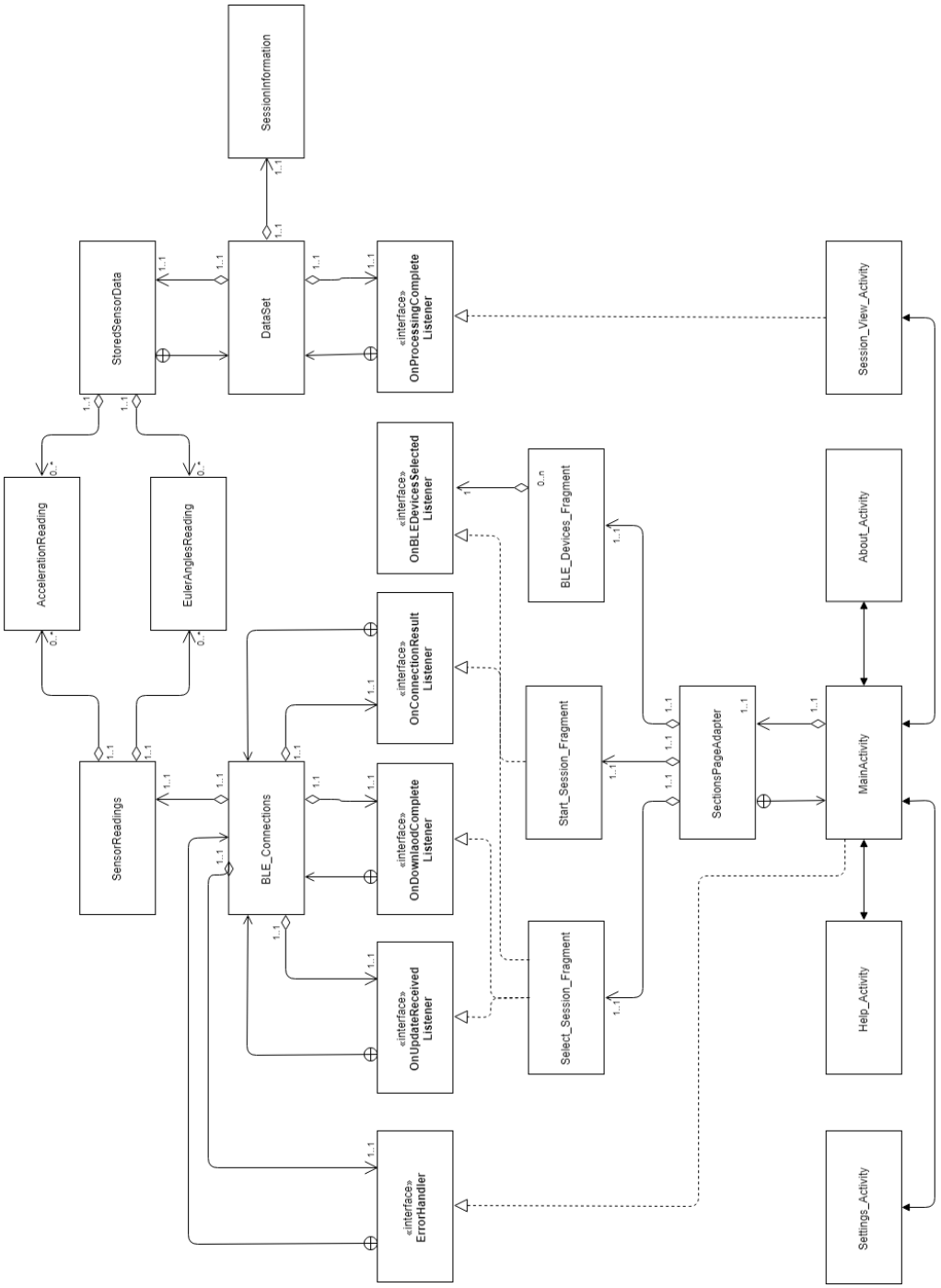
5.3 Tids- og ressursplan

Figur 2: Gantt-skjema

C Klassediagram



Figur 17: Fullstendig klassediagram



Figur 18: Fullstendig relasjonsdiagram

D Kilde kode

Vedlagt ligger en clone av Bitbucket repositoriet, som inneholder alle filer involvert i kodingen av prosjektet, samt tilhørende JavaDoc.