Høgskulen på Vestlandet

BACHELOROPPGAVE:

BO21E-11 TEST AV FALLSENSOR

Sivert Elias Åmelfot

Revisjon 4

31.05.2021

Dokumentkontroll

Rapportens tittel: BO21E-11 Test av fallsensor	*Dato/Versjon* 31.05.2021/ R4
	Rapportnummer: B021E-11
Forfatter(e): Sivert Elias Åmelfot	*Studieretning:* HEAU18
	Antall sider m/vedlegg 60
Høgskolens veileder: Marcus Landschulze	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Bergen Kommune	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): *Øritsland, Morten Torger <Morten.Oritsland@bergen.kommune.no>*	

Revisjon	Dato	Status
1	31.01.21	Første utkast, forarbeid
2	04.05.21	Andre utkast. Hovuddel av skriving.
3	24.05.21	Utført endringer etter råd frå veileder.
4	28.05.21	La til kapittel om brukargrensesnitt for MATLAB-app. Omformulere setninger og flytte avsnitt. Rette skrivefeil.

Samandrag

Denne oppgåva har som mål å teste fallsensoren Vibby OAK frå Tunstall. Fallsensoren kan brukast anten som klokke eller halskjede. Dersom ein dett i golvet skal fallsensoren registrere fallet og varsle nærkontaktar eller andre hjelpeordninger. Responssenteret i Bergen kommune er ansvarlege for å følgje opp slike alarmer. Dei opplever at fallsensoren Vibby OAK ikkje detekterer fall på ein påliteleg måte, og ønsker derfor ei utgreiing om kva grenser den blir løyst ut på. Dette er slik at dei kan bestemme brukargruppa som har størst nytte av fallsensoren.

For å teste dette vart ein fallrigg for simulering av fall konstruert, samt laga programvare for datainnsamling og visualisering. Det vart nytta eigne sensorar for datainnsamling då det ikkje vart gitt løyve til å opne fallsensoren.
Grunna ei uventa konseptuell svakheit i konstruksjonen har fallriggen ein oscillasjon som gjer det vanskeleg å løyse ut fallalarmen samt samanlikne resultat frå denne oppgåva med ekte fall. Det gjer det også umogleg å bestemme kor konsistent sensoren er og finne eventuelle grenser for fall. Desse problema blir gjennomgått i kapittel 5.3 og 6.1.

Dersom Responssenteret opplever at fallsensoren i klokkemodus ikkje er påliteleg, kan det vere hensiktsmessig å bytte til halskjedemodus. Forsking viser at å plassere ein fallsensor nære kroppens massesenter (frå hofte til brystkasse) gir meir pålitelege resultat enn på handleddet, sjå kapittel 6.2.

Innhaldsliste

Figurar, teikningar og tabellar

1 Innleiing

1.1 Oppdragsgivar

Bergen kommune byrådsavdeling for eldre, helse og frivillighet, seksjon for e-helse er oppdragsgivar.

«E-helseseksjonen har ansvar for alt som har med digitalisering og utvikling innen e-helseområdet, og samarbeider tett med Digitalisering og innovasjon konsern. Seksjonen har ansvar for e-helseporteføljen, herunder anskaffelser, utviklingsprosjekter, informasjonssikkerhet og drift av IKT-systemer i BEHF. Bergen kommune er involvert i flere nasjonale e-helse-prosjekter, og regionalt arbeid med Akson (En innbygger en journal) og e-komp ligger til denne seksjonen. I tillegg inngår arbeid med prosjektet Folkekommune i E-helseseksjonen.» [1]

Responssenteret er dei som handterer og følger opp alarmane i heimetenesta. Det er difor dei som har erfaring med sjølve fallsensorane.

«Responssenteret er et døgnbemannet alarmmottak som følger opp alle utløste alarmer, for eksempel trygghetsalarm med sensorer, GPS, medisineringsstøtte eller kamera. Senteret er bemannet med helsepersonell, og er et kompetansesenter for bruk av velferdsteknologi. Responssenteret har ansvar for opplæring, veiledning og informasjon til ansatte, brukere, pårørende og innbyggere. Målet til responssenteret er å styrke tjenestetilbudet i hjemmetjenesten, med fokus på økt trygghet, sikkerhet og selvstendighet for Bergens innbyggere.» [2]

1.2 Problemstilling

Bergen kommune har kjøpt inn fallsensoren Vibby OAK frå Tunstall. Responssenteret opplever at fallsensoren ikkje detekterer fall på ein påliteleg måte, dei har bekrefta dette ved å ringe til brukaren eller pårørande og spurt om talet fall i forhold til det som er blitt registrert. Responssenteret ønsker derfor å teste pålitelegheita til fallsensoren og finne grenser for deteksjon av fall. Dette er for å finne ut kva brukargruppe denne sensoren kan fungere best for med tanke på fysisk og mental/kognitiv evne og kapasitet.

Bergen kommune ga ikkje løyve til å opne fallsensoren, og Tunstall ønska ikkje å dele teknisk dokumentasjon av konfidensielt omsyn. Det betyr at oppgåva må ta i bruk eksterne sensorar for å vurdere pålitelegheit og finne grenser for deteksjon.

2 Kravspesifikasjon

- Finne grenser for deteksjon av fall, dvs. fallhøgde og fart.
    - Måle akselerasjon og retardasjon for å få eit fullstendig bilete.
    - Fart blir rekna ut frå akselerasjonen.
- Knytte desse fysiske størrelsane til noko som er relaterbart for oppdragsgivar, slik at dei kan vurdere om produktet er godt nok eigna for deira bruksområde.
- Vurdere om fallsensoren kan detektere fall med fleire steg.

3 Analyse av problemet

3.1 Kort om fallsensoren

Fallsensoren Vibby OAK er designa og laga av Vitalbase. Tunstall implementerer deretter sine eigne kommunikasjonsprotokoller for integrering i deira økosystem av e-helse-hjelpemiddel. Fallsensoren kan brukast anten som klokke eller halskjede. For å detektere fall består fallsensoren av eit akselerometer og eit barometer / lufttrykksensor. For at eit fall skal bli registrert, må fallsensoren oppleve ein hurtig reduksjon i høgde, ein brå deakselerasjon (t.d. samanstøyt mot golvet) og deretter ligge relativt stille. Dersom den detekterer eit fall, vil den begynne å vibrere samt sende eit signal til ein heimesentral, som så igjen ringjer til pårørande eller responssenteret slik at brukaren kan få hjelp.

Figur 1: Fallsensoren Vibby OAK

3.2 Utforming av moglege løysingar

For å finne grensene for deteksjon av fall er det ønskeleg å bruke dei same sensorane fallsensoren inneheld, slik at ein kan lese av dei same verdiane fallsensoren måler. Dette gjer det lettare å vurdere korleis fallsensoren opplever eit fall og oppdage eventuelle feil i eige testoppsett.
Akselerometer kan måle vibrasjon, dette er nyttig sidan fallsensoren begynner å vibrere dersom den detekterer eit fall. Dette kan ein bruke for å automatisk detektere om fallsensoren detekterte eit fall.

Ettersom fallsensoren har to ulike modus, klokke og halskjede, må begge testast. I produktdatabladet [3] er det oppgitt at i klokkemodus må fallsensoren vere minst 60cm over bakken, medan i halskjedemodus må den vere minst 1m over bakken. Det stiller krav til at testriggen må kunne justerast i høgde. For å systematisk teste ulike parameter, er det viktig å ha eit system som kan simulere fall på ein svært kontrollert og repeterbar måte.
Det ideelle er å simulere eit fall som er så likt eit ekte fall som mogleg, men det krev kunnskap som er utanfor omfanget til denne oppgåva.

3.2.1 Maskinvare

Uavhengig av maskin eller konstruksjon som skal stå for sjølve simuleringa av fallet, må det skaffast sensorar som kan måle dei ønska fysiske verdiane. I tillegg trengs ein mikrokontroller som kan samle inn denne dataa og overføre dei til ein PC.
Det er planlagt å bruke ein Arduino Uno Rev.3, sjå 3.2.5 for grunngjeving. Sensorane må kjøpast inn.

Ettersom eit fall skjer over kort tid, er det ønskeleg at sensorane har ein høg samplingsrate og har eit måleområde som omfattar alle moglege verdiar før, under og etter fallet. Då vil ein få eit så fullstendig bilete som mogleg over korleis dei målte størrelsane varierer med tida. Akselerometer med ein høg samplingsrate vil gjere det mogleg å detektere høgfrekvente mekaniske vibrasjonar og fjerne desse før visualisering om ønskeleg.
God presisjon og lav støy er viktig, slik at desse to ikkje degraderer signalet nemneverdig og gjer

eventuell signalbehandling vanskeleg. I tillegg er god nøyaktigheit viktig for akselerometeret, då dei målte verdiane for akselerasjon direkte vil påverke utrekna verdiar for hastigheit. Hos barometeret er det den relative differansen i høgda før og etter fall som er interessant, då er presisjon / relativ nøyaktigheit viktigare enn (absolutt) nøyaktigheit.

Det er ikkje utført grundige studiar over nødvendige spesifikasjonar for sensorane, då dette vil ta unødvendig lang tid. Sensorane er i staden vurdert ut frå om dei er gode i forhold til andre tilgjengelege sensorar. Det er stilt krav til at sensorane må vere på eit sensorkort med eigen straum – og spenningsregulering og ha anten I^2C eller SPI datatilkopling. I^2C er å foretrekke då det er enklast i bruk.

Etter eit kort søk vart følgande sensorkort funne og bestilt for innkjøp:

- SparkFun SEN-15335
    - InvenSense ICM 20948 MEMS IMU
    - Akselerometer med 3 aksar, 1000Hz samplingsrate, måleområde ±16g
- Adafruit BMP390
    - Bosch BMP390L lufttrykksensor
    - 200Hz samplingsrate, nøyaktigheit ±50 Pa, presisjon ±3 Pa (tilsvarande ±0.25m)

Det blei også kjøpt inn kablar for å spare tid ved oppsett. Sjå Appendiks C for meir detaljert informasjon om kvar sensor, referansar til datablad og koplingsskjema.

Ved å velje ein Inertial Measurement Unit (IMU) står moglegheitene opne for å måle andre fysiske størrelsar enn lineær akselerasjon (som angulær hastigheit / rotasjonsrate), om ein seinare ser at dette er ønskeleg.

For å ha ein praktisk plattform å jobbe med vurderast det å 3D-printe ein festebrakett med skruehól for å feste Arduino, akselerometer, barometer og fallsensor. Festebraketten må tilpassast maskina eller konstruksjonen som skal stå for simuleringa av fallet.

3.2.2 Programvare

Det må utviklast eit program til den valde mikrokontrolleren for innsamling og overføring av data. Utforming er avhengig av produsentens bibliotek til sensoren. Det er planlagt å ikkje utføre signalbehandling på mikrokontrolleren, då det krev kostbar prosesseringstid. Det kan vere nyttig å ha lagra rådata om ein i framtida ønsker å t.d. sjå på kvar av aksane i akselerometeret eller endre signalbehandling.

Det må utviklast eit program til PC for å motta, lagre og visualisere data frå mikrokontrolleren. Det må støtte innføring av parameter som modus for fallsensor, fallhøgde, talet steg i fallet, testtid, evt. demping og ønska filsti og/eller – namn der fila skal lagrast. Det er planlagt å lagre data i Excel, Visual Studio har Office Developer Tools slik at ein kan programmatisk legge inn data i Excel gjennom eit anna program.

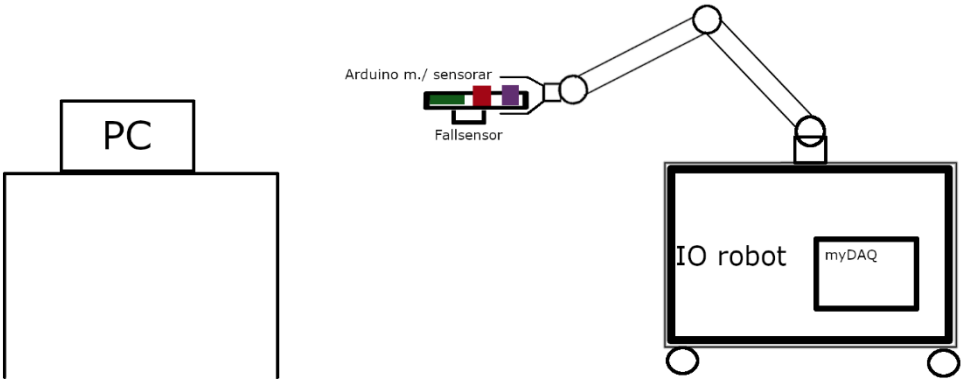
3.2.3 Løysingsalternativ 1: Robotarm

Ved å bruke ein robotarm kan det settast opp kontrollert testing av fallsensoren der ein kan justere parameter som fallhøgde, hastigheit og akselerasjon. Skulen har bl.a. robotarma Universal Robots UR5e. Denne roboten har digitale inngangar og utgangar, som gjer kommunikasjon med eksterne einingar mogleg.

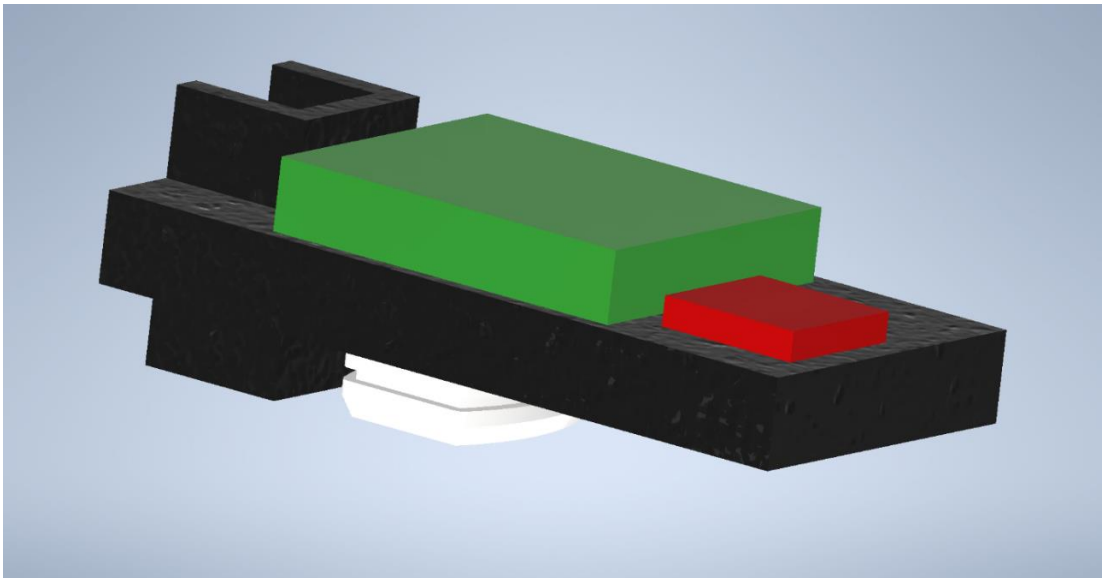
Løysinga nytter beskrivinga av maskinvare og programvare nemnt i dei to førre delkapitla, men dataprogrammet skal i tillegg automatisere falltesting. Programmet skal sende ønska verdiar på parameter (fallhøgde, hastigheit, akselerasjon) til Arduino, som vidaresender det til roboten.

Det er uvisst om det let seg gjere å nytte Arduino til å styre robotarmen, då det må strekkast lange kablar langs armen. I så tilfelle kan National Instruments myDAQ (Data Aquisition device) brukast, den kan plasserast rett ved IO (Input-Output) hos robotarm og koplast til PC med ein (lang) USB-kabel. Sjå Figur 2.

Dette gjer det mogleg å fullstendig automatisere testing av fallsensoren i éin modus, som gjer det lettare og raskare å utføre mange målingar med sett av ulike parameter.



Figur 2: Oversikt over testoppsett



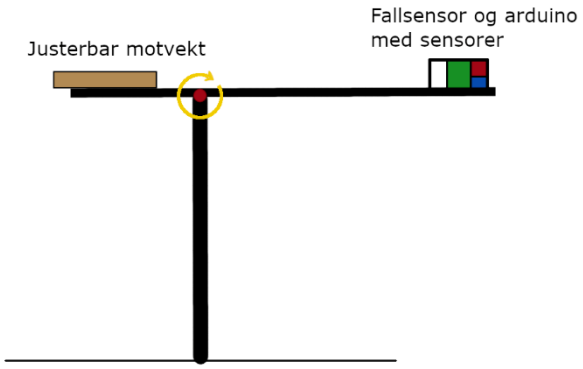
Figur 3: Konsept, brakett for Arduino og sensorar

3.2.4 Løysingsalternativ 2: Fallrigg

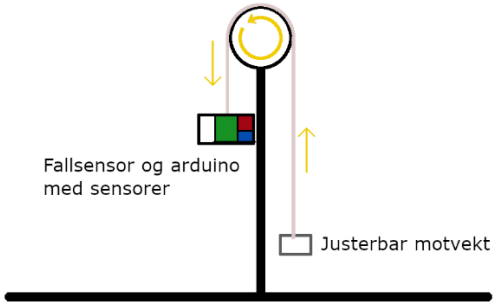
Dette løysingsalternativet er lagt til retrospektivt. Forslaget nytter beskrivinga av maskinvare og programvare nemnt tidlegare i kapittelet.

To enkle prinsipp vart vurderte: arm med motvekt og snor over trinse. Begge konsepta lar ein justere fallhøgde, ønska akselerasjon under fallet og eventuelt demping. Desse løysingane er lettare å implementere mtp. programvare, då dei ikkje treng å ta omsyn til enda ei ekstern eining (robotarm) som må programmerast.

Akselerasjon under fallet kan justerast ved å flytte eller auke massen til motvekta. Fallhøgda kan justerast ved å ha ein utløysarmekanisme som er låst til ei høgde. Dette gjer i tillegg at ein ikkje påfører forstyrringar til systemet like før eit fall.



Figur 4: Konsept - Arm med motvekt



Figur 5: Konsept - Trinse med motvekt

3.2.5 Vurderingar i forhold til verktøy og HW/SW komponentar

I utgangspunktet vart Microsoft Excel nytta til å loggføre data då det er utbreidd og lett tilgjengeleg. Excel kan programmatisk manipulerast gjennom Visual Studio (VS) Office Developer Tools / Office Interoperability. Microsoft og tredjepartar har tilgjengeleg gratis opplæringsmateriale på nettet. Signalbehandling og visualisering kan utførast i MATLAB.

Det vart bestemt å bruke Arduino Uno Rev.3 då ein allereie har den for hand og at den er enkel i bruk mtp. dataoverføring over USB. Det er ein populær mikrokontroller som har brei støtte for tredjeparts komponentar/sensorkort med tilhøyrande programvare.

3.3 Endringar og konkusjon

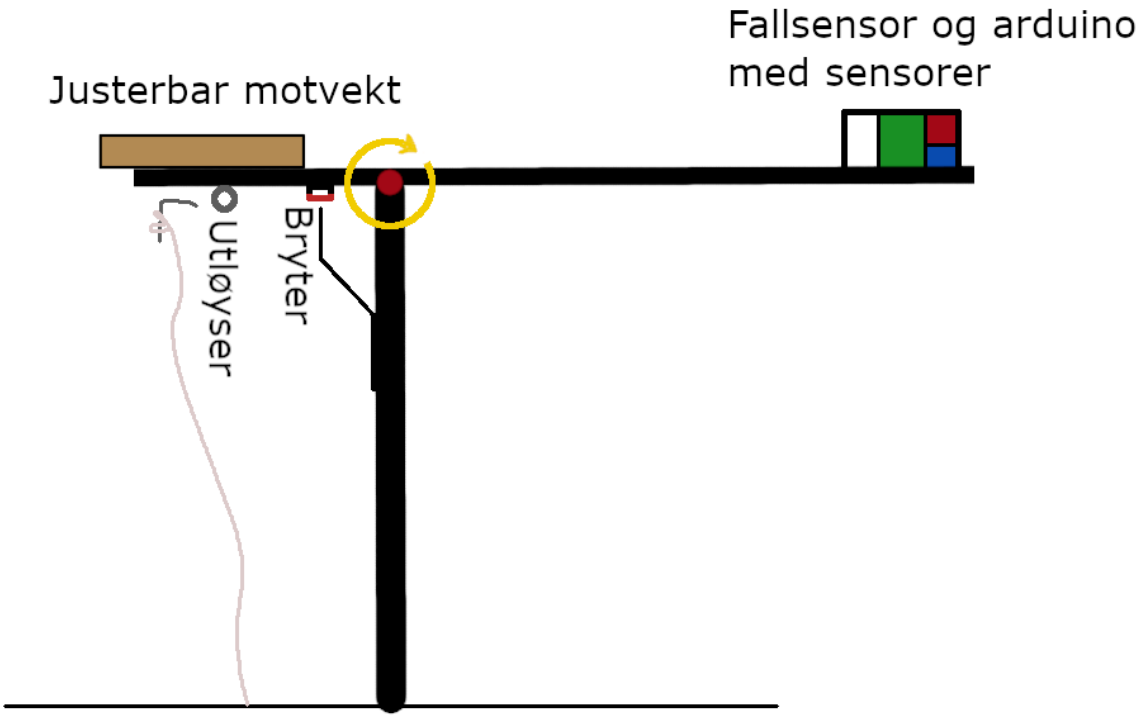
Etter tidlege testar med robotarmen, viste det seg at den ikkje klarte å løyse ut fallsensoren, då akselerasjon/deakselerasjon var for lav. Den originale framdriftsplanen vart dermed forkasta, og det måtte utviklast ei ny løysing for simulering av fall, greidd ut i kap. 3.2.4.

Det verka som at det var enklare å justere fallhøgda for arma, då arma som held trinsa må vere høgare enn oppsettet for arm-konseptet for å levere same justeringsmoglegheiter. I tillegg verka det lettare å konstruere ein mekanisme for deteksjon av start av fallet på armen. Difor vart arm med motvekt vald som forslaget oppgåva gjekk vidare med. I tillegg viste det seg også at det vart tungvint å nytte Excel til å lagre data, sidan ein må forhalde seg til celler ved programmering. I staden vart MATLAB nytta til både lagring, signalbehandling og visualisering av data.

4 Realisering av valt løysing

4.1 Fallrigg

Kapittelet går gjennom den praktiske implementasjonen av løysingsforslaget greidd ut i kap. 3.2.4. Grunna tidsavgrensing samt planlegging og godkjenning av budsjett vart det brukt komponentar funne i metall- og trevirke-container på skulen.

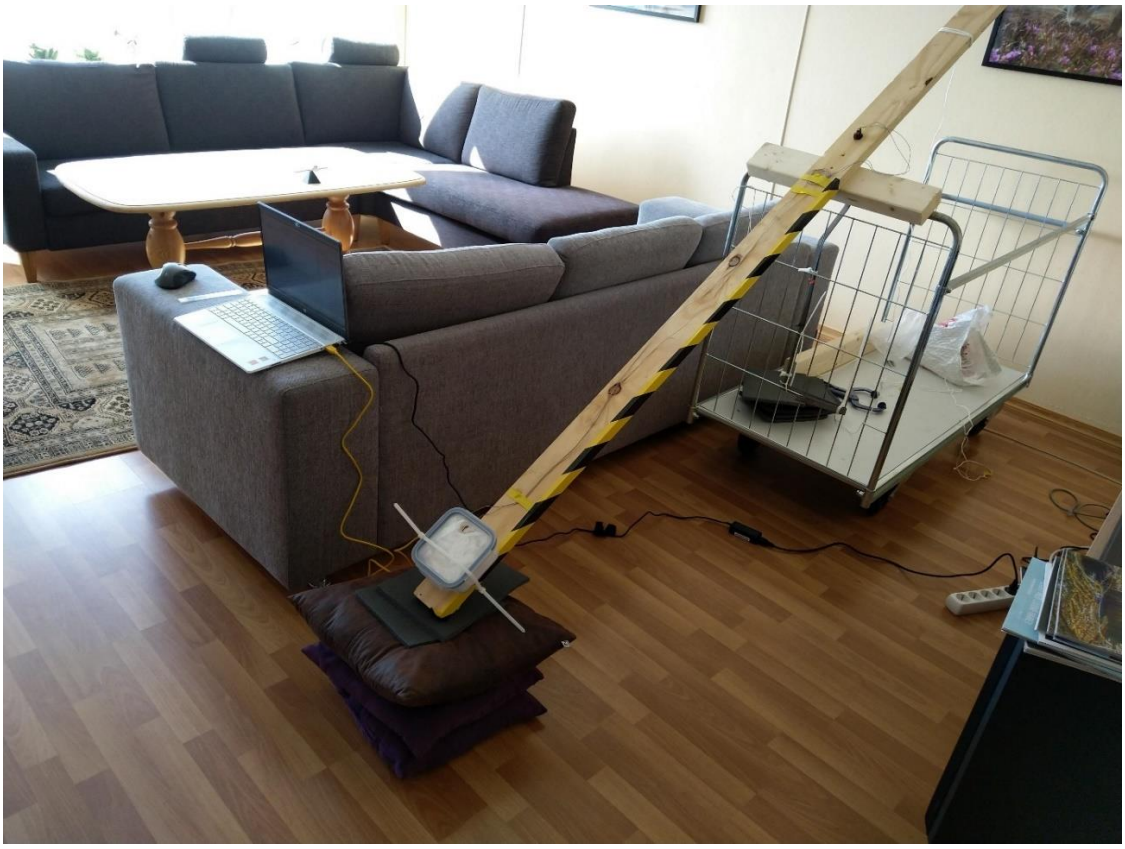


Figur 6: Prinsippskisse fallrigg

For å starte eit fall har fallriggen ein utløysarmekanisme: Ein tråd med krok i enden er festa i ein augeskrue ved basen til fallriggen. Kroken kan hukast i ein annan augeskrue i armen på fallriggen, og kan enkelt dragast ut sidelengs utan å påverke resten av fallriggen i nemneverdig grad.
For å enkelt kunne detektere når eit fall startar (uavhengig av kor tid fallet skjer etter ein startar logging i program på PC) vart det installert ein trykknapp/bryter på armen. Denne knappen blir trykt inn av ei metallstong når armen er på sitt høgste punkt (før fall). Ved fall løftast knappen av stanga, og brytaren opnast. Sjå evt. koplingsskjema på Figur 20.

Oversiktsbilete under viser fallriggen samt knappen og den justerbare motvekta.

Ei tralle vart vald som base for fallriggen, då den gjer det mogleg å trille rundt, plassere diverse utstyr på, hadde ei ramme med passeleg høgde og kan feste utløysarmekanismen i botnen. Armen vart festa til tralla med patentband. Det vart bora eit hól i armen for å få brytaren til å henge fast. Leidningane vart lodda på brytaren i eine enden og til pin-kontaktar som passar i koplingsbrettet til Arduino i andre enden.



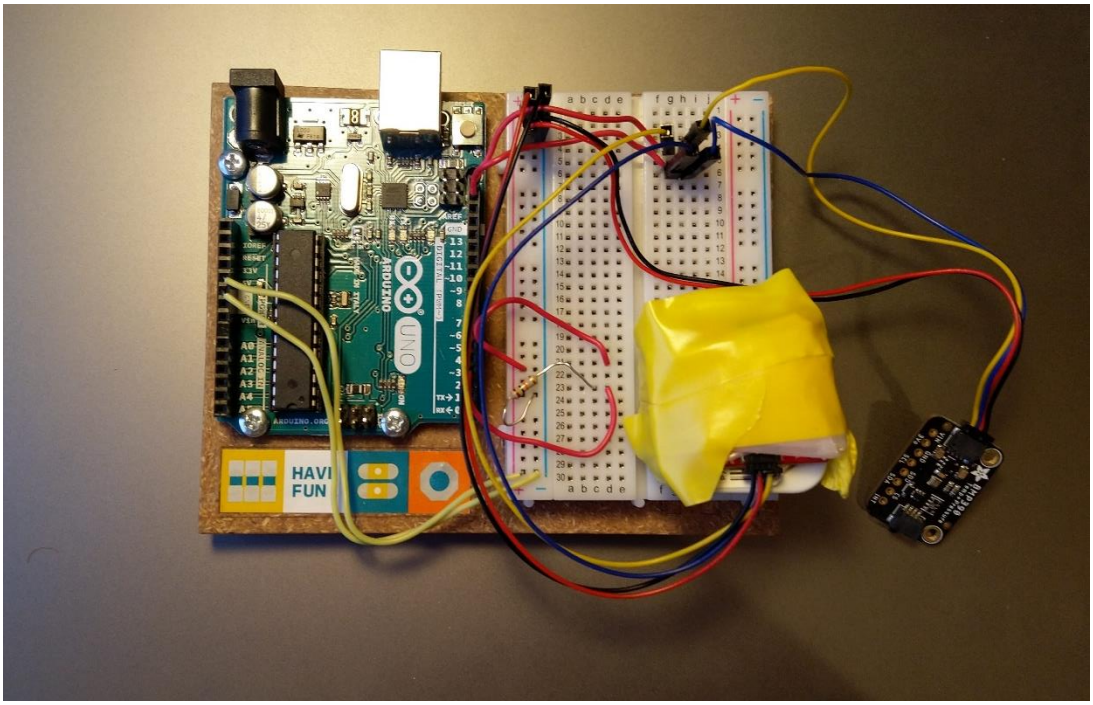
Figur 7: Praktisk implementasjon av fallrigg

Figur 8: Knapp og stang for deteksjon av starten av eit fall

Figur 9: Justerbar motvekt lar ein teste fallsensoren med ulik akselerasjon under fallet

4.2 Maskinvare

Løysingsforslaget greidd ut i kapittel 3.2.1 er behaldt, med unntak av 3D-printa festebrakett. Figuren under viser korleis sensorane og leidningar til knappen er kopla opp på Arduinoens koplingsbrett. For implementasjonsdetaljar med bl.a. koplingsskjema, sjå Appendiks C . Akselerometeret er teipa fast til fallsensoren, slik at det er lett å detektere vibrasjon dersom fallsensoren detekterer eit fall.
For å unngå at mikrokontrollaren skulle ta skade av (mekanisk) støt frå fallet, vart maskinvara plassert i ei plastboks med dempande materiale. Det vart laga eitt hól i sida på boksa slik at Arduino kunne koplast til PC med USB, og eitt i loket for å sikre at trykket inne i boksa endra seg i takt med trykket utanfor. Boksa vart så festa til armen på fallriggen med strips, sjå Figur 7 over.

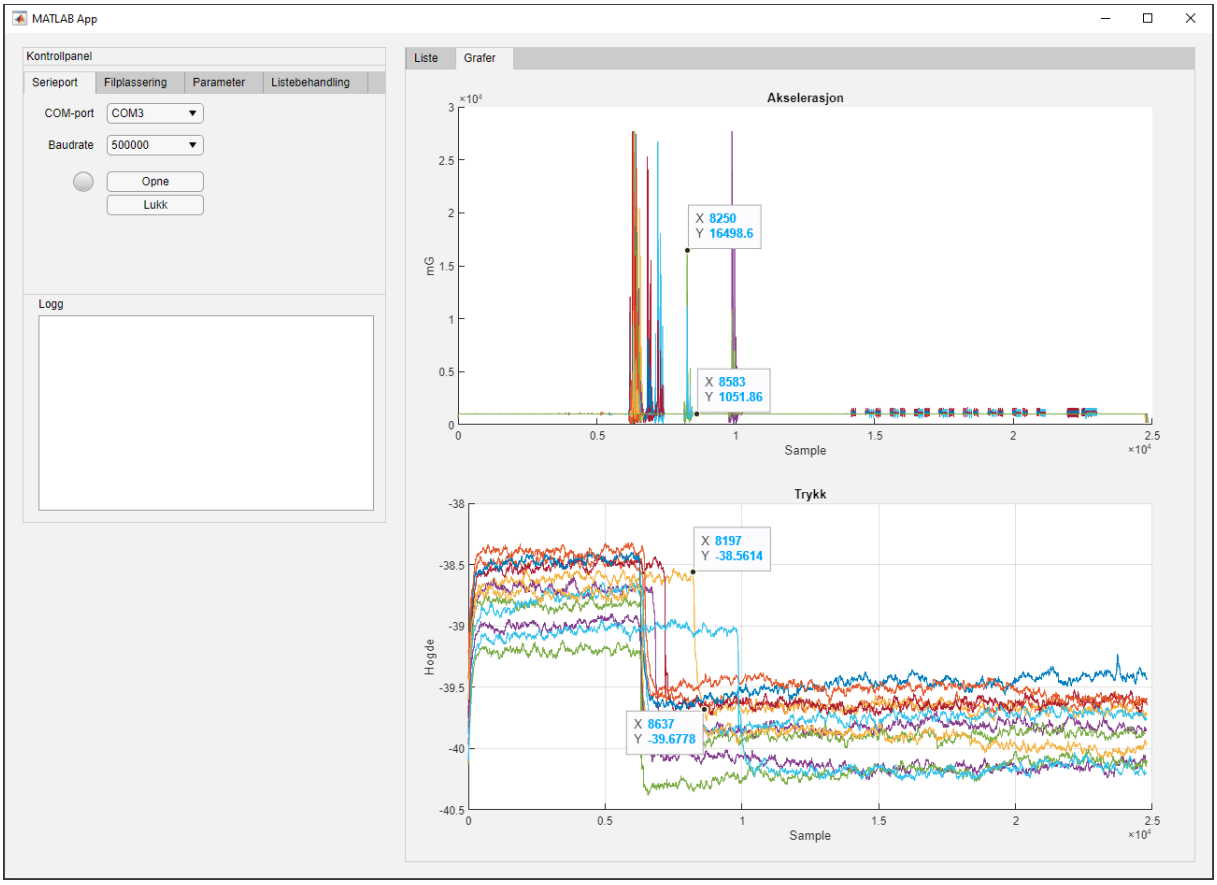


Figur 10: Oversikt over maskinvare

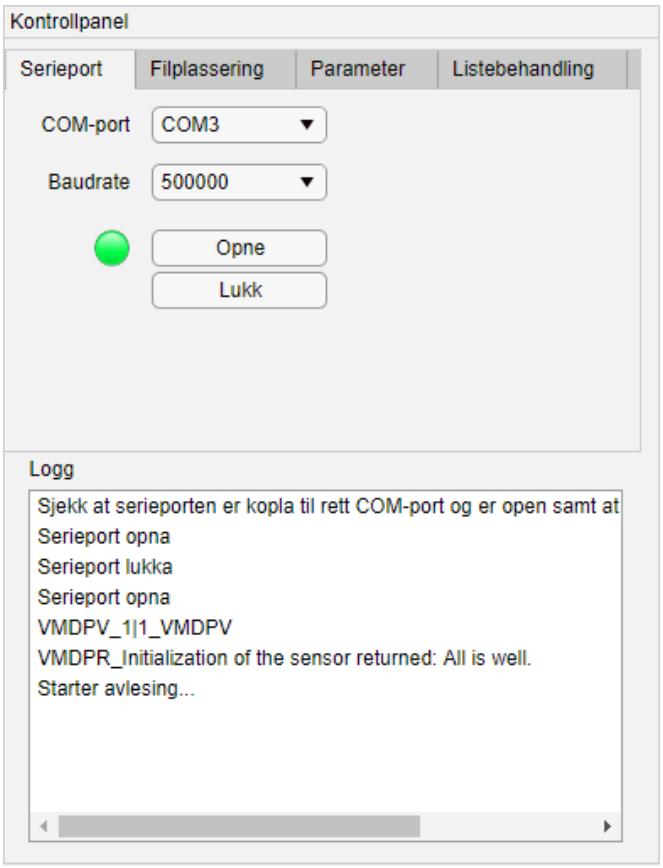
4.3 Programvare

Program for datainnsamling- og overføring til Arduino vart laga først. Deretter vart det laga ein MATLAB-app for lagring og visualisering av innsamla data. Bileta under viser utklipp av brukargrensesnittet til denne appen.

Programkode for både Arduino m./ sensorar og MATLAB-app finnast i Appendiks D

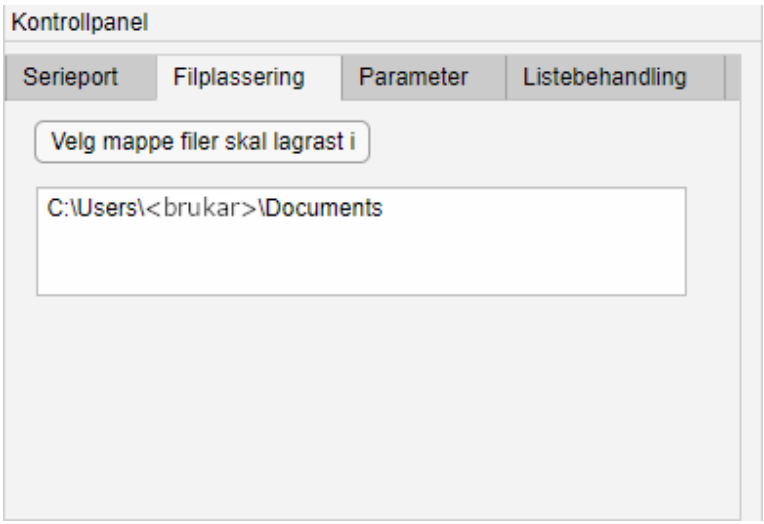


Figur 11: Oversikt over heile brukargrensesnittet



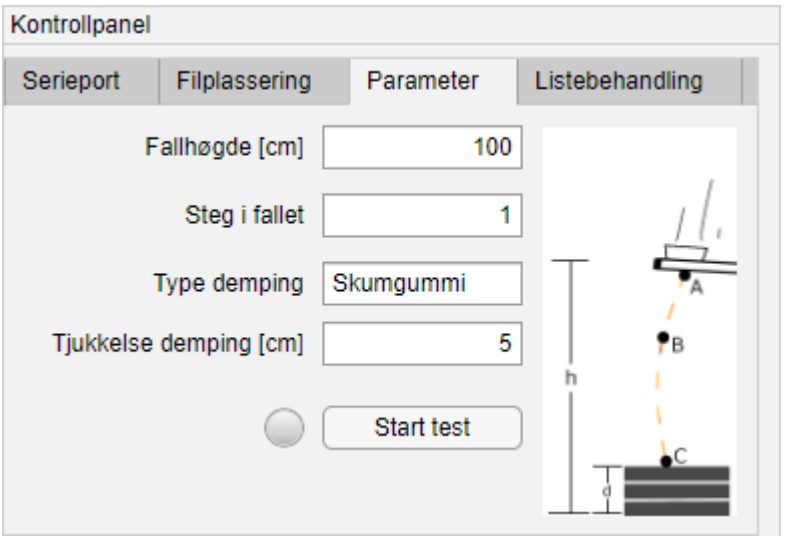
Figur 12: Oversikt over kontrollpanelet

Fana 'Serieport' lar ein velje ein tilgjengeleg COM-port og baudrate samt viser status på tilkoplinga. Ein kan også lukke ein open serieport om ein vel feil eller ønsker å nytte eininga knytt til COM-porten i eit anna program.
'Logg' nedst er synleg uavhengig av vald fane i kontrollpanelet og viser kva handlingar som er utført og eventuelle feil som har oppstått.



Figur 13: Filplassering

Her visast valt filsti der målinger blir lagra. Filsti velgast ved å trykke på knappen øverst og navigere seg fram i Windows Filutforsker. Dette må gjerast før ein kan utføre ei måling.



Figur 14: Parameter

Denne fana held parameter for testen som skal utførast. Her kan ein også starte testen etter ein har ført inn ønska verdiar i tekstfelta. Figuren på høgre side av felta viser kva parametrane representerer.
Resultata frå måleserien blir automatisk lagra med tidsstempling i mappa vald i Figur 13 over.

Figur 15: Listebehandling

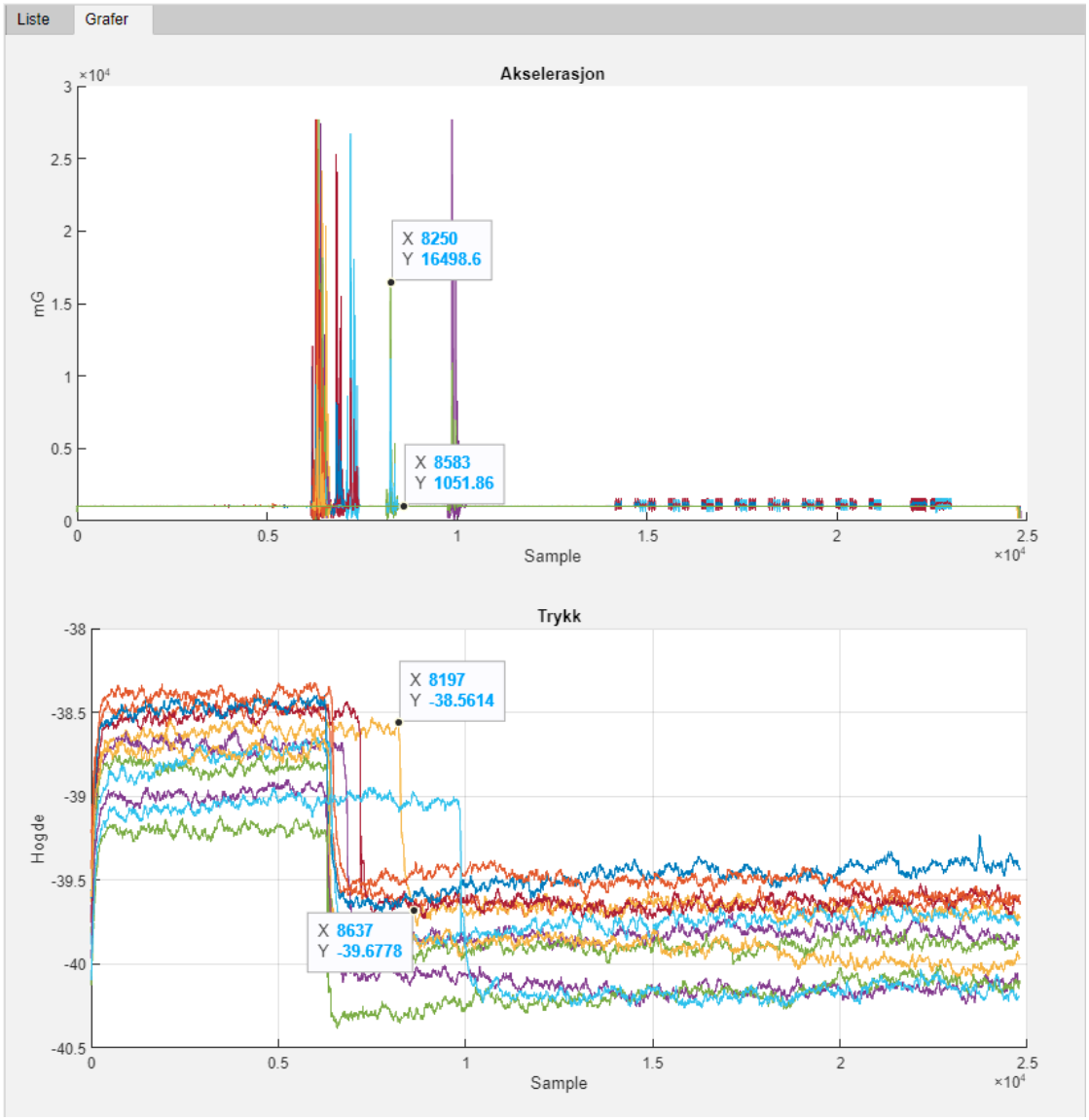
Denne fana lar ein legge til eller fjerne markerte måleserier/filer frå listevisninga vist i Figur 16 under. Ein kan legge til eller fjerne mange måleserier samstundes.

Liste | Grafer

Filnavn	Fallhogde	Steg	Type demping	Tjukkelse demping	Maks aks.	Fall
data_2021_04_14_17_41_05	133	1	Skumgummi og puter	13	16425	☐
data_2021_04_14_17_46_32	133	1	Skumgummi og puter	13	16499	☐
data_2021_04_14_17_51_58	133	1	Skumgummi	2	25245	☑
data_2021_04_14_18_01_09	133	1	Skumgummi	2	25635	☐
data_2021_04_14_18_04_19	133	1	Skumgummi	2	27428	☐
data_2021_04_14_18_07_43	133	1	Skumgummi	2	26706	☑
data_2021_04_14_18_11_03	133	1	Skumgummi	2	27711	☐
data_2021_04_14_18_14_29	133	1	Skumgummi	2	24102	☐
data_2021_04_14_18_17_56	133	1	Skumgummi	2	27711	☐
data_2021_04_14_18_21_30	133	1	Skumgummi	2	27711	☑
data_2021_04_14_18_27_51	133	1	Skumgummi	2	27711	☐
data_2021_04_14_18_31_31	133	1	Skumgummi	2	27711	☐

Figur 16: Listevisning over innlasta måleserier

Lista over viser innlasta måleseriar og tilknytte parameter ført inn i Figur 14 samt dei utvalde dataa maksimal akselerasjon og om fall vart detektert av fallsensoren. Ved å trykke på ei vilkårleg kolonne i ei rad og trykke på 'Fjerne' vist i Figur 15, vil heile rada fjernast frå lista og grafane i Figur 17 under.



Figur 17: Grafar frå måleseriar

Figuren viser plot av kvar av måleseriane lasta inn i Figur 16 over. Grafane er interaktive slik at ein kan zoome inn på eit område og trykke på eit av plotta for å få opp meir informasjon om det valde punktet (dei små boksane med X- og Y-koordinatar). Ein kan også lagre eitt av plotta om gongen som eit bilete med høg oppløysing (betre enn figuren).

4.4 Demping

For å unngå å skade golvet vart det nytta tynt, elektrisk isolerande materiale liknande skumgummi funne på høgspenningslaben på skulen. Kvar pute var om lag 1 cm tjukk. Det er ukjent kva fjørkonstanten til desse putene er.
I tillegg vart det testa med sofaputer for å simulere fall i sofa eller seng og sjå korleis dette påverka deteksjon av fall.

5 Testing

5.1 Ytre faktorar

Testane vart utført i ukontrollerte omgivnadar i to lokale. Temperatur er antatt å ligge mellom 16-21°C. Luftfuktigheit er ukjent. Trykkendringar i form av vind vart kontrollerte til dei grader det var mogleg. Det er likevel sannsynleg at dette har spelt inn på målingane.

5.2 Programvare

Test av programkode for akselerometer med og utan skriving til serieport gir ein maksimal samplingsrate på høvesvis ca. 660 Hz og 960 Hz.
Grunna den lave presisjonen til barometeret vart det lagt størst vekt på å samle inn og vurdere målingar frå akselerometeret.
Ved bruk av programkode for både akselerometer og barometer blir samplingsraten redusert betrakteleg, det tar om lag 5-7 ms å sample barometeret. For å unngå for stor reduksjon av samplingsraten til akselerometeret, blir barometeret berre sampla ein gong for kvar sjuande sampling av akselerometeret. Dette i tillegg til tida det tar å sende data over serieporten gir ein reell samplingsrate på ca. 205 Hz for akselerometeret og 29 Hz for barometeret.

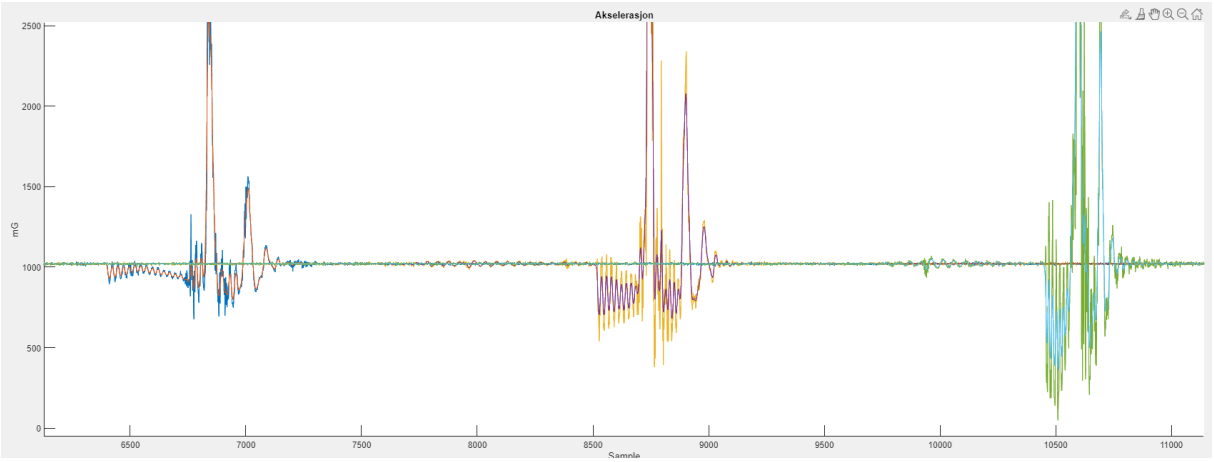
5.3 Fallrigg

Gjennomføring av testar blir utført slik:

1) Lås armen i øvre posisjon med utløysarmekanismen.
2) Sjekk at metallstonga trykker inn knappen, elles vil ikkje fall bli detektert korrekt i MATLAB-programmet.
3) Før inn parameter for fallet som skal utførast i brukargrensesnittet. Dette gjer ein kun første gongen for eit sett av testar med like parameter.
4) Vent 2-3 min frå steg 1 er fullført. Dette for å unngå eventuelle sperrer for hyppig gjentatte fall.
5) Start loggføring og vent ca. 30 sek. Dra deretter i utløysarmekanismen for å starte fallet. Ventinga i dette steget er for å sikre stabile målingar i starten, spesielt frå barometeret.
6) Vent til logging er ferdig, gjenta deretter frå steg 1.

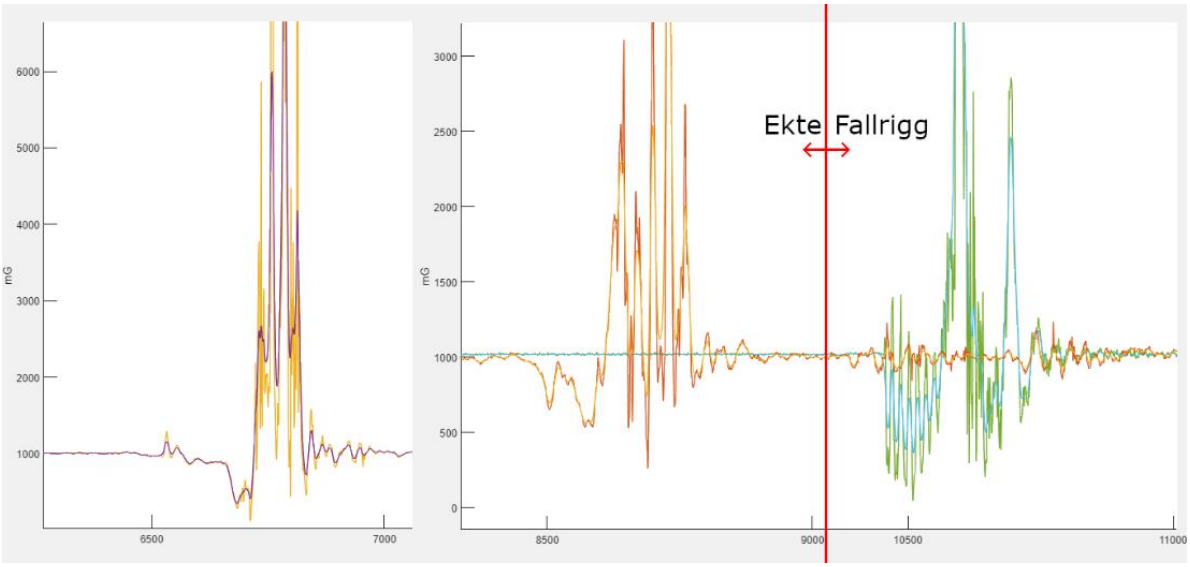
Testtida som er brukt (og foreløpig hardkoda i program til Arduino) er 120 sekund. Om ein slepp armen etter 30-40 sekund gir dette tilstrekkeleg tid til å fange opp heile vibrasjonsmønsteret fallsensoren avgir ved detektert fall.

Målingar frå akselerometeret viser at fallriggen har ein oscillasjon i sjølve fallet, sjå figur under.



Figur 18: Oscillasjon ved ulike akselerasjoner under fall

Motvekta på fallriggen (Figur 9) vart flytta nærare rotasjonssenteret til armen for kvart av dei tre falla, noko som aukar akselerasjonen under fallet. Figuren over viser korleis amplituden til oscillasjonen aukar ved reduksjon i motvekt (nærare fritt fall).
For å vurdere om denne oscillasjonen kunne vere eit problem, vart det utført nokre testar med ekte fall. Testperson heldt sensorane inntil brystet i same høgde som fall med fallriggen vart utført frå og fall sidelengs med litt demping (for å unngå skade). Figuren under viser forskjellen mellom to ekte fall (venstre) og eitt simulert fall med fallrigg (høgre).



Figur 19: Ulik karakteristikk i fallet for ekte fall (V) i forhold til fallrigg (H)

For å minimere eller fjerne denne oscillasjonen vart det prøvd å stabilisere riggen mest mogleg, slik at det ikkje skulle oppstå bevegelse i sjølve riggen som kan påverke målingane. Hjula på tralla vart demontert, hengslene mellom tralla og armen vart fylt med tekstilar for å dempe friksjon i metall-mot-metall forbindelsen og laterale bevegelsar. Desse tiltaka førte ikkje til forbetring mtp. oscillasjonen. Ei mogleg forklaring på oscillasjonen er den lange arma i seg sjølv og/eller materialvalet [4].

6 Diskusjon

Grunna mange uventa hindringar vart den originale framdriftsplanen forkasta. I underkapitla er kvar del av oppgåva diskutert.

6.1 Fallrigg

Det vart i utgangspunktet planlagt å nytte robotarm UR5e for å simulere fall, då den gir svært kontrollerbare og repeterbare testforhold. Tidlege testar viste at robotarma ikkje klarte å løyse ut fallalarmen grunna for lav akselerasjon og deakselerasjon. Robotarma vart difor bytta ut til fordel for fallriggen. Som understreka i kap. 5.3 medførte dette ein uønskt oscillasjon, men grunna tidsavgrensinga til prosjektet vart denne løysinga behaldt.

Dersom fallsensoren bruker ei grensebasert algoritme og er avhengig av karakteristikken vist i [5, p. 778] [6, p. 196] kan dette forklare kvifor fallsensoren slit med å detektere fall i desse testane.
Det er uheldig at fallriggen har ein karakteristikk så ulik ekte fall, då det blir ugyldig å påstå at resultata frå desse testane speglar att fallsensorens prestasjonar under vanleg bruk hos målgruppa.

6.2 Fallsensor

Bergen kommune ga ikkje løyve til å opne fallsensoren for grundigare studiar og Tunstall ønska ikkje å dele teknisk dokumentasjon av konfidensielt omsyn. Det er ikkje kjent kva barometer, akselerometer, blåtann- og radiomodul som er brukt, ei heller samplingsrate og måleområde for sensorane. Det er då svært vanskeleg å seie noko om kva som skjer inne i fallsensoren mtp. avlesing av sensorar og signalbehandling.

Responssenteret fortel at eldre ofte fell sakte, gjerne at dei mister styrken i eine beinet og sig sakte ned. Det er sjølvsagt vanskeleg å skilje denne typen bevegelse frå ein ønska bevegelse som å sette seg ned i ein stol eller legge seg i senga med eit avgrensa tal sensorar.
Tunstall skriv dette i installasjonsrettleiinga: *«The fall detection technology in the Vibby does not allow analysis and interpretation of all fall situations. Soft falls, slumping falls, descent-controlled falls against a wall or a chair, etc…are not be detected by the Vibby. »* [3].
Dette kan bety at denne sensoren ikkje er eigna til Bergen kommune sitt bruksområde.

I møte med Tunstall vart det lagt mykje vekt på batterilevetid og opplyst om at Vitalbase har utfordringar med å nå dei strenge krava fransk e-helse har på det området. Det er rimeleg å anta at fallsensoren ikkje nytter ei maskinlæringsbasert algoritme for deteksjon av fall, då dei gjerne krev meir prosessering og dermed meir straum [7, p. 23].
[8] viser til tidlegare studiar der det er funne at mesteparten av menneskeleg bevegelse skjer under 20 Hz, det betyr at sensorar må ha ein samplingsrate på 40 Hz for å fange opp denne bevegelsen (etter Nyquist – Shannon samplingsteorem).
Skulle fallsensoren likevel nytte ei maskinlæringsbasert algoritme viser [9, p. 782] at visse algoritmer får dårlegare treffsikkerheit ved samplingsrater under enn 50 Hz.

[10, 11, 12, 13, 14, 15] plasserer sensorar rundt kroppens massesenter, dvs. frå hofte til brystkasse. Desse plasseringane gir best resultat for dei valgte algoritmene i desse rapportane.

6.3 Arduino

På bakgrunn av tidlegare erfaring med bibliotek til Arduino for ulike sensorkort vart det antatt at implementasjon av programvare til akselerometer og barometer skulle vere smertefritt. Det viste seg tidleg at samplingsraten til både akselerometer og barometer var mykje lavare enn det sensoren kan levere etter den oppgitte spesifikasjonen. For å få eit mest mogleg nøyaktig bilete på korleis akselerasjon og trykk varierer under eit fall vart det brukt svært mykje tid på å prøve å modifisere biblioteka. Det lukkast med å auke samplingsraten til akselerometeret, men ikkje barometeret. Sjå kapittel 5 for resultat av tiltaka.
I tillegg medfører tidsbruk frå skriving over serieporten eit mindre tap av samplingsrate hos sensorane.

Kanskje skulle ein valt andre sensorar eller ein kraftigare mikroprosessor med meir internt minne.

6.4 Måleresultat

Plot av lufttrykket viser at trykksensoren ikkje gir målingar ein kan legge stor vekt på grunna den relativt lave presisjonen. Ein ser likevel ein trend i høgdeforskjell under sjølve fallet, men denne kan bli påverka ved å f.eks. opne eller lukke eit vindauge.

Grunna oscillasjonen frå fallriggen kan resultata frå testane vere misvisande. Ein kan følgeleg ikkje vurdere fallsensoren utifrå urealistiske testar som ikkje speglar att reelle situasjonar i bruk.

Ein kan undrast over at målingar utført på Høgskulen hadde større prosent deteksjon enn dei som vart utført i anna lokale. Det er ikkje mogleg å sjå forskjellar i karakteristikken til målingar av verken akselerasjon eller trykk i desse to tilfella. Einaste forskjellen mellom måleseriar utført dei to stadane er den absolutte verdien av lufttrykket (avhengig av høgdeforskjell over havet, lufttemperatur og –fuktigheit). Ein kan ikkje basere algoritma for falldeteksjon på ein absolutt trykkverdi, så det er truleg andre variabler som fører til denne skilnaden.

I tabellane under er påverknaden av ulike variablar sett opp mot kvarandre, og viser sannsynet for deteksjon.

Fallhogde [cm]	Steg	Demping [cm]
118	1	2
Akselerasjon [g]		
0.5	0,25	0,15
0.25	0,05	0,10
Modus	Klokke	Pendant

Tabell 1: Akselerasjon vs. modus

Tabellen viser korleis ulike akselerasjoner påverker fallsensoren i dei to modiane. Det kan sjå ut til at fallsensor i pendant-modus er meir konsistent enn i klokke-modus.

Modus	Fallhogde [cm]	Akselerasjon [g]						
Klokke	105	1						
Steg								
1	0	0,1	0,2	0,44	0,5	0,5	0,31	0
2							0	0
Demping [cm]	34	25	13	9	6,5	4,5	2	1

Tabell 2: Steg i fallet vs. tjukkleik demping

Tabellen viser hovudsakelg korleis demping påverker fallsensoren. Den viser også korleis eit ekstra steg i fallet påverkar sannsynet for deteksjon. Det vart ikkje prioritert å teste dette då fallriggen ikkje har ein mekanisme for konsistent utløysing av det andre steget i eit to-stegs fall.

Modus	Steg	Akselerasjon [g]			
Pendant	1	1			
Fallhogde [cm]					
105	0	0,2	0,1	0,2	0,1
133	0,3	0,1	0,2	0,3	0,15
Tjukkelse demping [cm]	34	25	13	4	2

Tabell 3: Fallhøgde vs. tjukkleik demping

Tabellen viser korleis sannsynet for deteksjon varierer med ulik fallhøgde og demping. Det kan sjå ut til at resultata er relativt konsistente i det testa området for fallhøgde og demping. Dersom ein samanliknar med resultata i tabellen over ser det ut til at fallalarmen i klokkemodus har større sannsyn for deteksjon. Det er viktig å merke seg at storparten av desse resultata vart henta frå skulen, medan testar i pendantmodus vart henta frå andre lokale. Det er imidlertid vanskelig å legge mykje vekt på dei, sidan resultat frå testar varierer mykje frå dag til dag og evt. frå stad til stad.

Type demping	Demping [cm]	Akselerasjon [g]
Seng	5	1
Fallhogde [cm]		
38		0,8
70	0,8	0,5
110	0,9	
Modus	Pendant	Klokke

Tabell 4: Fallhøgde vs. modus

Tabellen viser fritt fall i seng. Fallriggen er ikkje nytta her, plastboksa med sensorar vart sleppt for hand. Fall av denne typen har følgeleg ikkje oscillasjonen frå fallriggen. Legg merke til at sannsynet for deteksjon av fall er betrakteleg mykje høgare enn i dei tidlegare tabellane, sjølv med lav fallhøgde.

7 Konklusjon

Grunna oscillasjon i fallriggen og manglande konsistens frå dag til dag eller stad til stad, var det ikkje mogleg å oppdage grenser for fallhøgde, hastigheit eller akselerasjon som ga deteksjon av fall. Det gav difor ikkje meining å teste fall med fleire steg samt lage illustrasjonar eller videoar som viser grenser for deteksjon.

Det er viktig å påpeike at det er utført relativt få falltestar/måleseriar per sett av parameter, men dei gir likevel ein grei peikepinn på kva som skjer. Det kan difor hende at det faktiske sannsynet for deteksjon av fall avviker betydeleg frå det som er funne i oppgåva.

Frå testane utført ser det ut til at fallsensoren ikkje klarer å detektere fall dersom dei hamnar utanfor ein definert karakteristikk, for eksempel grunna oscillasjon i fallriggen. Dersom mange ekte fall hos eldre hamnar utanfor denne karakteristikken, kan det forklare kvifor Bergen kommune opplever at sensoren ikkje er påliteleg.
Etter andre studiar vist til i kap. 6.2 anbefalast det at Responssenteret prøver fallsensoren både i klokke- og halskjede-modus for å sjå om dei opplever liknande resultat.

Vidare arbeid kan konstruere ein ny fallrigg som unngår problemet med oscillasjon, samt har innebygd støtte for å utføre fleire steg i eit fall på ein kontrollert måte. Eventuelt kan ein sjekke om UR10/e har høgare akselerasjon og deakselerasjon enn UR5e slik at denne klarer å løyse ut fallalarmen.

Det kan også vere hensiktsmessig å prøve å nytte ein kraftigare mikrokontroller eller mikro-PC, som for eksempel Raspberry Pi 4. Denne har mykje innebygd minne samt fire kjerner, som gjer det mogleg å sample ein sensor per kjerne pluss ei kjerne til overføring av data trådlaust eller over USB. Ein kan også lagre data frå eitt fall lokalt og overføre etter ferdig utført test.
Det er uvisst om det let seg gjere å nytte dei same sensorkorta som i dette prosjektet og om ein kan nytte same biblioteka.

Det vil då vere mogleg å ha ein høgare samplingsrate for akselerometeret, som gjer det mogleg å detektere mekaniske vibrasjonar frå robotarm eller fallrigg. Ein kan så fjerne desse i signalbehandling om ein treng eit reinare signal for visualisering.

Ein kan også forsøke å modifisere biblioteket til den valde trykksensoren, slik at det er mogleg å køyre denne i Normal Mode. Dette vil gi høgare samplingsrate og dermed mindre støy i signalet. Eventuelt kan ein sjå om det finnast andre sensorkort med meir nøyaktige og presise barometer.

Referansar

[1] Bergen Kommune, «Byrådsavdeling for eldre, helse og frivillighet, seksjon for e-helse,» 12 08 2020. [Internett]. Available: https://www.bergen.kommune.no/omkommunen/avdelinger/byradsavd-for-eldre-helse-og-frivillighet/om-oss. [Funnet 19 01 2021].

[2] Bergen Kommune, «Responssenteret - Om oss,» 09 06 2020. [Internett]. Available: https://www.bergen.kommune.no/omkommunen/avdelinger/responssenter/om-oss. [Funnet 19 01 2021].

[3] Tunstall, «Vibby Installation Guide v.6,» side 14. [Internett]. Available: https://www.tunstall.co.uk/our-products/product-catalogue/vibby-fall-detector/. [Funnet 21 01 2021].

[4] V. A. Kudinov, A. V. Eremin, I. V. Kudinov og A. I. Dovgallo, «Rod resonant oscillations considering material relaxation properties,» *Procedia Engineering,* vol. 176, pp. 226-236, 2017.

[5] A. M. Sabatini, G. Ligorio, A. Mannini, V. Genovese og P. Laura, «Prior-to- and Post-Impact Fall Detection Using Inertial and Barometric Altimeter Measurements,» *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 24, nr. 7, pp. 774-783, 2016.

[6] A. Bourke, J. V. O'Brien og G. M. Lyons, «Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm,» *Gait & Posture,* vol. 26, nr. 2, pp. 194-199, 2007.

[7] E. Casilari-Pérez og F. García-Lagos, «A comprehensive study on the use of artificial neural networks in wearable fall detection systems,» *Expert Systems With Applications,* vol. 138, 2019.

[8] C. V. Bouten, K. T. Koekkoek, M. Verduin, R. Kodde og J. D. Janssen, «A Triaxial Accelerometer and Portable Data Processing Unit for the Assessment of Daily Physical Activity,» *IEEE Transactions on Biomedical Engineering,* vol. 44, nr. 3, pp. 136-147, 1997.

[9] L. Gao, A. K. Bourke og J. Nelson, «Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems,» *Medical Engineering & Physics,* vol. 36, nr. 6, pp. 779-785, 2014.

[10] I. Cleland, B. Kikhia, C. Nugent, A. Boytsov, J. Hallberg, K. Synnes, S. McClean og D. Finlay, «Optimal placement of accelerometers for the detection of everyday activities,» *Sensors,* vol. 13, nr. 7, pp. 9183-9200, 2013.

[11] J. Dai, X. Bai, Z. Yang, Z. Shen og D. Xuan, «PerFallD: A Pervasive Fall Detection System Using Mobile Phones,» *8th IEEE International Conference on Pervasive Computing and Communications Workshops,* pp. 292-297, 2010.

[12] M. Kangas, A. Konttila, P. Lindgren, I. Winblad og T. Jämsä, «Comparison of low-complexity fall detection algorithms for body attached accelerometers,» *Gait & Posture,* vol. 28, pp. 285-291, 2008.

[13] M. Kangas, A. Konttila, I. Winblad og T. Jämsä, «Determination of simple thresholds for accelerometry-based parameters for fall detection,» *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* August 2007.

[14] M. Saleh og R. Le Bouquin Jeannés, «Elderly Fall Detection Using Wearable Sensors: A Low Cost Highly Accurate Algorithm,» *IEEE Sensors Journal,* vol. 19, nr. 8, pp. 3156-3164, 2019.

[15] G. Zhao, Z. Mei, D. Liang, K. Ivanov, Y. Guo, Y. Wang og L. Wang, «Exploration and Implementation of a Pre-Impact Fall Recognition Method Based on an Inertial Body Sensor Network,» *Sensors,* vol. 12, nr. 11, pp. 15338-15355, 2012.

[16] InvenSense, «ICM-20948 Datasheet,» [Internett]. Available: https://invensense.tdk.com/download-pdf/icm-20948-datasheet-2/. [Funnet 05 05 2021].

[17] Bosch Sensortech GmbH, «BMP390L Datasheet Rev. 1.2,» Februar 2020. [Internett]. Available: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp390l/#documents. [Funnet 08 04 2021].

Appendiks A Forkortingar og ordforklaringar

VS	Visual Studio	Programvareutviklingsverktøy
IO	Input-Output	Tilkoplingspunkt for leidningar, inngangar og utgangar
USB	Universal Serial Bus	Standard kabel og kontakt for overføring av data
IMU	Inertial Measurement Unit	Eining som måler lineær akselerasjon, rotasjonsrate og av og til retning, ofte brukt til navigasjon i bl.a. droner
MEMS	Micro-electro-mechanical system	Utruleg små maskiner, sensorar blir ofte laga på denne måten for å spare plass og vekt.
I^2C / IIC	Inter-Integrated Circuit	Kommunikasjonsprotokoll
SPI	Serial Peripheral Interface	Kommunikasjonsprotokoll
JST	Japan Solderless Terminal	Elektrisk kontakt, ofte til hobbybruk

Appendiks B Framdriftsplan / prosjektdagbok

Som nemnt i kap. 3.3 vart den originale framdriftsplanen forkasta. Grunna stor usikkerheit i prosjektet framover vart det ikkje brukt tid på å lage ein ny plan. Det er ført timeliste / dagbok over det som er utført kvar dag. Den originale framdriftsplanen og dagboka ligg på kvart sitt ark i eit Excel-dokument, dette er lagt ved som vedlegg.

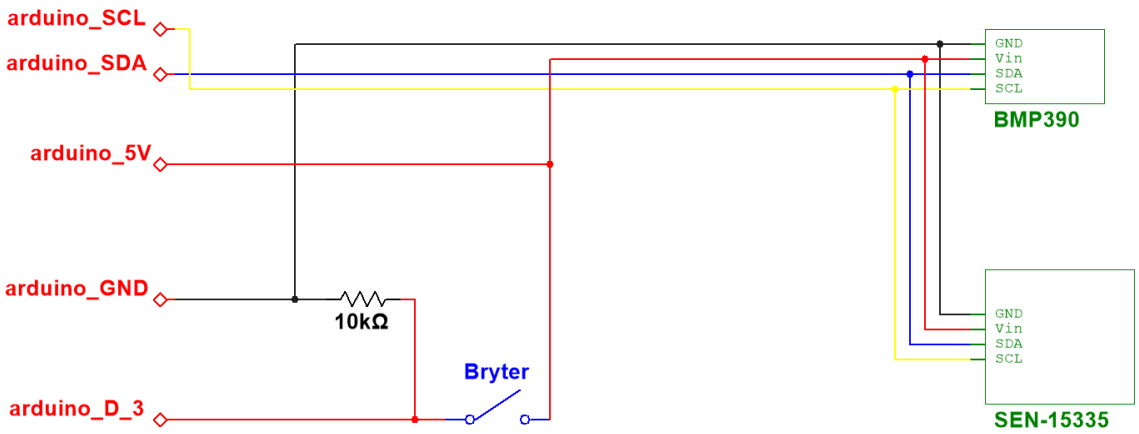
Appendiks C Meir om maskinvare

C.1 Oppsummering

- Mikrokontroller: Arduino Uno Rev.3
- Barometer: Adafruit BMP390 (Bosch BMP390L)
- Akselerometer: SparkFun SEN-15335 (InvenSense ICM-20948 IMU)

Begge sensorkorta vart kopla til Arduino med JST 4-pin Qwiic I^2C-kontakter og bruker standard addresser.

Figuren under viser koplingsskjemaet som vart nytta. «Bryter» viser til trykknappen på armen vist i Figur 8.



Figur 20: Koplingsskjema for Arduino og sensorar

C.2 Akselerometer

Frå datablad [16]: *«The ICM-20948 is the world's lowest power 9-axis MotionTracking device that is ideally suited for Smartphones, Tablets, Wearable Sensors, and IoT applications. […] ICM-20948 supports an auxiliary I2C interface to external sensors, on-chip 16-bit ADCs, programmable digital filters, an embedded temperature sensor, and programmable interrupts. The device features an operating voltage range down to 1.71V. Communication ports include I2C and high speed SPI at 7 MHz.»*

Programkode laga i oppgåva er basert på SparkFun_ICM_20948 v.1.1.2 bibliotek til Arduino. Modifisert til å kun initialisere akselerometer, auke samplingsrate til 1000Hz og måleområde til ±16g samt berre lese registerbank som held målingar til akselerometer. Det var brukt digitalt lavpassfilter med standard innstillingar frå biblioteket.

3.2 ACCELEROMETER SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8V, VDDIO = 1.8V, T_A=25°C, unless otherwise noted.

NOTES: All specifications apply to Low-Power Mode and Low-Noise Mode, unless noted otherwise

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	ACCEL_FS=0		±2		*G*	1
	ACCEL_FS=1		±4		*G*	1
	ACCEL_FS=2		±8		*G*	1
	ACCEL_FS=3		±16		*G*	1
ADC Word Length	Output in two's complement format		16		Bits	1
Sensitivity Scale Factor	ACCEL_FS=0		16,384		LSB/*g*	1
	ACCEL_FS=1		8,192		LSB/*g*	1
	ACCEL_FS=2		4,096		LSB/*g*	1
	ACCEL_FS=3		2,048		LSB/*g*	1
Initial Tolerance	Component-level		±0.5		%	2
Sensitivity Change vs. Temperature	-40°C to +85°C ACCEL_FS=0		±0.026		%/ºC	2
Nonlinearity	Best Fit Straight Line		±0.5		%	2, 3
Cross-Axis Sensitivity			±2		%	2, 3
ZERO-G OUTPUT						
Initial Tolerance	Component-level, all axes		±25		*mg*	2
Initial Tolerance	Board-level, all axes		±50		*mg*	2
Zero-G Level Change vs. Temperature	0°C to +85°C		±0.80		mg/°C	2
ACCELEROMETER NOISE PERFORMANCE						
Noise Spectral Density	Based on Noise Bandwidth = 10 Hz		230		µg/√Hz	2
LOW PASS FILTER RESPONSE	Programmable Range	5.7		246	Hz	1, 3
ACCELEROMETER STARTUP TIME	From Sleep mode		20		ms	2, 3
	From Cold Start, 1 ms V_{DD} ramp		30		ms	2, 3
OUTPUT DATA RATE	Low-Power Mode	0.27		562.5	Hz	1
	Low-Noise Mode ACCEL_FCHOICE=1; ACCEL_DLPFCFG=x	4.5		1.125k	Hz	
	Low-Noise Mode ACCEL_FCHOICE=0; ACCEL_DLPFCFG=x			4.5k	Hz	

Tabell 5: Spesifikasjonar for akselerometer.

4.22 POWER MODES

Table 12 lists the user-accessible power modes for ICM-20948.

MODE	NAME	GYRO	ACCEL	MAGNETOMETER	DMP
1	Sleep Mode	Off	Off	Off	Off
2	Low-Power Accelerometer Mode	Off	Duty-Cycled	Off	On or Off
3	Low-Noise Accelerometer Mode	Off	On	Off	On or Off
4	Gyroscope Mode	On	Off	Off	On or Off
5	Magnetometer Mode	Off	Off	On	On or Off
6	Accel + Gyro Mode	On	On	Off	On or Off
7	Accel + Magnetometer Mode	Off	On	On	On or Off
8	9-Axis Mode	On	On	On	On or Off

Tabell 6: Modi for akselerometer.

Valde innstillingar i figurane over er omringa av ei raud boks. I Tabell 5 er sensitiviteten oppført til 2048 LSB/g ved måleområde ±16g. Sensoren har 16-bits oppløysing. Minste endring i akselerasjon sensoren kan detektere ved desse innstillingane er $\frac{2*16000\,mg}{2^{16}\,bit} = \frac{32000}{65535} \cong 0.5\,mg/LSB$. Dette er meir enn bra nok for bruken i denne oppgåva.

C.3 Barometer

Frå datablad [17]: *«The BMP390 is a digital sensor with pressure and temperature measurement based on proven sensing principles. The sensor module is housed in an extremely compact 10-pin metal-lid LGA package with a footprint of only 2.0 x 2.0* mm^2 *and max 0.8 mm package height. Its small dimensions and it slow power consumption of 3.2 µA@1Hz allow the implementation in battery driven devices such as mobile phones, GPS modules or watches. »*

Brukte Adafruit_BMP3XX v.2.0.2. Biblioteket støtter ikkje modus for kontinuerleg sampling, som er anbefalt av Bosch for dei fleste bruksområde. Det var brukt mykje tid på å prøve å modifisere dette biblioteket utan hell.
Innstillingar er som følger: 2X temperatur-oversampling og 4X trykk-oversampling. Dette gir ein maksimal samplingsrate:

$$T_{conv}[\mu s] = 234 + (392 + 4 * 2020) + (163 + 2 * 2020) \cong 12909$$

$$f_{max} = \frac{1}{T_{conv}} \cong 77{,}5\ Hz$$

Det vart valt ein IIR-filter koeffisient på 16 , sidan det såg ut som eit greitt kompromiss mellom responstid og støy.

Table 1: Key Features of BMP390

Package	2.0 mm x 2.0 mm x 0.75 mm metal lid LGA
Digital interface	I²C (up to 3.4 MHz) and SPI (3 and 4 wire, up to 10 MHz)
Supply voltage	V_{DD} main supply voltage range: 1.65 V to 3.6 V V_{DDIO} interface voltage range: 1.2 V to 3.6 V
Relative accuracy	typ. ± 3 Pa, equiv. to ± 0.25 m (700 ... 1100 hPa, 25 . . . 40 °C)
Absolute accuracy	typ. ± 50 Pa (300 ...1100 hPa, -0 ...+65 °C)
Temperature coefficient offset	mean value ± 0.6 Pa/K (25 – 40°C @900 hPa)
Current consumption	3.2 µA at 1 Hz pressure and temperature 1.4 µA in sleep mode
Operating range	-40 – +85 °C, 300–1250 hPa
The product is RoHS compliant, halogen-free, MSL1	

Tabell 7: Nøkkeleigenskapar til barometeret

Table 2: General electrical parameter specifications

OPERATING CONDITIONS BMP390						
Parameter	**Symbol**	**Condition**	**Min**	**Typ**	**Max**	**Unit**
Operating temperature range	T_A	operational	-40	25	+85	°C
		full accuracy	0		+65	
Operating pressure range	P	full accuracy	300		1250	hPa
Sensor supply voltage	V_{DD}	ripple max. 50mVpp	1.65	1.8	3.6	V
Interface supply voltage	V_{DDIO}		1.2	1.8	3.6	V
Sleep current	I_{DDSL}	VDD = VDDIO = 1.8 V		1.4		µA
	I_{DDSL}	VDD = VDDIO = 3.6 V		1.5		µA
Peak current	I_{peak}	during pressure measurement		660	730	µA
Peak current	I_{DDT}	during temperature measurement		240	320	µA
Relative accuracy pressure[1]	A_{rel}	700 . . . 1100 hPa 25 . . . 40 °C, 10 kPa steps		±0.03		hPa
				+/-25 cm		cm
Offset temperature coefficient[4]	TCO	900 hPa 25 . . . 40 °C		±0.6[2]		Pa/K
Absolute accuracy pressure	A^P_{full}	900 hPa 25 °C		±0.33		hPa
	A^P_{full}	300. . . 1100 hPa 0 . . . 65 °C		±0.50		hPa
	A^P_{ext}	900. . . 1100 hPa 25 . . . 40 °C		±0.40		hPa
	A^P	1100. . . 1250 hPa 0. . . 65 °C		±0.50		hPa
	A^P	300. . . 1100 hPa -20 °C		±0.50		hPa
	A^P	900. . . 1100 hPa, 25-65°C post soldering + 168 HTOL				hPa
Resolution of output data in highest resolution mode at lowest bandwidth	R^P	Pressure		0.016		Pa
RMS Noise in pressure	$V_{p,full}$	25 Hz ODR, 16xOSR_P & 2xOSR_T		0.9		Pa
	$V_{p,filtered}$	Lowest bandwidth, highest resolution See chapter 3.4.4		0.02		Pa
Absolute accuracy temperature[5]	A^T	@ 25 °C		±0.5		°C
		0 . . . +65 °C		±1.50		°C
Long term stability[6]	ΔP_{stab}	12 months		±0.16		hPa
Solder drifts		Minimum solder height 50 µm		< ±0.8		hPa
Start-up time	$t_{startup}$	Time to first communication after both VDD > 1.8 V and VDDIO > 1.8 V			2	ms
Possible sampling rate	f_{sample}	*osrs_t* = *osrs_p* = 1; See chapter 3.9			200	Hz
ODR accuracy				+/-2	+/- 12[7]	%
Power-on time from stand-by mode					3	ms

Tabell 8: Spesifikasjonar for generelle elektriske parameter

Table 5: *osr_p* settings

Oversampling setting	*osr_p*	Pressure oversampling	Typical pressure resolution	Recommended temperature oversampling
Ultra low power	000	×1	16 bit / 2.64 Pa	×1
Low power	001	×2	17 bit / 1.32 Pa	×1
Standard resolution	010	×4	18 bit / 0.66 Pa	×1
High resolution	011	×8	19 bit / 0.33 Pa	×1
Ultra high resolution	100	×16	20 bit / 0.17 Pa	×2
Highest resolution	101	×32	21 bit / 0.085 Pa	×2

Tabell 9: Oversamplings-innstillingar

Table 7: Noise in pressure

Typical RMS noise in pressure [Pa]								
Oversampling setting	IIR filter coefficient							
	off	2	4	8	16	32	64	128
Ultra low power	3.7	2.0	1.2	0.8	0.4	0.2	0.1	0.1
Low power	2.7	1.5	0.9	0.5	0.3	0.2	0.1	0.1
Standard resolution	2.0	1.1	0.7	0.4	0.3	0.2	0.1	0.04
High resolution	1.6	0.9	0.6	0.3	0.2	0.1	0.1	0.03
Ultra high resolution	1.2	0.6	0.4	0.2	0.1	0.1	0.04	0.03
Highest resolution	0.9	0.5	0.3	0.2	0.1	0.1	<0.1	<0.1

Tabell 10: Støy for OS og IIR-filter

Appendiks D Programvare

Lenke til Arduino-program på OneDrive, Lenke til MATLAB-skript og app på GitHub

D.1 MATLAB-applikasjon

For å motta data sendt frå Arduino vart det laga ein Matlab-app. Den gjer det mogleg å knytte nokre parameter/variabler for ein test til ein serie med måleresultat, starte målingar og visualisere dei i ein graf. Det er ikkje lagt vekt på generell utforming, det er eit veldig spesialisert og lite gjenbrukbart program.

```
classdef Falldata
    properties
        Filnavn
        Fallhogde
        Steg
        Type_demping
        Tjukkelse_demping
        Data
        Fall
        Akselerasjon
        Akselerasjon_filt
        Hogde
    end

    methods
        function obj = Falldata()
            obj.Filnavn = "";
            obj.Fallhogde = 0;
            obj.Steg = 0;
            obj.Type_demping = "";
            obj.Tjukkelse_demping = 0;
            obj.Fall = 0;
            obj.Data = [];
            % obj.Data_filtrert = {};
        end

        function obj = LoggingFerdig(obj)
            % Brukast når logging er ferdig
            % INPUTS: Objekt av datatypen 'Falldata'
            % OUTPUTS: Returnerer input med endra datamedlem. Setter felta
            % Akselerasjon (og _filtrert), Fall og Hogde

            % Finner og lagrer absoluttverdien av akselerasjon
            Ax=obj.Data(:,1);
            Ay=obj.Data(:,2);
            Az=obj.Data(:,3);
            A_tot = sqrt(Ax.^2+Ay.^2+Az.^2);
            obj.Akselerasjon = A_tot;
            % Lavpassfiltrert akselerasjon
            [A_tot_lp, ~, ~] = butterfiltfilt(A_tot, 10, 200, 2, 'lowpass',
'both');
            obj.Akselerasjon_filt = A_tot_lp;

            % Vurderer om det var fall på bakgrunn av vibrasjoner fra
            % klokka
            knapp = obj.Data(:,5);
            knapp_lav = find(~knapp,1); % Første sample der knappen er lav

            % Har valgt nokre tilfeldige verdier som indikerer om fall har
            % oppstått. 900/1100mg, dvs. 1g +- 0.1g
            if(~isempty(A_tot(A_tot(knapp_lav+3000:end)<900 |
A_tot(knapp_lav+3000:end)>1100)))
                obj.Fall = true;
            else
                obj.Fall = false;
            end
```

```
% Rekner ut høgde i forhold til havnivå. Den absolutte verdien
            % er ikkje av interesse, men heller den relative forskjellen
            % (sjølve fallet på ca. 1m)
            trykk_havnivaa = 101325;
            P = obj.Data(:,4);
            obj.Hogde = P;

            for i=1:length(P)
                obj.Hogde(i) = 44330.0 * (1.0 - (P(i)/trykk_havnivaa
)^0.1903);
            end

        end

        function obj = LastData(obj, path, file)
            % INPUT: Filsti- og navn på fila du ønsker å laste inn.
            % OUTPUT: Eit objekt med data frå fila.

            try
                S = load(strcat(path, file));
                lagraNavn = string(fieldnames(S));
                obj = S.(lagraNavn);

            catch ME
                disp(strcat('Feil: ', ME.message))
            end
        end % LastData


    end % methods
end % classdef
```

```
classdef Test_fallsensor_v2 < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        KontrollpanelPanel              matlab.ui.container.Panel
        TabGroup                        matlab.ui.container.TabGroup
        SerieportTab                    matlab.ui.container.Tab
        OpneButton                      matlab.ui.control.Button
        COMportDropDownLabel            matlab.ui.control.Label
        COMportDropDown                 matlab.ui.control.DropDown
        BaudrateDropDownLabel           matlab.ui.control.Label
        BaudrateDropDown                matlab.ui.control.DropDown
        Lamp                            matlab.ui.control.Lamp
        LukkButton                      matlab.ui.control.Button
        FilplasseringTab                matlab.ui.container.Tab
        VelgmappefilerskallagrastiButton  matlab.ui.control.Button
        TextArea                        matlab.ui.control.TextArea
        ParameterTab                    matlab.ui.container.Tab
        FallhgdecmEditFieldLabel        matlab.ui.control.Label
        FallhgdecmEditField             matlab.ui.control.NumericEditField
        TypedempingEditFieldLabel       matlab.ui.control.Label
        TypedempingEditField            matlab.ui.control.EditField
        StegifalletEditFieldLabel       matlab.ui.control.Label
        StegifalletEditField            matlab.ui.control.NumericEditField
        TjukkelsedempingcmEditFieldLabel  matlab.ui.control.Label
        TjukkelsedempingcmEditField     matlab.ui.control.NumericEditField
        Image                           matlab.ui.control.Image
        StarttestButton                 matlab.ui.control.Button
        Lamp_2                          matlab.ui.control.Lamp
        ListebehandlingTab              matlab.ui.container.Tab
        LeggtilButton                   matlab.ui.control.Button
        FjerneButton                    matlab.ui.control.Button
        ListBox                         matlab.ui.control.ListBox
        LoggLabel                       matlab.ui.control.Label
        TabGroup2                       matlab.ui.container.TabGroup
        ListeTab                        matlab.ui.container.Tab
        UITable                         matlab.ui.control.Table
        GraferTab                       matlab.ui.container.Tab
        UIAxes                          matlab.ui.control.UIAxes
        UIAxes2                         matlab.ui.control.UIAxes
    end
```

```
properties (Access = public)
        serieport % Serieporten som brukast for å lese data frå Arduino
        konsolltekst
        filsti
        fallparameter
        falldata
        % Legg til liste for falldata av datatypen Falldata (eigendef.
        % klasse)
        listeFalldata
        UITable_valgtIndeks
        plotHandle_a
        plotHandle_af
        plotHandle_h

    end

    methods (Access = private)

        function nyttFall = LoggData(app)
            s = app.serieport;
            import('Klasser.Falldata')
            if(~strcmp(string(app.filsti),'0') && ~isempty(s))
                app.KontrollpanelPanel.Enable = 'off';
                app.Lamp_2.Color = 'green';

                nyttFall = Falldata();
                nyttFall.Fallhogde = app.FallhgdecmEditField.Value;
                nyttFall.Steg = app.StegifalletEditField.Value;
                nyttFall.Type_demping= app.TypedempingEditField.Value;
                nyttFall.Tjukkelse_demping =
app.TjukkelsedempingcmEditField.Value;

                pause(5)
                while(s.NumBytesAvailable > 0)
                    verdi = { readline(s) };
                    app.konsolltekst = [app.konsolltekst string(verdi)];
                end
                app.ListBox.Items = app.konsolltekst;

                rawut=[];
                cnt=0;
                flag=1;
                pause(1)

                %write(s, "START", "string");
                writeline(s,'START')
                app.konsolltekst = [app.konsolltekst "Starter avlesing..."];
                app.ListBox.Items = app.konsolltekst;
```

```
% Må finne ut kor lang tid det tar totalt å logge, samt når knappen går lav

                while flag==1
                    raw=readline(s);
                    if(cnt == 0)
                        %tid_startLogg = datetime('now');
                        tic_knappEndra = tic;
                        cnt = 1;
                    end
                    if (~isempty(rawut) && cnt<2 && ~all(rawut(:,5)))
                        %tid_knappEndra = datetime('now');
                        toc_knappEndra = toc(tic_knappEndra);
                        tic_stopp = tic;
                        cnt = 2;
                    end
                    flag = ~contains(raw,'Avslutter');

                    try

                        raw2=str2num(raw);
                        rawut=[rawut;raw2];
                    catch
                        app.konsolltekst = [app.konsolltekst "Problem med
konvertering fra tekst til tall"];
                        app.ListBox.Items = app.konsolltekst;
                        %disp('Problem med konvertering fra tekst til tall')
                    end

                    %time_raw=now;
                end
                toc_stopp = toc(tic_stopp);
                tid_slutt=datetime('now');
                % tider = [tid_knappEndra tid_slutt];
                % dt = diff(tider);

                tid_slutt=datestr(tid_slutt,'yyyy_mm_dd_HH_MM_SS');
                nyttFall.Filnavn = strcat('data_', tid_slutt);
                nyttFall.Data = rawut;
                nyttFall = nyttFall.LoggingFerdig();

                % Lagre .csv
                fil=strcat(app.filsti, '\data_', tid_slutt,'.csv');
                dlmwrite(fil,rawut,'precision',8);

                % TODO: Må vurdere om denne er på eit greitt format. Den
                % gir feil i overgang frå knappetrykk
                tidsintervall = (toc_knappEndra + toc_stopp) / length(rawut);
                tidVec_forKnapp = toc_knappEndra:-tidsintervall:0;
                tidVec_forStopp = 0:tidsintervall:toc_stopp;
```

```
% Tidsvektoren må vere like stor som antall målinger, kutter sluttida då
                % den ikkje er kritisk
                while(length(tidVec_forKnapp) + length(tidVec_forStopp) >
length(rawut))
                    tidVec_forStopp = tidVec_forStopp(1:end-1);
                end
                tidVec = [-tidVec_forKnapp tidVec_forStopp];
                nyttFall.Data(:,6) = tidVec;

                % Lagre .mat
                fil2 = strcat(app.filsti, '\data_', tid_slutt);
                save(fil2, 'nyttFall'); % standard format er .mat
                disp(strcat('data_', tid_slutt, ' er lagra til fil'))

                app.listeFalldata = [app.listeFalldata nyttFall]
                %VurderData(app, nyttFall, tid_slutt);
                % Vurder data
                % Plotting av data
                app.Lamp_2.Color = '#d3d3d3';
                app.KontrollpanelPanel.Enable = 'on';

            elseif (strcmp(string(app.filsti),'0') && ~isempty(s))
                % Velg filplassering
                nyttFall = [];
                nyTekst = {'Sjekk at du har valgt ei filplassering.'};
                app.konsolltekst = [app.konsolltekst nyTekst];
                app.ListBox.Items = app.konsolltekst;

            elseif (~strcmp(string(app.filsti),'0') && isempty(s))
                % Sjekk at serieport er open
                nyttFall = [];
                nyTekst = {'Sjekk at serieporten er kopla til rett COM-port og
er open.'};
                app.konsolltekst = [app.konsolltekst nyTekst];
                app.ListBox.Items = app.konsolltekst;
            else
                % Velg filplassering og opne serieport
                nyttFall = [];
                nyTekst = {'Sjekk at serieporten er kopla til rett COM-port og
er open samt at du har valgt ei filplassering.'};
                app.konsolltekst = [app.konsolltekst nyTekst];
                app.ListBox.Items = app.konsolltekst;
            end
        end
```

```
function fall = LastData_arr(app)
            import('Klasser.Falldata');
            [file, path] = uigetfile('.mat', 'MultiSelect', 'on', 'Velg ei
datafil som skal plottast');

            % Sjekk om brukaren har valgt ei fil, for å unngå unntak dersom
            % ikkje
            fannFiler = false;
            if(~strcmp(string(path), '0'))
                if(ischar(file))
                    if(~strcmp(string(file), '0'))
                        fannFiler = true;
                    end
                elseif(iscell(file))
                    for i=1:length(file)
                        if(~strcmp(string(file(i)), '0'))
                            fannFiler = true;
                            break;
                        end
                    end
                end
            end

            %if(~strcmp(string(file), '0') && ~strcmp(string(path), '0'))
            if(fannFiler)
                deleteEntry = []; % Tomme filer som ikkje skal plottast
                % Må sjekke om eg har lasta inn ei eller fleire filer
                % 'char' og 'cell'
                if(ischar(file)) % Ei fil
                    fall = LastData_fil(app, path, file);
                elseif(iscell(file)) % Fleire filer
                    %[fall] = Klasser.Falldata();
                    fall = {};

                    for i=1:length(file)
                        fall{i} = LastData_fil(app, path, file{i});
                        if(isempty(fall{i}))
                            deleteEntry(end+1) = i;
                        end
                    end
                end

                if(~isempty(deleteEntry))
                    fall(deleteEntry) = [];
                end
            else
                fall = [];
            end
        end
```

```
        function  fall = LastData_fil(app, path, file)
            import('Klasser.Falldata');
            fall = Falldata();

            if(~strcmp(string(file), '0'))
                file = strtok((file), '.');

                if(~isempty(app.listeFalldata))
                    erLagtTil =
any(strcmp(string(file),{app.listeFalldata(:).Filnavn}));
                    if(~erLagtTil)
                        fall = fall.LastData(path, file);
                        app.listeFalldata = [app.listeFalldata fall];
                        fall = LastData_obj(app, fall);
                    else
                        nyTekst = {strcat(file, ' er allereie lagt inn.')};
                        app.konsolltekst = [app.konsolltekst nyTekst];
                        app.ListBox.Items = app.konsolltekst;
                    end
                else
                    fall = fall.LastData(path, file);
                    app.listeFalldata = [app.listeFalldata fall];
                    fall = LastData_obj(app, fall);
                end
                if(isempty(fall.Data)) % Dersom fila manglar data av ein eller
annan grunn. Gjer det lettare å sjekke i PlotData()
                    fall = [];
                end
            end
        end

        function fall = LastData_obj(app, fall)
            if (~isempty(fall))
                try
                    import('Klasser.Falldata');

                    oldVals = get(app.UITable, 'Data');
                    max_acc = str2double(compose('%3.f',
max(fall.Akselerasjon)));
                    newVals = {fall.Filnavn, fall.Fallhogde , fall.Steg,
fall.Type_demping, fall.Tjukkelse_demping, max_acc, fall.Fall};
                    newVals = [oldVals; newVals];
                    set(app.UITable, 'Data', newVals)
                catch ME
                    nyTekst = { strcat('Feil: ', ME.message) };
                    disp(nyTekst)
                    app.konsolltekst = [app.konsolltekst nyTekst];
                    app.ListBox.Items = app.konsolltekst;
                end
            end
        end
```

```
function PlotData(app, fall)
            if(~isempty(fall))
                if(iscell(fall))
                    for i=1:length(fall)
                        app.plotHandle_a{end+1} =
plot(app.UIAxes,fall{i}.Akselerasjon);
                        hold(app.UIAxes, 'on')
                        app.plotHandle_af{end+1} = plot(app.UIAxes,
fall{i}.Akselerasjon_filt);
                        %hold(app.UIAxes, 'off')

                        app.plotHandle_h{end+1} =
plot(app.UIAxes2,fall{i}.Hogde);
                        hold(app.UIAxes2, 'on')
                    end
                else
                    app.plotHandle_a{end+1} =
plot(app.UIAxes,fall.Akselerasjon);
                    hold(app.UIAxes, 'on')
                    app.plotHandle_af{end+1} = plot(app.UIAxes,
fall.Akselerasjon_filt);
                    %hold(app.UIAxes, 'off')

                    app.plotHandle_h{end+1} = plot(app.UIAxes2,fall.Hogde);
                    hold(app.UIAxes2, 'on')
                end
            end
        end

    end
```

```
% Callbacks that handle component events
    methods (Access = private)


        % Code that executes after component creation
        function startupFcn(app)
            % Legg til eit utvalg av baudrater i DropDown ved oppstart
            baudrater = {300, 600, 1200, 2400, 4800, 9600, 14400, 19200,
28800, 38400, 57600, 115200, 230400, 250000, 256000, 500000 };
            baudrater = string(baudrater);
            app.BaudrateDropDown.Items = baudrater;

            % Velg den første serieporten i DropDown dersom nokon er
tilgjengelig
            serieporter = serialportlist("available");
            if(~isempty(serieporter))
                app.COMportDropDown.Items = serieporter(1);
            end

            % Lampa som indikerer at serieporten er tilkopla er av ved
oppstart
            app.Lamp.Color = '#d3d3d3';
            app.Lamp_2.Color = '#d3d3d3';

            % Setter mine properties til standardverdier
            app.filsti = 0;
            app.FallhgdecmEditField.Value = 100;
            app.StegifalletEditField.Value = 1;
            app.TypedempingEditField.Value = "Skumgummi";
            app.TjukkelsedempingcmEditField.Value = 5;

            app.konsolltekst = []
            import('Klasser.Falldata');
            %app.UITable.ColumnName = {'Filnavn', 'Fallhogde', 'Steg', 'Type
demping', 'Tjukkelse demping', 'Maks aks.', 'Fall'};

            app.plotHandle_a = {};

        end
```

```
% Drop down opening function: COMportDropDown
        function COMportDropDownOpening(app, event)
            serieporter = serialportlist("available");
            app.COMportDropDown.Items = serieporter;
        end


        % Button pushed function: OpneButton
        function OpneButtonPushed(app, event)
            try
                if(isempty(app.serieport))
                    app.serieport = serialport(app.COMportDropDown.Value,
str2double(app.BaudrateDropDown.Value));
                    app.Lamp.Color = 'green';
                    nyTekst = {'Serieport opna'}
                    app.konsolltekst = [app.konsolltekst nyTekst];
                    app.ListBox.Items = app.konsolltekst;
                    %pause(2);
                    %app.TabGroup.SelectedTab = app.FilplasseringTab;
                end
            catch ME
                %disp(strcat('Problem med tilkopling av serieporten.',
ME.message))
                nyTekst = {strcat('Problem med konvertering fra tekst til
tall', ME.message)}
                app.konsolltekst = [app.konsolltekst nyTekst];
                app.ListBox.Items = app.konsolltekst;
            end
        end


        % Close request function: UIFigure
        function UIFigureCloseRequest(app, event)
            if(exist('app.serieport', 'var'))
                app.serieport = [];
                disp('Lukker serieport')
            end
            delete(app)
        end


        % Button pushed function: LukkButton
        function LukkButtonPushed(app, event)
            if(~isempty(app.serieport))
                app.serieport = [];
                %disp('Serieport lukka.')
                app.konsolltekst = [app.konsolltekst "Serieport lukka"];
                app.ListBox.Items = app.konsolltekst;
                app.Lamp.Color = '#d3d3d3';
            end
        end
```

```
% Button pushed function: VelgmappefilerskallagrastiButton
        function VelgmappefilerskallagrastiButtonPushed(app, event)
            filsti = uigetdir([],'Velg mappa der loggfiler skal lagrast.');
            if(filsti~=0)
                app.TextArea.Value = filsti;
                app.filsti = filsti;
            end
        end


        % Button pushed function: StarttestButton
        function StarttestButtonPushed(app, event)
            fall = LoggData(app);
            LastData_obj(app,fall);
            PlotData(app,fall);
        end


        % Button pushed function: LeggtilButton
        function LeggtilButtonPushed(app, event)
            fall = LastData_arr(app);
            PlotData(app, fall)
        end


        % Button pushed function: FjerneButton
        function FjerneButtonPushed(app, event)
            if(~isempty(app.UITable_valgtIndeks))
                indekser = app.UITable_valgtIndeks(:,1);

                % Må slette innlegg i UITable, begge plot (2 for aks.) samt i
                % listeFalldata datafelt. Antar at synleg indeks i UITable er
                % same indeks som i variablane

                for i=1:length(indekser)
                    delete(app.plotHandle_a{indekser(i)});
                    delete(app.plotHandle_af{indekser(i)});
                    delete(app.plotHandle_h{indekser(i)});
                end
                app.plotHandle_a(indekser) = [];
                app.plotHandle_af(indekser) = [];
                app.plotHandle_h(indekser) = [];

                app.UITable.Data(indekser,:) = [];
            end
        end


        % Cell selection callback: UITable
        function UITableCellSelection(app, event)
            app.UITable_valgtIndeks = event.Indices;
        end
    end
```

```
% Component initialization
    methods (Access = private)


        % Create UIFigure and components
        function createComponents(app)


            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 1292 904];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.CloseRequestFcn = createCallbackFcn(app,
@UIFigureCloseRequest, true);


            % Create KontrollpanelPanel
            app.KontrollpanelPanel = uipanel(app.UIFigure);
            app.KontrollpanelPanel.Title = 'Kontrollpanel';
            app.KontrollpanelPanel.Position = [20 381 389 509];


            % Create TabGroup
            app.TabGroup = uitabgroup(app.KontrollpanelPanel);
            app.TabGroup.Position = [0 244 389 239];


            % Create SerieportTab
            app.SerieportTab = uitab(app.TabGroup);
            app.SerieportTab.Title = 'Serieport';


            % Create OpneButton
            app.OpneButton = uibutton(app.SerieportTab, 'push');
            app.OpneButton.ButtonPushedFcn = createCallbackFcn(app,
@OpneButtonPushed, true);
            app.OpneButton.Position = [90 110 104 22];
            app.OpneButton.Text = 'Opne';


            % Create COMportDropDownLabel
            app.COMportDropDownLabel = uilabel(app.SerieportTab);
            app.COMportDropDownLabel.HorizontalAlignment = 'right';
            app.COMportDropDownLabel.Position = [18 183 58 22];
            app.COMportDropDownLabel.Text = 'COM-port';


            % Create COMportDropDown
            app.COMportDropDown = uidropdown(app.SerieportTab);
            app.COMportDropDown.Items = {};
            app.COMportDropDown.DropDownOpeningFcn = createCallbackFcn(app,
@COMportDropDownOpening, true);
            app.COMportDropDown.Position = [90 183 104 22];
            app.COMportDropDown.Value = {};
```

```
% Create BaudrateDropDownLabel
            app.BaudrateDropDownLabel = uilabel(app.SerieportTab);
            app.BaudrateDropDownLabel.HorizontalAlignment = 'right';
            app.BaudrateDropDownLabel.Position = [18 149 58 22];
            app.BaudrateDropDownLabel.Text = 'Baudrate';


            % Create BaudrateDropDown
            app.BaudrateDropDown = uidropdown(app.SerieportTab);
            app.BaudrateDropDown.Items = {};
            app.BaudrateDropDown.Position = [90 149 104 22];
            app.BaudrateDropDown.Value = {};


            % Create Lamp
            app.Lamp = uilamp(app.SerieportTab);
            app.Lamp.Position = [54 110 22 22];


            % Create LukkButton
            app.LukkButton = uibutton(app.SerieportTab, 'push');
            app.LukkButton.ButtonPushedFcn = createCallbackFcn(app,
@LukkButtonPushed, true);
            app.LukkButton.Position = [90 85 104 22];
            app.LukkButton.Text = 'Lukk';


            % Create FilplasseringTab
            app.FilplasseringTab = uitab(app.TabGroup);
            app.FilplasseringTab.Title = 'Filplassering';


            % Create VelgmappefilerskallagrastiButton
            app.VelgmappefilerskallagrastiButton =
uibutton(app.FilplasseringTab, 'push');
            app.VelgmappefilerskallagrastiButton.ButtonPushedFcn =
createCallbackFcn(app, @VelgmappefilerskallagrastiButtonPushed, true);
            app.VelgmappefilerskallagrastiButton.Position = [17 183 172 22];
            app.VelgmappefilerskallagrastiButton.Text = 'Velg mappe filer skal
lagrast i';


            % Create TextArea
            app.TextArea = uitextarea(app.FilplasseringTab);
            app.TextArea.Editable = 'off';
            app.TextArea.Position = [18 115 341 56];


            % Create ParameterTab
            app.ParameterTab = uitab(app.TabGroup);
            app.ParameterTab.Title = 'Parameter';
```

```
% Create FallhgdecmEditFieldLabel
            app.FallhgdecmEditFieldLabel = uilabel(app.ParameterTab);
            app.FallhgdecmEditFieldLabel.HorizontalAlignment = 'right';
            app.FallhgdecmEditFieldLabel.Position = [72 183 85 22];
            app.FallhgdecmEditFieldLabel.Text = 'Fallhøgde [cm]';


            % Create FallhgdecmEditField
            app.FallhgdecmEditField = uieditfield(app.ParameterTab,
'numeric');
            app.FallhgdecmEditField.Position = [166 183 100 22];


            % Create TypedempingEditFieldLabel
            app.TypedempingEditFieldLabel = uilabel(app.ParameterTab);
            app.TypedempingEditFieldLabel.HorizontalAlignment = 'right';
            app.TypedempingEditFieldLabel.Position = [76 115 81 22];
            app.TypedempingEditFieldLabel.Text = 'Type demping';


            % Create TypedempingEditField
            app.TypedempingEditField = uieditfield(app.ParameterTab, 'text');
            app.TypedempingEditField.Position = [166 115 100 22];


            % Create StegifalletEditFieldLabel
            app.StegifalletEditFieldLabel = uilabel(app.ParameterTab);
            app.StegifalletEditFieldLabel.HorizontalAlignment = 'right';
            app.StegifalletEditFieldLabel.Position = [92 149 65 22];
            app.StegifalletEditFieldLabel.Text = 'Steg i fallet';


            % Create StegifalletEditField
            app.StegifalletEditField = uieditfield(app.ParameterTab,
'numeric');
            app.StegifalletEditField.Position = [166 149 100 22];


            % Create TjukkelsedempingcmEditFieldLabel
            app.TjukkelsedempingcmEditFieldLabel = uilabel(app.ParameterTab);
            app.TjukkelsedempingcmEditFieldLabel.HorizontalAlignment =
'right';
            app.TjukkelsedempingcmEditFieldLabel.Position = [25 85 132 22];
            app.TjukkelsedempingcmEditFieldLabel.Text = 'Tjukkelse demping
[cm]';


            % Create TjukkelsedempingcmEditField
            app.TjukkelsedempingcmEditField = uieditfield(app.ParameterTab,
'numeric');
            app.TjukkelsedempingcmEditField.Position = [166 85 100 22];
```

```
% Create Image
            app.Image = uiimage(app.ParameterTab);
            app.Image.Position = [279 5 97 204];
            app.Image.ImageSource = 'figur_fallparameter_v2.png';


            % Create StarttestButton
            app.StarttestButton = uibutton(app.ParameterTab, 'push');
            app.StarttestButton.ButtonPushedFcn = createCallbackFcn(app,
@StarttestButtonPushed, true);
            app.StarttestButton.Position = [166 43 100 22];
            app.StarttestButton.Text = 'Start test';


            % Create Lamp_2
            app.Lamp_2 = uilamp(app.ParameterTab);
            app.Lamp_2.Position = [137 44 20 20];


            % Create ListebehandlingTab
            app.ListebehandlingTab = uitab(app.TabGroup);
            app.ListebehandlingTab.Title = 'Listebehandling';


            % Create LeggtilButton
            app.LeggtilButton = uibutton(app.ListebehandlingTab, 'push');
            app.LeggtilButton.ButtonPushedFcn = createCallbackFcn(app,
@LeggtilButtonPushed, true);
            app.LeggtilButton.Position = [145 128 100 22];
            app.LeggtilButton.Text = 'Legg til';


            % Create FjerneButton
            app.FjerneButton = uibutton(app.ListebehandlingTab, 'push');
            app.FjerneButton.ButtonPushedFcn = createCallbackFcn(app,
@FjerneButtonPushed, true);
            app.FjerneButton.Position = [145 96 100 22];
            app.FjerneButton.Text = 'Fjerne';


            % Create ListBox
            app.ListBox = uilistbox(app.KontrollpanelPanel);
            app.ListBox.Items = {};
            app.ListBox.Position = [17 13 359 209];
            app.ListBox.Value = {};


            % Create LoggLabel
            app.LoggLabel = uilabel(app.KontrollpanelPanel);
            app.LoggLabel.Position = [17 223 32 22];
            app.LoggLabel.Text = 'Logg';
```

```
% Create TabGroup2
            app.TabGroup2 = uitabgroup(app.UIFigure);
            app.TabGroup2.Position = [429 18 846 872];


            % Create ListeTab
            app.ListeTab = uitab(app.TabGroup2);
            app.ListeTab.Title = 'Liste';


            % Create UITable
            app.UITable = uitable(app.ListeTab);
            app.UITable.ColumnName = {'Filnavn'; 'Fallhogde'; 'Steg'; 'Type
demping'; 'Tjukkelse demping'; 'Maks aks.'; 'Fall'};
            app.UITable.RowName = {};
            app.UITable.CellSelectionCallback = createCallbackFcn(app,
@UITableCellSelection, true);
            app.UITable.Position = [18 21 808 807];


            % Create GraferTab
            app.GraferTab = uitab(app.TabGroup2);
            app.GraferTab.Title = 'Grafer';


            % Create UIAxes
            app.UIAxes = uiaxes(app.GraferTab);
            title(app.UIAxes, 'Akselerasjon')
            xlabel(app.UIAxes, 'Sample')
            ylabel(app.UIAxes, 'mG')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [18 433 791 394];


            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.GraferTab);
            title(app.UIAxes2, 'Trykk')
            xlabel(app.UIAxes2, 'Sample')
            ylabel(app.UIAxes2, 'Hogde')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.XGrid = 'on';
            app.UIAxes2.YGrid = 'on';
            app.UIAxes2.Position = [18 21 791 382];


            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
```

```
% App creation and deletion
    methods (Access = public)


        % Construct app
        function app = Test_fallsensor_v2


            % Create UIFigure and components
            createComponents(app)


            % Register the app with App Designer
            registerApp(app, app.UIFigure)


            % Execute the startup function
            runStartupFcn(app, @startupFcn)


            if nargout == 0
                clear app
            end
        end


        % Code that executes before app deletion
        function delete(app)


            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

D.2 Kjeldekode Arduino

Sidene under inneheld kjeldekode som er brukt. Merk at det berre er modifiserte funksjoner frå biblioteka til sensorane som er lagt ved, då resten av biblioteka er umodifiserte frå den versjonen som er oppgitt over.

Modifikasjonar av bibliotek til sensorane er lista i Appendiks E og Appendiks F under.

```
/*
 Name:        acc_odr_test.ino
 Created:     3/23/2021 10:31:05 AM
 Author:      <brukar>
*/

// the setup function runs once when you press reset or power the board
#include <Adafruit_SPIDevice.h>
#include <Adafruit_I2CRegister.h>
#include <Adafruit_I2CDevice.h>
#include <Adafruit_BusIO_Register.h>
#include <ICM_20948.h>
#include <bmp3_defs.h>
#include <bmp3.h>
#include <Adafruit_BMP3XX.h>
#include <Arduino.h>

//#define DEBUG
#define WIRE_PORT Wire
#define AD0_VAL 1
#define KNAPP 3
#define SEALEVELPRESSURE_HPA (1013.25)

ICM_20948_I2C myICM;
Adafruit_BMP3XX bmp;

const unsigned long LOGGTID_S = 12;
unsigned long tid_start_logging;
static int cnt = 0, cntP = 0, knapp = 1;
bool start_logging = false;
static int aX, aY, aZ, len;
static char buffer[64]; // Størrelsen på transmission buffer til Arduino UNO Rev.3 /
ATmega328p

void setup() {
	Serial.begin(500000);

	WIRE_PORT.begin();
	WIRE_PORT.setClock(400000);

	// Setup SEN-15335
	bool initialized = false;
	while (!initialized) {
		myICM.begin_acc_only(WIRE_PORT, AD0_VAL);

		Serial.print(F("Initialization of the sensor returned: "));
		Serial.println(myICM.statusString());

		// Ønsker å bruke lavpassfilter for akselerometer.
		// Bruker standardverdier som er definert i funksjonen
		// startupDefault
		myICM.enableDLPF(ICM_20948_Internal_Acc, true);

#ifdef DEBUG
		// Gyro avslått:
		// Sjekk at gyro er av sidan det ikkje er i bruk
		// Datablad: Kap. 8.5 PWR_MGMT_2 og Kap.7.1 User Bank 0 Register Map
		uint8_t val;
		myICM.setBank(0);
		myICM.read(AGB0_REG_PWR_MGMT_2, &val, 1);
		Serial.print("Disable gyro: ");
		Serial.println(val, BIN);
```

```
		// Måleområde:
		// Datablad: Kap. 10.15 ACCEL_CONFIG og Kap. 7.3 User Bank 2 Register Map
		myICM.setBank(2);
		myICM.read(AGB2_REG_ACCEL_CONFIG, &val, 1);
		Serial.print("Accel Full-Scale: ");
		Serial.println(val, BIN);

		// Samplingsrate:
		// Datablad: Kap. 7.3 User Bank 2 Register Map og Kap. 10.11/12
ACCEL_SMPLRT_DIV_1/2
		uint8_t div1, div2;
		uint16_t SMPLRT_DIV;
		myICM.setBank(2);
		myICM.read(AGB2_REG_ACCEL_SMPLRT_DIV_1, &div1, 1);
		myICM.read(AGB2_REG_ACCEL_SMPLRT_DIV_2, &div2, 1);
		SMPLRT_DIV = (div1 << 8) | div2;

		Serial.print("Samplerate register 1: ");
		Serial.print(div1, BIN);
		Serial.print("\tReg2: ");
		Serial.println(div2, BIN);
		Serial.print("Samplerate [Hz]: ");
		Serial.println(1125 / (1 + SMPLRT_DIV));
#endif // DEBUG

		if (myICM.status != ICM_20948_Stat_Ok) {
			Serial.println("Trying again...");
			delay(500);
		}
		else initialized = true;
	}

	// SETUP BMP390
	if (!bmp.begin_I2C()) {
		Serial.println("Could not find a valid BMP3 sensor, check wiring!");
		while (true);
	}

	//Set up oversampling and filter initialization
	// Datablad: Kap. 3.5 Filter Selection og Kap. 3.9 Measurement Timings
	// Sensoren er i Forced Mode, instilling for ODR gjeld ikkje då.
	// Viktig å overhalde maks. samplingsrate oppgitt i kap. 3.9

#ifdef DEBUG
	Serial.print("Klarte a sette TEMP_OS til valgt verdi: ");
	Serial.println(bmp.setTemperatureOversampling(BMP3_OVERSAMPLING_2X));
	Serial.print("Klarte a sette PRES_OS til valgt verdi: ");
	Serial.println(bmp.setPressureOversampling(BMP3_OVERSAMPLING_4X));
	Serial.print("Klarte a sette IIR-filter til valgt verdi: ");
	Serial.println(bmp.setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_15));
	Serial.print("Klarte a sette Output Datarate til valgt verdi (ikkje gyldig i
Forced Mode): ");
	Serial.println(bmp.setOutputDataRate(BMP3_ODR_100_HZ));
	Serial.print("Sjekk at innstillinger er gyldige. 0 = alt OK: ");
	Serial.println(bmp.changeSettings());
#else
	bmp.setTemperatureOversampling(BMP3_OVERSAMPLING_2X);
	bmp.setPressureOversampling(BMP3_OVERSAMPLING_4X);
	bmp.setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_15);
	bmp.setOutputDataRate(BMP3_ODR_100_HZ);
	bmp.changeSettings();
#endif // DEBUG
```

```
}

// the loop function runs over and over again until power down or reset
void loop() {
	// Antar at Arduino får ein tekststreng liknande "START\n"
	// for å starte logging
	if (Serial.available()) {
		String s = Serial.readStringUntil('\n');
		if (s.length() > 0) {
			start_logging = !start_logging;
			cnt = 0;
			cntP = 0;
			tid_start_logging = millis();

			// Utfører ei måling av trykk for å unngå udefinert trykk ved
start
			bmp.performReading();
			cntP++;
		}
	}


	while (start_logging) {
		// Avlesing av akselerasjon
		myICM.getAcc();
		cnt++;

		aX = myICM.accX();
		aY = myICM.accY();
		aZ = myICM.accZ();

		// Trykk blir kun målt ein gong for kvar sjuande måling av akselerasjon
		// Dette er for å ikkje redusere samplingsraten til akselerometeret for
mykje
		if (cnt % 7 == 0) {
			bmp.performReading();
			cntP++;
		}
		knapp = digitalRead(KNAPP);

		// Utskrift av målinger
		len = sprintf(buffer, "%i, %i, %i, ", aX, aY, aZ);
		for (int i = 0; i < len; i++)
		{
			Serial.write(buffer[i]);
		}
		Serial.print(bmp.pressure);
		len = sprintf(buffer, ", %i\n", knapp);
		for (int i = 0; i < len; i++)
		{
			Serial.write(buffer[i]);
		}
		if (millis() > tid_start_logging + (LOGGTID_S * 1000)) {
			start_logging = false;
			Serial.println("Avslutter logging.");
#ifdef DEBUG
			Serial.print("Antall lesinger acc: ");
			Serial.println(cnt);
			Serial.print("Antall lesinger trykk: ");
			Serial.println(cntP);
			Serial.print("Frekvens: ");
			Serial.println(cnt / LOGGTID_S);
```

```
#endif // DEBUG
                }
        }
}
```

Appendiks E BMP390:

E.1 Adafruit_BMP3XX.cpp

For å unngå å bruke tid på å skrive til ulike register for kvar måling, er det innført eigen funksjon for å endre innstillingar. Denne ser ikkje ut til å fungere i Forced Mode, som sensoren har kjørt i under alle målingar i denne oppgåva. Det let seg ikkje gjere å sette sensoren i Normal Mode (som er anbefalt av Bosch til dette og dei fleste andre bruksområde).

```
bool Adafruit_BMP3XX::performReading(void) {
//  int8_t rslt;
//  /* Used to select the settings user needs to change */
//  uint16_t settings_sel = 0;
//  /* Variable used to select the sensor component */
//  uint8_t sensor_comp = 0;
//
//  /* Select the pressure and temperature sensor to be enabled */
//  the_sensor.settings.temp_en = BMP3_ENABLE;
//  settings_sel |= BMP3_SEL_TEMP_EN;
//  sensor_comp |= BMP3_TEMP;
//  if (_tempOSEnabled) {
//    settings_sel |= BMP3_SEL_TEMP_OS;
//  }
//
//  the_sensor.settings.press_en = BMP3_ENABLE;
//  settings_sel |= BMP3_SEL_PRESS_EN;
//  sensor_comp |= BMP3_PRESS;
//  if (_presOSEnabled) {
//    settings_sel |= BMP3_SEL_PRESS_OS;
//  }
//
//  if (_filterEnabled) {
//    settings_sel |= BMP3_SEL_IIR_FILTER;
//  }
//
//  if (_ODREnabled) {
//    settings_sel |= BMP3_SEL_ODR;
//  }
//
//  // set interrupt to data ready
//  //settings_sel |= BMP3_SEL_DRDY_EN | BMP3_SEL_LEVEL | BMP3_SEL_LATCH;
//
//
//  /* Set the desired sensor configuration */
//#ifdef BMP3XX_DEBUG
//  Serial.println("Setting sensor settings");
//#endif
//  rslt = bmp3_set_sensor_settings(settings_sel, &the_sensor);
//
//  if (rslt != BMP3_OK)
//    return false;
//
//
//  /* Set the power mode */
//  the_sensor.settings.op_mode = BMP3_MODE_FORCED;
//#ifdef BMP3XX_DEBUG
//  Serial.println(F("Setting power mode"));
//#endif
//  rslt = bmp3_set_op_mode(&the_sensor);
//  if (rslt != BMP3_OK)
//    return false;


  /* Variable used to store the compensated data */
  int8_t rslt;
  rslt = changeSettings();
  struct bmp3_data data;
  uint8_t sensor_comp = 0;
  sensor_comp |= BMP3_TEMP;
  sensor_comp |= BMP3_PRESS;

  /* Temperature and Pressure data are read and stored in the bmp3_data instance
```

```
   */
#ifdef BMP3XX_DEBUG
  Serial.println(F("Getting sensor data"));
#endif
  rslt = bmp3_get_sensor_data(sensor_comp, &data, &the_sensor);
  if (rslt != BMP3_OK)
    return false;

  /*
#ifdef BMP3XX_DEBUG
  Serial.println(F("Analyzing sensor data"));
#endif
  rslt = analyze_sensor_data(&data);
  if (rslt != BMP3_OK)
    return false;
    */

  /* Save the temperature and pressure data */
  temperature = data.temperature;
  pressure = data.pressure;

  return true;
}
```

```
int8_t Adafruit_BMP3XX::changeSettings(void) {
    int8_t rslt = BMP3_OK;
    uint16_t settings_sel = 0;

    /* Select the pressure and temperature sensor to be enabled */
    the_sensor.settings.press_en = BMP3_ENABLE;
    the_sensor.settings.temp_en = BMP3_ENABLE;
    settings_sel |= BMP3_SEL_PRESS_EN | BMP3_SEL_TEMP_EN;

    /* Set pressure settings */
    if (_presOSEnabled) {
        settings_sel |= BMP3_SEL_PRESS_OS;
    }

    /* Set temperature settings */
    if (_tempOSEnabled) {
        settings_sel |= BMP3_SEL_TEMP_OS;
    }

    /* Set IIR filter settings */
    if (_filterEnabled) {
        settings_sel |= BMP3_SEL_IIR_FILTER;
    }

    /* Set data rate settings */
    if (_ODREnabled) {
        settings_sel |= BMP3_SEL_ODR;
    }

    // set interrupt to data ready
    settings_sel |= BMP3_SEL_DRDY_EN | BMP3_SEL_LEVEL | BMP3_SEL_LATCH;

    /* Set the desired sensor configuration */
#ifdef BMP3XX_DEBUG
    Serial.println("Setting sensor settings");
#endif
    rslt = bmp3_set_sensor_settings(settings_sel, &the_sensor);

    if (rslt != BMP3_OK)
        return rslt;


    /* Set the power mode */
    the_sensor.settings.op_mode = BMP3_MODE_FORCED;
#ifdef BMP3XX_DEBUG
    Serial.println(F("Setting power mode"));
#endif
    rslt = bmp3_set_op_mode(&the_sensor);

    if (rslt != BMP3_OK) {
        {
         /*    ("error set power mode \n");
            if (rslt == BMP3_E_INVALID_ODR_OSR_SETTINGS)
                printf("BMP3_E_INVALID_ODR_OSR_SETTINGS \n");*/
            return rslt;
        }

    }

    // If everything went OK, then
    return rslt;
}
```

Appendiks F SEN-15335

Det har også blitt lagt til noko programkode i bibliotek for IMU. Hensikta her var å skru av gyro då den ikkje er i bruk og for å sette akselerometeret i Low-Noise Mode, samt justere diverse parameter for akselerometer. Det er uvisst om det å skru av gyro auka/gjorde det mogleg å auke samplingsraten til akselerometeret.

F.1 ICM_20948_C.c

```
ICM_20948_Status_e ICM_20948_disable_gyro(ICM_20948_Device_t* pdev, bool yes)
{
	ICM_20948_Status_e retval = ICM_20948_Stat_Ok;
	ICM_20948_PWR_MGMT_2_t reg;

	ICM_20948_set_bank(pdev, 0); // Must be in the right bank

	retval = ICM_20948_execute_r(pdev, AGB0_REG_PWR_MGMT_2, (uint8_t*)&reg,
sizeof(ICM_20948_PWR_MGMT_2_t));
	if (retval != ICM_20948_Stat_Ok)
	{
		return retval;
	}

	if (yes)
	{
		reg.DISABLE_GYRO = 7;
	}
	else
	{
		reg.DISABLE_GYRO = 0;
	}

	retval = ICM_20948_execute_w(pdev, AGB0_REG_PWR_MGMT_2, (uint8_t*)&reg,
sizeof(ICM_20948_PWR_MGMT_2_t));
	if (retval != ICM_20948_Stat_Ok)
	{
		return retval;
	}
	return retval;
}
```

F.2 ICM_20948.cpp

```
ICM_20948_Status_e ICM_20948::disableGyro(bool yes) {
    status = ICM_20948_disable_gyro(&_device, yes);
    return status;
}
```

```
ICM_20948_Status_e ICM_20948::startupAccOnly(void)
{
    ICM_20948_Status_e retval = ICM_20948_Stat_Ok;

    retval = checkID();
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    retval = swReset();
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }
    delay(50);

    retval = sleep(false);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    retval = lowPower(false);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    retval = disableGyro(true);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

       // sensors: ICM_20948_Internal_Acc, ICM_20948_Internal_Gyr,
ICM_20948_Internal_Mst
    // options: ICM_20948_Sample_Mode_Continuous or ICM_20948_Sample_Mode_Cycled
    retval = setSampleMode(ICM_20948_Internal_Acc, ICM_20948_Sample_Mode_Continuous);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    ICM_20948_fss_t FSS;
    FSS.a = gpm16;
    retval = setFullScale(ICM_20948_Internal_Acc, FSS);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    ICM_20948_dlpcfg_t dlpcfg;
    dlpcfg.a = acc_d473bw_n499bw;
    retval = setDLPFcfg(ICM_20948_Internal_Acc, dlpcfg);
```

```
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    retval = enableDLPF(ICM_20948_Internal_Acc, false);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }
    retval = enableDLPF(ICM_20948_Internal_Gyr, false);
    if (retval != ICM_20948_Stat_Ok)
    {
        status = retval;
        return status;
    }

    return status;
}
```

```
ICM_20948_Status_e ICM_20948_I2C::begin_acc_only(TwoWire& wirePort, bool ad0val,
uint8_t ad0pin)
{
    // Associate
    _ad0 = ad0pin;
    _i2c = &wirePort;
    _ad0val = ad0val;

    _addr = ICM_20948_I2C_ADDR_AD0;
    if (_ad0val)
    {
        _addr = ICM_20948_I2C_ADDR_AD1;
    }

    // Set pinmodes
    if (_ad0 != ICM_20948_ARD_UNUSED_PIN)
    {
        pinMode(_ad0, OUTPUT);
    }

    // Set pins to default positions
    if (_ad0 != ICM_20948_ARD_UNUSED_PIN)
    {
        digitalWrite(_ad0, _ad0val);
    }

    // _i2c->begin(); // Moved into user's sketch

    // Set up the serif
    _serif.write = ICM_20948_write_I2C;
    _serif.read = ICM_20948_read_I2C;
    _serif.user = (void*)this; // refer to yourself in the user field

    // Link the serif
    _device._serif = &_serif;

    // Perform default startup
    status = startupAccOnly();
    if (status != ICM_20948_Stat_Ok)
    {
        return status;
    }
    return status;
}
```