



## AARHUS SCHOOL OF ENGINEERING

### SUNDHEDSTEKNOLOGI 3. SEMESTERPROJEKT

---

# Rapport

---

#### *Gruppe 2*

Anne Bundgaard Hoelgaard	(201404492)
Mette Hammer Nielsen-Kudsk	(201408391)
Ditte Heebøll Callesen	(201408392)
Martin Banasik	(201408398)
Albert Jakob Fredshavn	(201408425)
Johan Mathias Munk	(201408450)

#### *Vejleder:*

Studentervejleder  
Peter Johansen  
Aarhus Universitet

16. december 2015

# Resumé

---

Dette projekt besæftiger sig med blodtryksmåling, hvor der ud fra et blodtrykssignal kan bestemmes en systolisk og diastolisk værdi, samt en puls værdi. Formålet er at bygge en hardwaredel, der kan bearbejde et analogt signal. Signalet skal forstærkes, filtreres og sendes igennem en Data Acquisition, der kan lave det analoge signal om til et digitalt signal. Herefter skal signalet ind i softwareneden. Her er formålet at få programmeret algoritmer, der kan detektere puls, samt detekterer systolisk og diastolisk værdi. Signalet bliver observeret fra starttidspunktet, hvor softwaren finder maksimums- og minimumsværdier for hvert tredje sekund. Maksimumsværdien svarer til den systoliske værdi, minimumsværdien svarer til den diastoliske værdi.

Herudover skal programmet kunne nulpunktsjusteres ved opstart og kalibreres når en forsker finder det nødvendigt. Projektet er lavet til forskningsbrug og derfor blev det vurderet, at det er nødvendigt for en forsker, at kunne gemme sine målinger. Programmeringen gør det derfor muligt for forskeren, at kunne gemme sine målinger i en database.

Resultatet for dette projekt viser, at det er muligt vha. en hardware- og softwaredel, at måle et blodtryk. Gennem videreudvikling kan det blive muligt at detektere forhøjet blodtryk og lavt blodtryk i samme program. Dette ville være nyttigt, hvis projektet i fremtiden skulle bruges til patienter. I projektet er der dog udelukkende blevet arbejdet med måling af blodtryk til forskningsbrug.

# **Abstract**

---

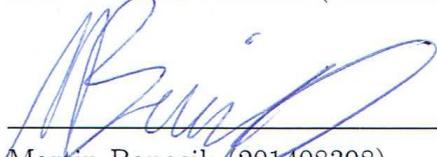
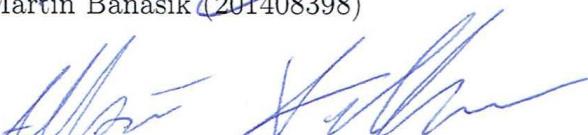
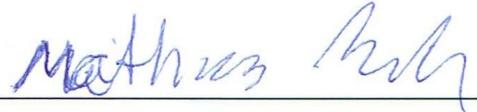
This project investigates blood pressure measurement and the examination of systolic and diastolic values, as well as a pulse, derived from blood pressure signals. The goal of this project is to construct hardware which is able to manipulate and adapt an analog signal from pressure force. A filter is used to amplify the signal before it travels to a Data Acquisition device. This device will transform the analog signal to a digital signal. Thereafter the transformed signal goes into the software-part, also constructed specifically for this project. Here, the objective of the program is to consist of developed algorithms which are able to determine and depict systolic and diastolic values. The signal is being observed from its starting time, where the software discovers maximum and minimum values, every third second. The maximum value correlates to the systolic value and the minimum correlates to the diastolic value.

Aside from the above, the program is able to do zero-point adjustment from startup, as well as calibrating whenever a researcher finds it necessary. This project is made for research, and it is therefore fundamental that researchers are able to save their data values. Therefore, the program includes a database, where data values can be stored.

The result of this project presents; it is possible to measure a blood pressure using hardware and software components. Through future development it can be possible to detect elevated and suppressed blood pressures by the same program. This would be useful if, in the near future, this project were to be used on patients. Although, throughout the project, the main focus has been measurements of blood pressure for research purposes.

# Underskrifter

## Gruppemedlemmer

	<u>16/12-2015</u>
Anne Bundgaard Hoelgaard (201404492)	Dato
	<u>16/12 - 2015</u>
Mette Hammer Nielsen-Kudsk (201408391)	Dato
	<u>16/12 - 2015</u>
Ditte Heebøll Callesen (201408392)	Dato
	<u>16/12 - 15</u>
Martin Banasik (201408398)	Dato
	<u>16/12 - 2015</u>
Albert Jakob Fredshavn (201408425)	Dato
	<u>16/12 - 15</u>
Johan Mathias Munk (201408450)	Dato

## Vejleder

	<u>16/12-15</u>
Peter Johansen	Dato

# Godkendelsesformular

## Godkendelsesformular

Antal sider: 37

Forfattere:

Anne Bundgaard Hoelgaard (201404492)

Mette Hammer Nielsen-Kudsk (201408391)

Ditte Heebøll Callesen (201408392)

Martin Banasik (201408398)

Albert Jakob Fredshavn (201408425)

Johan Mathias Munk (201408450)

Godkendes af      Peter Johansen

Kunde                Aarhus Universitet

Ved underskrivelse af dette dokument accepteres det af begge parter som værende kravene til udviklingen af det ønskede system.

Dato: 16/12-2015

Kundens underskrift

Leverandørens underskrift

# Ordliste

---

Ord	Forklaring
(F)URPS+	Et akronym, der repræsenterer klassificering af softwarens kvalitet
GUI	Graphical User Interface (Grafisk brugergrænseflade)
DAQ	Data acquisition
SysML	Systems Modeling Language – sprog til visuel fremstilling af systemer
UML	Unified modelling language – sprog til oversigtsfremstilling af klasser i programmering
KSS	Kommunikation og Samarbejde i Sundhedsvæsnet
BDD	Block Definition Diagram
IBD	Intern Block Diagram
SD	Sekvensdiagram
ISE	Indledende System Engineering
KVI	Kardiovaskulær instrumentering
Hjerteinsufficiens	Hjertesvigt
DSB	Digital Signalbehandling
MTTR	Mean Time To Restore
MTBF	Mean Time Between Failure
Anatomi	En organismes indre og ydre opbygning
Veroboard	Printplade med isolerede kobberører
CMRR	Common Mode Rejection Ratio
Hæmodynamik	Læren om blodets bevægelse
Transducer	Tryktransducer, omsætter en trykændring til en spænding
MoSCoW	Metode, benyttes til at beskrive, hvad programmet skal kunne og det der ønskes, men ikke vil ske
Apopleksi	Hjerneblødninger og blodpropper

# Indholdsfortegnelse

---

Resumé	i
Abstract	ii
Underskrifter	iii
Godkendelsesformular	iv
Ordliste	v
<b>Kapitel 1 Indledning</b>	<b>1</b>
<b>Kapitel 2 Projektformulering og afgrænsning</b>	<b>3</b>
<b>Kapitel 3 Blodtryk</b>	<b>5</b>
3.1 Hvad betyder blodtryk? . . . . .	5
3.2 Hypertension . . . . .	6
3.3 Hypotension . . . . .	7
<b>Kapitel 4 Systembeskrivelse</b>	<b>8</b>
<b>Kapitel 5 Krav</b>	<b>10</b>
<b>Kapitel 6 Projektbeskrivelse</b>	<b>13</b>
6.1 Projektgennemførelse . . . . .	13
6.2 Projektstyring . . . . .	14
6.3 Metoder . . . . .	15
6.4 Systemarkitektur . . . . .	16
6.5 Hardware . . . . .	16
6.5.1 Design . . . . .	16
6.5.2 Implementering . . . . .	16
6.5.3 Test . . . . .	19
6.6 Software . . . . .	22
6.6.1 Design . . . . .	22
6.6.2 Implementering . . . . .	24
6.6.3 Test . . . . .	28
6.6.4 Integrationstest . . . . .	29
6.7 Udviklingsværktøjer . . . . .	30
6.8 Resultater og diskussion . . . . .	30
6.9 Opnåede erfaringer . . . . .	32
6.10 Perspektivering - Fremtidigt arbejde . . . . .	33
<b>Kapitel 7 Konklusion</b>	<b>34</b>

<b>Kapitel 8 Bilag</b>	<b>35</b>
<b>Litteratur</b>	<b>36</b>
<b>Figurer</b>	<b>37</b>

# Indledning 1

---

Rundt om i hele verden, måles der i dag blodtryk. Hvorfor? Fordi blodtrykket kan fortælle meget omkring modstanden i en persons blodårer, hvilket kan være vigtigt, da forsnævret, sammentrukne eller overforkalket årer kan føre til hypertension (forhøjet blodtryk), der kan føre til f.eks. akut myokardieinfarkt. Der forskes nu til dags meget inden for blodtryk og hvad blodtrykket egentlig kan fortælle noget om. I forbindelse med forskning, udarbejdes der i dette semesterprojekt en blodtryksmåler til forskningsbrug. Ved et blodtrykssignal kan der detekteres systoliske og diastoliske værdier, der som baggrund kan benyttes af forskeren til at analysere blodtrykket. Formålet med dette projekt er at hjælpe forskningen inden for blodtryk.

Blodtryksmåleren skal kunne modtage en spænding fra en transducer, nulpunktsjustere ved opstart og kalibere efter forskerens ønske. Signalet skal vises i en graf, på et display, hvor værdier for puls, systoliske- og diastoliske tryk vises. Her starter forsker en måling og gemmer målingerne i en database, hvis forskeren ønsker det.

I Kravspecifikationen findes de krav, der er blevet stillet for projektet. Herunder er også de krav, som er blevet stillet mellem projektgruppen og vejleder. Under Systemarkitekturen findes informationer om, hvordan software- og hardwaredelen er designet, implementeret og testet.

Projektet har overvejende anvendt viden fra kurserne Sundhedsvidenskab, Kardiovaskulær Instrumentering, Programmering, Analog Signalbehandling, Digital Signalbehandling og Indledende System Engineering fra Ingeniørhøjskolen ved Århus Universitet. Sidstnævnte kursus har især været baggrund for arbejdspresserne og arbejdsmedoderne, som også kommer til udtryk i rapporten gennem diverse diagrammer og procesbeskrivelser.

Dette projekt består af to dele - en projektrapport og en projektdokumentation. Projektrapporten giver indblik i udarbejdningsprocessen af projektet og består af projektformulering, systembeskrivelse, anatomi, konklusion m.m. Her bliver projektvalg og -erfaringer omkring styrker og svagheder i arbejdspressen beskrevet. Projektdokumentationen giver indblik i baggrunden for, og tilblivelsen af projektet og slutresultatet.

## Versionshistorik

Version	Dato	Ansvarlig	Beskrivelse
0.1	04-11-2015	MHNK	Oprettelse af L <sup>A</sup> T <sub>E</sub> Xdokumenter
0.2	11-11-2015	ABH	Udviklingsværktøjer og krav
0.3	11-11-2015	DHC	Metoder
0.4	18-11-2015	ABH	Systembeskrivelse
0.5	19-11-2015	DHC	Perspektivering - Fremtidigt arbejde
0.6	24-11-2015	MHNK	Blodtryk
0.7	24-11-2015	AJF	Projektgennemførelse og -styring
0.8	24-11-2015	JMM	Projektformulering og afgrænsning
1.0	01-12-2015	MBA, MHNK	Referencelister i L <sup>A</sup> T <sub>E</sub> X
1.1	02-12-2015	MBA, MHNK	Figurlister i L <sup>A</sup> T <sub>E</sub> X
1.2	02-12-2015	MHNK	Resume
1.3	07-12-2015	MHNK	Abstract
1.4	09-12-2015	MHNK	Konklusion
1.5	09-12-2015	DHC	HW Systemarkitektur
1.6	11-12-2015	ABH	Tilrette krav og systembeskrivelse ift. endelig løsning
1.7	13-12-2015	MHNK	Indledning
1.8	13-12-2015	MBA	Opnåede resultater
1.9	13-12-2015	MHNK	Resultater og diskussion
2.0	15-12-2015	ABH	SW Systemarkitektur
2.1	15-12-2015	Alle	Korrekturlæsning

# Projektformulering og afgrænsning 2

---

I daglig klinisk praksis er der ofte behov for kontinuert at monitorere patienters blodtryk, i særdeleshed på intensive afdelinger, samt operationsstuer, hvor blodtrykket er en vigtig parameter i monitorering af patienters kardiovaskulære status.

Denne kontinuerlige monitorering er også nødvendig i forskningsverdenen. Det er i forskerens interesse, at kunne måle blodtrykket når der laves hæmodynamiske undersøgelser. Her skal det være muligt for forskeren at kunne aflæse det diastoliske tryk, systoliske tryk og pulsen, samt få vist en pæn kurve over blodtrykket. Det er målet, at opbygge en prototype, der kan registrere de spændinger, i millivolt (mV), der kommer fra transduceren og analogt forstærke samt filtrere signalet. Dette signal skal derefter konverteres til det digitale domæne.

Herfra skal der programmeres en brugergrænseflade, der fremfører disse målinger, samt gør det muligt, for forskeren, at gemme målingen i en database til senere brug. Resultatet bliver derfor et elektronisk kredsløb med forbindelse til et softwareprogram. For at det gemte data kan sammenlignes, kræver det at alt er blevet gemt med samme forudsætning, dvs. at alt data er gemt som rådata. Dette bliver håndteret i softwareneden, hvor beregninger implementeres. Når forskeren kigger på blodtryksgrafen, vil det filtreret signal vises. I tilfælde af at det er i forskers interesse at se på et ufiltreret signal, vil dette være muligt, ved et tryk på en knap.

## MoSCoW

### *Must*

- Et elektronisk kredsløb, som forstærker signalet fra transduceren og filtrerer det med ét indbygget analogt filter
- Et program til at vise blodtrykket som funktion af tiden. Programmet skal opfylde en række obligatoriske krav. Det skal kunne:
  - Programmeres i C#
  - Kunne kalibrere blodtrykssignalet og foretage en nulpunktsjustering
  - Vise blodtrykssignalet kontinuert
  - Kunne gemme de målte data i en database
  - Kunne filtrere blodtrykket i selve programmet via et digitalt filter, som skal kunne slås til og fra (monitor mode = filtreret og afrundet; signaldiagnose mode = råt signal med alle udsving, ufiltreret)
  - Afbildning af systolisk/diastolisk blodtryk med tal

*Could*

- Hardwaredelen skal bestå af ét Veroboard med påsatte komponenter

*Would*

- Alarmering, hvis blodtrykket afviger fra indbyggede grænseværdier
- Forskeren skal kunne hente de gemte data ned igen

**Ansvarsområder**

Idet gruppens størrelse ikke lægger op til samlet at arbejde på alle dele samtidig, er projektets ansvarsområder blevet fordelt som følgende:

<b>Navn</b>	<b>Ansvarsområder</b>
Ditte Heebøll Callesen	Hardware, dokumentation, rapport
Albert Jakob Fredshavn	Hardware, dokumentation, rapport
Martin Banasik	Hardware, dokumentation, rapport
Johan Mathias Munk	Software, dokumentation, rapport
Mette Hammer Nielsen-Kudsk	Software, dokumentation, rapport
Anne Hoelgaard	Software, dokumentation, rapport
<b>Navn</b>	<b>Skrevet afsnit i Rapport</b>
Ditte Heebøll Callesen	Metode, systemarkitektur (hardware) og perspektivering
Albert Jakob Fredshavn	Projektgennemførelse og projektstyring
Martin Banasik	Systemarkitektur (hardware) og opnåede resultater
Johan Mathias Munk	Projektformulering, afgrænsning og systemarkitektur (software)
Mette Hammer Nielsen-Kudsk	Resumé, abstract, indledning, blodtryk, resultater, diskussion og konklusion
Anne Hoelgaard	Systembeskrivelse, Krav, Systemarkitektur (software) og udviklingsværktøjer

# Blodtryk 3

---

## 3.1 Hvad betyder blodtryk?

Alle har på et tidspunkt i deres liv fået målt blodtryk, men hvad betyder det egentlig? Blodtryksmåling er en enkelt undersøgelse, der giver vigtige informationer om blodkarrenes og hjertets tilstand. Blodtryk kan måles både invasivt og ikke-invasivt. Dette projekt omhandler invasivt blodtryksmåling, som er en måling af trykket direkte i en blodåre. Blodtrykket måles i enheden millimeter kvisolv (mmHg). Et normalt blodtryk ligger på omkring 120/80 mmHg [1]. Hjertet pumper iltet blod ud i hele kroppen via arteriesystemet og sørger dermed for tilførslen af ilt og næringssubstanse til alle muskler og organer. De røde blodlegemer (erythrocytterne) er en vigtig bestanddel af blodet. Det er hæmoglobinet i de røde blodlegemer, som binder iltten og sørger for at iltten transportereres frem til vævene i kroppen. Når iltten er afgivet fra blodet til musklerne og andre væv, transportereres blodet tilbage til højre side af hjertet via venesystemet. Det af-iltede blod pumpes af højre hjertepumpekammer ud i lungerne, hvor blodet iltes på ny og derfra strømmer til venstre hjertehalvdel, for igen at blive pumpet ud i kroppen af venstre hjertepumpekammer.

Det høje tryk er det systoliske tryk, som kan måles når venstre ventrikkel trækker sig sammen. Det diastoliske blodtryk er blodtrykket i hjertets afslapningsfase (diastolen). Når hjertet trækker sig sammen (systolen) skaber det en trykbølge som forplanter sig ud igennem arteriesystemet. Trykbølgen kan identificeres som pulsen, der let kan mærkes f.eks. ved palpation af a. radialis ved håndleddet, hvor man mærker på årene. Trykket i venesystemet er meget lavere end i arteriesystemet, da blodet passivt skal strømme tilbage til højre forkammer, hvor trykket er lavt.

Venstre ventrikkel pumper iltet blod, under højt tryk, ud i aorta og arterierne. Disse blodårer er derfor tykvæggede og elastiske i modsætning til veneerne, der er ganske tyndvæggede, fordi de kun udsættes for et lavt tryk. Hjertet overfører, gennem systolen, energi til arterievæggen, som bruges i den resterende del af hjertets cyklus, til at presse blod gennem karsystemet.

For at forstå det følgende afsnit introduceres tre vigtige begreber nedenfor:

- Væskevolumen, der løber igennem et rør pr. tidsenhed, kaldes for væskestrømmen
- Distancen, som en væske flytter sig pr. tidsenhed er strømningshastigheden
- Blodvolumen, der løber gennem et væv pr. tids- og vægtenhed er gennemblødning

Der er en trykforskel imellem begyndelsen og slutningen af et rør. Væskestrømmen i røret afhænger af trykforskellen hen over røret og modstanden i røret. Dette kan udtrykkes i følgende

ligning som ligner Ohms lov for elektriske kredsløb:

$$Væskestrøm(Q) = \frac{Trykforskellen(\Delta P)}{Modstanden(R)} \quad (3.1)$$

Drivkraften for væskestrømningen (Q) er trykforskellen ( $\Delta P$ ) gennem røret. Hjertets kontraktioner gør, at strømmen i røret går fra et højere, til et lavere tryk. Modstanden (R) i en arterie er bestemt af bl.a. gnidningsmodstanden mellem arterievæggen og blodet, blodets viskositet og diameteren af arterien. Når blodet løber igennem arterierne, falder trykket efterhånden i blodet. Ved stigende modstand mod væskestrømmen forøges trykfaldet. Når modstanden i rørvæggen stiger, formindskes væskestrømningen, hvis trykforskellen samtidig ikke er steget.

Et rørs modstand bestemmes ud fra tre parametre:

- Længden af røret
- Den indre diameter på røret
- Viskositeten af væsken

Jo kraftigere hjertet pumper, desto større bliver trykforskellen og dermed blodstrømningen. Blodkarrets diameter er det, der har størst betydning for modstanden mod blodstrømmen. Hvis blodet presses igennem et snaevrt kar, er der en større del af blodet, der er tæt på karvæggen og blodet bliver derved bremset af friktionskraft. Modsat, hvis diameteren på karret havde været større, ville en mindre del af blodet være i kontakt med væggen og derved ville det ikke blive bremset lige så meget. Modstanden er derfor mindre og blodstrømningen større, i et stort kar. Blodets viskositet stiger, jo flere røde blodlegemer, der findes i blodet. Jo flere røde blodlegemer, desto højere viskositet. Hvis blodet har en høj viskositet og derved er tyktflydende, så skal der et større tryk til at holde en bestemt væskestrøm [1].

## 3.2 Hypertension

Hypertension er en meget almindelig lidelse, ca. 30% af den danske befolkning har forhøjet arterielt blodtryk [2]. Derfor er det vigtigt ofte at få målt sit blodtryk, da forhøjet blodtryk ikke kan mærkes og er den vigtigste årsag til hjerte-kar-sygdomme. Der er tale om hypertension når blodtrykket er 140/90 mmHg eller højere. Ved hypertension bliver arbejdsbelastningen af hjertet forøget, da der skal pumpes blod ud af hjertet mod en større modstand i arteriesystemet.

Forhøjet blodtryk gør at arbejdsbelastningen bliver større. Derved sker der lige så stille en fortykkelse af muskulaturen i venstre ventrikkel, da hjertet skal bruge flere kræfter på at pumpe blodet ud i aorta. Det øgede tryk påvirker blodkarrenes belastning, og kan medføre at mindre blodkar kan briste under det høje tryk. Hvis der er tale om blodkar i hjernen kan dette føre til en hjerneblødning. Hypertension er den hyppigste årsag til hjerneblødninger. Hypertension kan føre til en række andre komplikationer i form af åreforkalkning, hjerteinsufficiens [3], akut myokardieinfarkt, hjertekrampe, nyreskader og apopleksi.

Forhøjet blodtryk behandles med lægemidler, i form af blodtryksnedsættende medicin (antihypertensiva). Samtidig er non-farmakologiske metoder, som rygestop, motion, reduktion af saltindtagelse, vægttab og reduktion af alkoholforbrug vigtige i behandlingen af hypertension.

### 3.3 Hypotension

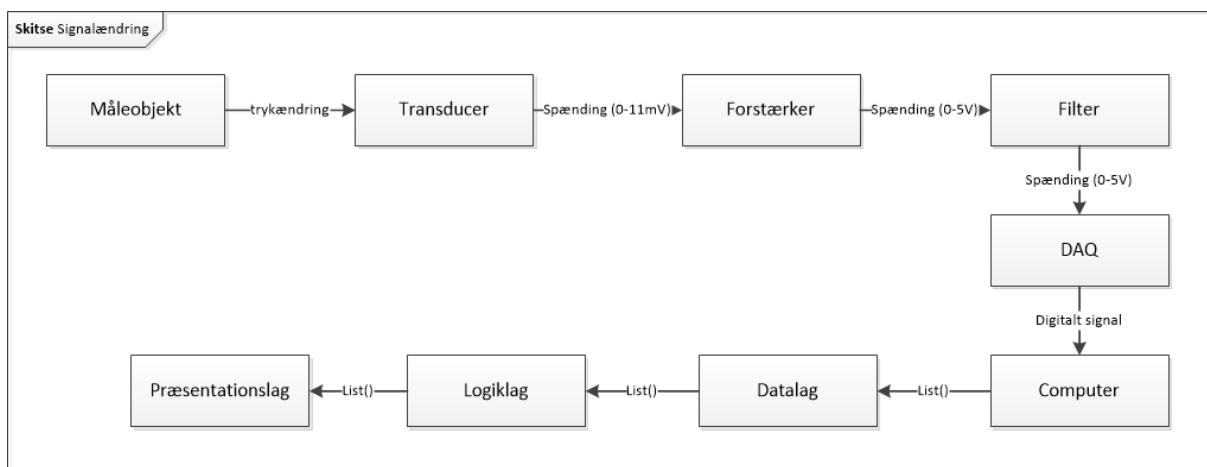
Hypotension er, modsat hypertension, et lavt blodtryk, og ikke nær så almindeligt. Der er tale om hypotension når blodtrykket er 90/60 mmHg. Hypotension ses især ved en række alvorlige akutte tilstænde som akut myokardieinfarkt, lungeemboli, sepsis eller alvorlige blødninger. Patienten kan i disse situationer være i en livstruende shock tilstand. Lavt blodtryk kan også i sjældne tilfælde forekomme kronisk, især ved sjældne stofskiftesygdomme med nedsat produktion af binyrebarkhormoner [4].

# Systembeskrivelse 4

Med udgangspunkt i projektformuleringen kommer dette projekts endelige system til at bestå af en software- og hardwaredel, der kan tilsluttes et måleobjekt, hvorpå et blodtryk kan måles. Systemet skal kunne implementeres i forskningsmiljøer, hvor en eller flere forskere ønsker at analysere målte blodtrykssignaler. Visionen er, at systemet skal være let tilgængeligt og effektivt, hvilket vil komme til udtryk ved, at systemet fungerer stabilt.

I dette projekt realiseres en prototype af systemet. Flere dele af systemet udvikles ud fra forsimplede metoder ift. hvordan det vil være optimalt at implementere dem i virkeligheden. Her tænkes på hardware- såvel som softwareelementer. Hardwaren i prototypen realiseres på et Veroboard, så det er muligt at tage den med sig, samt er mere holdbar over tid. Softwaren i prototypen består af flere moduler. Disse er opbygget efter principperne i en trelagsmodel. Dette er valgt for, at skabe et overblik over hvilke dele af software-koden, der har ansvaret for de enkelte funktionaliteter i systemet.

Hardwaren består af en forstærker og et filter. Forstærkerens opgave består i at forstærke det analoge signal fra maksimalt 11 mV til 5 V. Filteret sørger for at filtrere unødig støj fra det analog signal. Signalændringen fra måleobjektet til visning af signal på en graf er skitseret herunder. Det skal pointeres at dette kun er en skitse for at skabe overblik, derfor er flere processer i softwaren udeladt af diagrammet.



Figur 4.1: Skitse af signalændring

Database-laget består af en lokal database, samt indhentningen af blodtrykssignalet fra hardwaren. I den lokale database gemmes det indhentede blodtrykssignal i en tabel. Signalet gemmes med et tidsstempel, samt under et filnavn sammensat af et autogenereret Id og Forsøgsnavn, som forskeren indtaster på brugergrænsefladen ved begyndelsen af en måling.

Logik-laget er handlingslaget, og alt kommunikation til de resterende lag går gennem dette lag. Laget indeholder flere klasser, der indeholder metoder til indhentning af systoliske, diastoliske og puls værdier, ud fra det indhentede blodtrykssignal. Derudover indeholder laget også klasser, der har ansvaret for at foretage en filtrering af signalet når dette er valgt.

Præsentations-laget er forskerens vej ind i systemet, dette lag har til ansvar, at udskrive valgt data på brugergrænsefladen og at registrere tryk på knapper.

Systemet skal uadtil have en brugergrænseflade i form af en touch skærm eller almindelig computerskærm med tilhørende tastatur. Det er denne skærm som den primære aktører, forskeren, interagerer med. Det tilstræbes, at opbygge brugergrænsefladen simpelt og efter forskerens logik, så opbygningen giver mening for systemets bruger. Efter indhentning af blodtrykssignal er systemet i stand til grafisk at vise signalet kontinuerligt, samt udskrive blodtrykssignalets systoliske, diastoliske og puls værdier.

# Krav 5

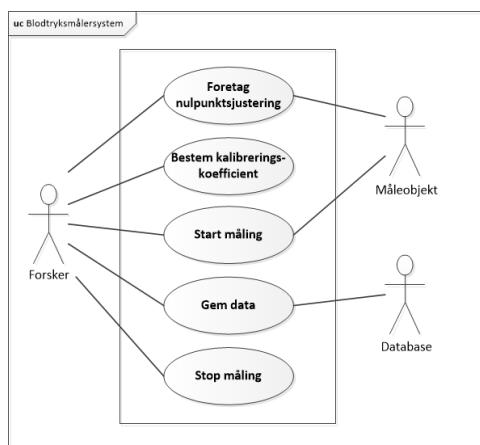
Til projektet er der opstillet krav, som er formuleret i projektoplægget. Disse funktioner skal implementeres i produktet. Kravene lyder på, at der skal udvikles et system, som kan tilsluttes et væskefyldt kateter, samt vise en blodtrykskurve på en skærm. Mere detaljeret vil det sige, at systemet skal indeholde to elementer. Først et elektronisk kredsløb, som forstærker signalet fra transduceren og filtrerer det med et indbygget analogt filter. Derefter et program til at vise blodtrykket, som funktion af tiden.

Yderligere skal dette program opfylde kravene:

- Programmeres i C#
- Kunne kalibrere blodtrykssignalet og foretage en nulpunktsjustering
- Vise blodtrykket kontinuert
- Kunne lagre de målte data i en tekstfil eller en database
- Kunne filtrere blodtrykket via et digitalt filter, denne funktion skal kunne slås til og fra

På baggrund af disse krav er der opstillet fem Use Cases, der tager højde for disse krav, samt beskriver aktørens interaktion med systemet. Disse Use Cases benyttes som kravspecifikation, der har til formål at specificere, hvilke krav der stilles til projektet. Udover ovennævnte krav vil der også blive arbejdet hen imod at systemet skal afbillede puls, det systoliske blodtryk og det diastoliske blodtryk med tal.

Kravene opstilles ud fra kundens ønsker, samt leverandørernes mulighed for realisering. Den fulde beskrivelse af hver enkelt Use Cases (fully dressed Use Cases) findes i Dokumentationen.



Figur 5.1: Use Case diagram

## Aktørbeskrivelse

Use Case diagrammet, på figur 5.1, viser de tre aktører: Forsker, Database og Måleobjekt. Herunder er der en detaljeret beskrivelse af hver aktør.

**Forsker** er en primær aktør. Det er denne aktør, som foretager blodtryksmålingen, kalibrer og fortager nulpunktsjustering. Målingerne for blodtrykssignalet vises på displayet, som forskeren har tilgang til.

**Måleobjekt** er en sekundær aktør. Måleobjektet kan bestå af In Vitro, patient eller andet, som kan skabe et blodtrykssignal.

**Database** er en sekundær aktør. Denne aktør er en Database, hvori blodtrykssignalets rådata gemmes. Ligeledes gemmes det indtastede Forsøgsnavn og det autogenererede Id.

## Use Case beskrivelse

Use Case diagrammet viser ligeledes de fem Use Cases, der er for systemet: Foretag nulpunktsjustering, Bestem kalibreringskoefficient, Start Måling, Gem data, Stop måling. Disse Use Cases beskriver interaktionen mellem aktørerne og systemet. Herunder er der en kort beskrivelse formålet med hver Use Case.

**UC1: Foretag nulpunktsjustering** Når systemet startes op vil det første der møder forskeren være en GUI, hvorfra nulpunktsjustering kan foretages. Det forudsættes at forskeren har åbnet for transduceren, så den modtager atmosfærisk tryk inden at forskeren trykker på Foretag-knap. Systemet indhenter så en nulpunktsjusteringsværdi, denne værdi gemmes i softwaren og alt indhentet signal herefter vil dermed være indstillet til en offset på nul. Herefter går systemet videre til næste Use Case.

**UC2: Bestem kalibreringskoefficient** Kalibreringen foregår uafhængig af om systemet kører. Forsker tilslutter hardware til væskesøjle ved 50 mmHg. Outputspænding aflæses fra hardwaren. Kalibreringskoefficienten kan så bestemmes ved en simpel beregning ud fra tryk og outputspænding. Denne værdi indtastet i en XML-fil, hvorfra softwaren kan tilgå koefficienten og derefter benytte den som omsætningskoefficient mellem volt og mmHg. Dermed er en kalibrering udført.

**UC3: Start måling** Det kræves at Måleobjektet, hvorpå blodtrykssignal ønskes fra, er tilsluttet. Derfor tilslutter Forskeren transduceren til Måleobjektet. Derpå kan målingen startes ved, at forskeren trykker på knappen Start Måling. Herefter indhenter systemet blodtrykssignalet, som bliver udskrevet på displayet. Værdier for puls, systolisk og diastolisk blodtryk udskrives på displayet.

**UC4: Gem data** Det er muligt for Forskeren at gemme det indhente blodtrykssignals rådata, indenfor en periode, valgt af Forskeren. Dette gøres ved, at Forskeren trykker på knappen Start Gem. Systemet vil herefter begynde at gemme rådata i en liste. Dette vil systemet blive ved med at udføre indtil Forskeren trykker på knappen Stop Gem. Herefter gemmes listen i Databasen. Ved tryk på Stop Gem vises filnavn på displayet.

**UC5: Stop måling** Det er muligt for Forskeren at stoppe visning af blodtrykssignalet. Dette gøres ved, at Forskeren trykker på knappen Stop Måling. Systemet vil herefter lukke forbindelsen

til indhentning af data og grafen vil fastholdes.

### **Ikke-funktionelle krav beskrivelse**

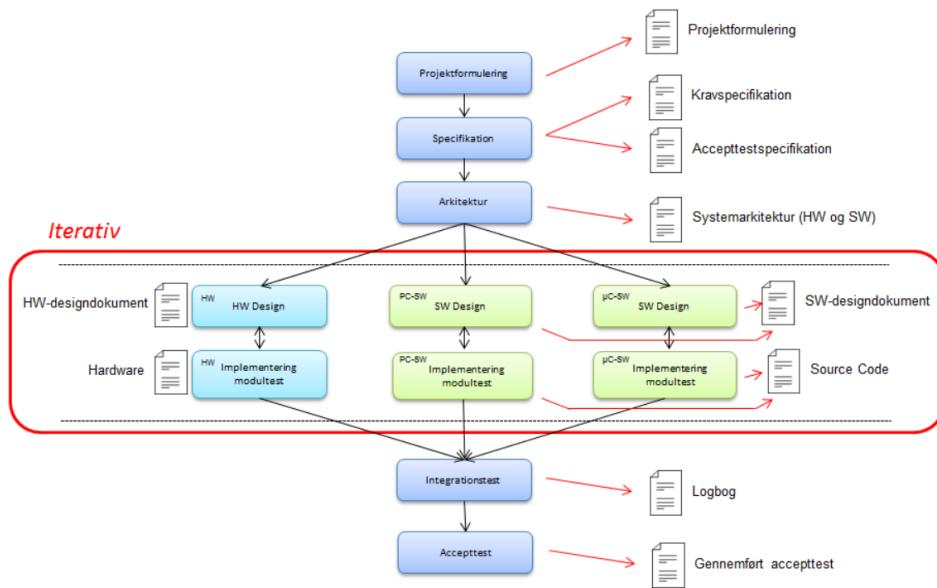
Ikke-funktionelle krav er struktureret efter (F)URPS+, hvor krav til systemets funktionalitet, brugervenlighed, pålidelighed, præsentation samt vedligehold er beskrevet. Disse krav er primært softwarekrav. Der opstilles bl.a. et krav om en maksimal tid, der må gå fra, at der er trykket på en knap, til at systemet reagerer. Når der er trykket på en knap, skal systemet foretage den ønskede proces. Eksempelvis, ved tryk på Stop Gem-knappen, skal systemet sende rådataen til databasen, hvor dataen bliver gemt. Ligeledes er opbygning af display også en del af de ikke-funktionelle krav.

# Projektbeskrivelse

6

## 6.1 Projektgennemførelse

Dette projekt er gennemført vha. forskellige udviklingsprocessor, hvilket er med til at sikre kvalitet, og at deadlines overholdes. En af disse modeller er ASE-modellen [5]. Denne model er en udviklingsmodel, der er udarbejdet af Aarhus Ingeniørhøjskole. Modellen er en gentagelig udviklingsproces drevet ud fra projektets Use Cases. Modellen er benyttet på den måde, at gruppedelmedlemmerne fastlægger en projektformulering, kravspecifikation og systemarkitektur, hvor de enkelte hardware- og softwaredele designes, implementeres og testes. Gennem en integrationstest ses det om hardware- og softwaredelene fungerer sammen som de skal. Dette ender med en gennemført accepttest, således at det testes om systemet lever op til kravene.

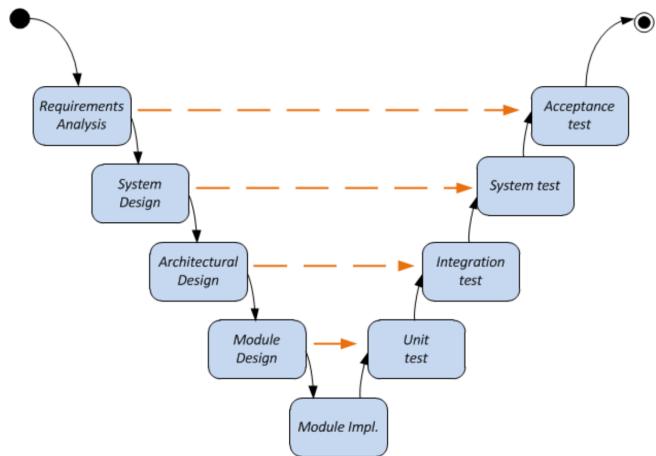


Figur 6.1: ASE-modellen

For at forstå ASE-modellen [5] er det vigtigt at gennemgå Use Cases; et værktøj, som skal beskrive interaktioner mellem aktører og selve systemet. Sammen med de ikke-funktionelle krav opnås et overblik over hvilke funktionalitetskrav, der stilles til systemet. På baggrund af kravspecifikationen kan accepttesten efterfølgende udarbejdes. I dette projekt er hardware- og softwaredesign på lige fod, da projektet består af begge ting ligeligt.

V-modellen [5] er en faseopdelt udviklingsmodel, der også er værd at nævne i dette projekt. Den beskriver udviklingsfaserne og testfaserne sideløbende i forhold til projektet, og den er

derfor benyttet til dette projekt sideløbende med ASE-modellen. Modellen fungerer således, at specifikationen af test foregår parallelt med udviklingen af selve systemet.



Figur 6.2: V-model

Den er blevet benyttet til hardware- og softwareudviklingen. Hardware og software skulle begge teste deres funktioner inden nye faser blev igangsat, for at verificere om disse funktioner virkede korrekt. Fordelen ved at teste på forskellige niveauer er, at det skal sikre de udviklede delsystemer, således at de virker som planlagt. Det er vigtigt, at hver fase er udført, før den næste fase påbegyndes.

Ved softwareudviklingen har det været svært at følge den stringenteste linje ved V-modellen gennem projektet. Derfor bærer softwareudviklingen mere præg af vandfladsmodellen. Hvor der først er påbegyndt en ny fase, efter den foregående er færdiggjort.

## 6.2 Projektstyring

Projektet er udarbejdet over et semester, hvor undervisningen og forelæsningerne delvist har udgjort grundlaget for teorien benyttet i projekt. Der blev i starten udarbejdet en tidsplan med enkelte faste deadlines og med mulighed for at sætte andre deadlines løbende i processen. Se tidsplanen i Bilag nr. 1.

Projektgruppen består af 6 gruppemedlemmer, som har været delt i to fokusområder, hardware og software. Fordelingen blev udarbejdet efter de enkeltes ønsker. Projektet har været afhængig af, at der har været god kommunikation mellem undergrupperne. Da gruppen har været opdelt, har der været projektmøde hver uge, hvor gruppen har opdateret hinanden og vejleder. Samtidigt har gruppen forsøgt at sidde samlet, for på den måde at kunne opdatere hinanden løbende.

Under projektet har alle medlemmer været med til, at sikre en administrativ kæde af deadlines til individuelle opgaver. Disse deadlines har sikret at opgaverne er blevet opfyldt til møderne og det har derfor været nemt at følge op på.

### 6.3 Metoder

Til at kunne overskue arkitektur og designet af projektet, er flere forskellige arbejdsmetoder benyttet for at skabe det bedst mulige resultat. For at finde, hvad blodtryksmåleren skal gøre, er der blevet udarbejdet Use Cases. Disse beskriver systemet funktionalitet. Use Cases viser, hvad brugeren skal opleve fra systemet, men ikke, hvordan det sker. I Use Case diagrammet bliver det vist, hvilke aktører der findes og hvordan de interagerer med systemet.

I projektet bruges accepttest til at teste blodtryksmåleren. Dette gøres ud fra kravspecifikationen, hvor det er angivet, hvilke krav der er stillet til systemet.

Accepttesten er en test, hvor det beskrives, hvad der skal ske og hvad brugeren skal gøre. Testen er for at undersøge om produktet opfylder de krav, der er blevet sat for det. Accepttesten giver et godt overblik for leverandøren og kunden, der nemt og hurtigt kan se om produktet virker som det skal.

Til beskrivelse af software- og hardwaredesign er diagrammer og skemaer blevet udarbejdet i SysML og UML. SysML er et grafisk modelleringsprog, som kan bruges til at overskueliggøre systemer.

Til software er der blandt andet lavet en applikationsmodel i SysML, som består af et domæne-, klasse- og sekvensdiagram. Domænemodellen viser sammenhængen mellem blokkene i systemet. Blokkene findes i Use Casene og derved bliver disse to ting koblet sammen.

Klassediagrammet viser, hvilke metoder blokkene har og hvordan de kommunikerer med hinanden. Her findes domæne-, kontrol- og grænsefladeklasser. Kontrolklasserne beskriver, hvordan data behandles mellem domæne- og grænsefladeklasser. Domæneklasser indeholder funktionalitet fra den pågældende softwareblok. Grænsefladeklasserne viser, hvordan systemet interagerer med omverdenen. Diagrammet gør det nemmere at fremme en lav kobling og høj samhørighed i softwaren.

Sekvensdiagrammet fortæller, hvad der sker i selve koden. Ingen går det ud fra Use Casene, hvor vægten nu er på softwaredelen. Derved beskrives det, hvordan metoder bliver kaldt og hvordan de forskellige klasser interagerer. Hver Use Case skal her gennemgås i softwaren, så der skabes et overblik over vejen gennem koden.

For at skabe overblik og indsigt i koden, er der i UML udarbejdet et aktivitetsdiagram og et klassediagram. Aktivitetsdiagrammer går i dybden med en specifik metode. Det er kun blevet gjort for relevante metoder. Her tydeliggøres det, hvordan hver metode fungerer og hvad den indeholder. Se Dokumentation under Software, Implementering. Klassediagrammet fortæller hvilke metoder, en klasse indeholder og hvordan klasserne hænger sammen.

Til hardwaren er der blevet brugt Block Definition Diagram (BDD), som viser hvilke blokke et system indeholder og hvilke porte de har. BDD er lavet til at give et overblik over systemet. Ud fra BDD'et er et Internal Block Diagram (IBD) lavet. Her vises, hvilke signaler, som findes i systemet og hvordan de sendes rundt. Her vises portene igen. Der skal være overensstemmelse mellem BDD og IBD.

Til udarbejdelsen af kredsløb blev Analog Discovery brugt til at simulere signalet, som i sidste ende skal komme fra transduceren. Først blev kredsløbet opbygget på et fumlebræt, hvor det blev testet for at afprøve om det lever op til kravene. Når det opfylder kravene, flyttes det over på et Veroboard. Veroboardet bliver igen testet før aflevering.

## 6.4 Systemarkitektur

I det følgende ses design, implementering og test for hhv. hardware og software.

## 6.5 Hardware

I hardwaredelen skal der ligge en forstærkning og et lavpasfilter. Det differentieret signal fra transduceren skal forstærkes og filtreres før det kan sendes ind i DAQ'en.

### 6.5.1 Design

#### Forstærkning

Transduceren mäter en trykændring, som den omsætter til en spænding. Dette er udtrykt ved et differentieret signal, som sendes ind i Forstærker-blokken.

Signalet fra transduceren er en lav spænding, som skal forstærkes op, for at passe med DAQ'ens input. Denne forstærkning udregnes ud fra det maksimale output fra transduceren og det maksimale input til DAQ'en. Se beregningerne under Implementering.

Under simulering bruges Analog Discovery som en funktionsgenerator, der simulerer det differentieret signal.

#### Lavpas

I projektet skal der laves et 2. ordens lavpasfilter. Filteret skal laves for at sikre, at der ikke opstår aliasering og til filtrering.

Aliasering [6] er, hvor signalet bliver gentaget. Når man har signalet i det digitale domæne, bliver spektret for signalet en periodisk funktion.

Lavpasfilteret skal være et Sallen-Key Butterworth-filter med en knækfrekvens på 50 Hz og en samplingsfrekvens på 1kHz. Ud fra oplysninger givet til projektet, vides det at filteret skal dæmpe signalet med 20 dB pr. dekade, under antagelse af at den forekommende støj er mindre end signalet, også når det forekommer over knækfrekvensen. Derfor oplyses filteret til at være 50 Hz, da dette giver en minimum dæmpning på 20 dB pr. dekade.

### 6.5.2 Implementering

#### Forstærkning

For at få den rette forstærkning er det blevet valgt, at benytte instrumentationsforstærkeren INA114. Her kan transduceren sættes på med det differentierede signal. INA114 er valgt da følgende gælder[7] for instrumentationsforstærkere:

- Differentielt input - single ended output
- Gain justering med ændring af kun én modstand
- Meget høj indgangsimpedans
- Stor Common Mode Rejection Ratio (CMRR)

For at udregne den korrekte forstærkning, bruges følsomheden fra transduceren og eksitations-spændingen. Først udregnes det maksimale output fra transduceren til 11.25 mV.

Da det er besluttet, at det maksimale input til DAQ'en [6] er 5 V, kan forstærkningen (Gain) nu udregnes:

$$\begin{aligned} 5V &= 11.25mV \cdot G \\ G &= 444.44 \end{aligned} \quad (6.1)$$

For at få den rette forstærkning udregnes den eksterne modstand ( $R_g$ ) til INA114 [8]. INA114's forstærkning afhænger af størrelsen på  $R_g$ , hvis modstanden er stor, er forstærkningen lille og omvendt.  $R_g$  udregnes ved formlen:

$$\begin{aligned} G &= 1 + \frac{50k\Omega}{R_g} \\ 444.44 &= 1 + \frac{50k\Omega}{R_g} \Rightarrow R_g = 112.75\Omega \end{aligned} \quad (6.2)$$

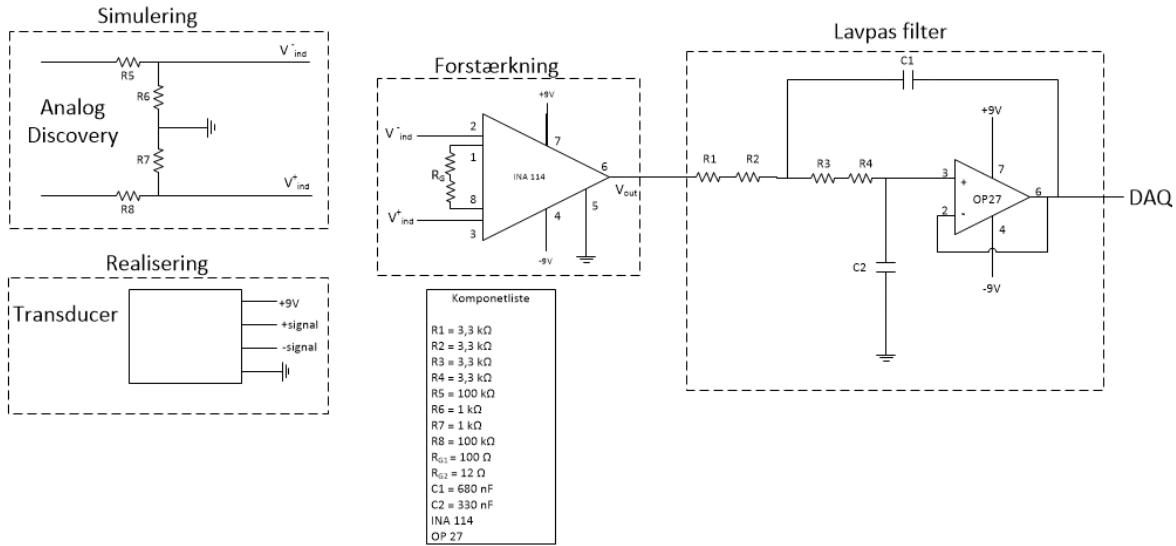
Derved fås en værdi for den eksterne modstand til INA114, som skaber den ønskede forstærkning.

Den ønskede forstærkning kan bruges, da det passer overens med båndbredden. Dette kan undersøges da produktet af forstærkning og båndbredde er konstant og båndbredden skal ligge over knækfrekvensen for filteret. Se beregning i Dokumentation ligning 3.4.

For at imødekommme usikkerheden ved Analog Discovery, når den arbejder ved lave spændinger, laves et kredsløb efter spændingsdelerprincippet. Signalerne fra Analog Discovery skal sendes igennem dette kredsløb, hvor de efter spændingsdelerprincippet gøres mindre.

Derved kan Analog Discovery sende signaler med en højere spænding ind i kredsløbet og usikkerheden mindskes. Hvis INA114 skal have 11.25mV skal Analog Discovery sende 1.1352V ind. Se i Dokumentationen under Hardware, Implementering for flere informationer om spændingsdelerkredsløbet.

På figur 6.3 ses et diagram af det endelige kredsløb med komponentværdier. Her ses, hvordan det ser ud ved realiseringen med transduceren og under simuleringen ved Analog Discovery.



Figur 6.3: Diagram over HW

### Lavpas

For at opnå den ønskede effekt i lavpasfilteret, blev det oplyst at  $f_c = 50$  Hz,  $f_s = 1\text{kHz}$ ,  $R_1 = R_2$  og  $C_2 = 680\text{nF}$ . Ud fra disse værdier, udregnes de resterende komponentværdier for filteret.

Overføringsfunktionen for et 2. ordens filter er:

$$H(z) = \frac{\omega_n^2}{(s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2)} \quad (6.3)$$

For at finde overføringsfunktionen for det gældende system, vides det at følgende ligninger gælder [9]:

$$\begin{aligned} \omega_n &= 2 \cdot \pi \cdot 50 = \frac{1}{\sqrt{R_1 \cdot R_2 \cdot C_1 \cdot C_2}} \\ 2 \cdot \zeta \cdot \omega_n &= \frac{1}{C_2} \cdot \left( \frac{R_1 + R_2}{R_1 \cdot R_2} \right) \end{aligned} \quad (6.4)$$

Dette indsættes i den generelle overføringsfunktion og det simplificeres, blandt andet ved at det vides at  $R_1 = R_2$ . Se Bilag nr. 9 for nærmere udregninger:

$$H(z) = \frac{\frac{1}{C_1 \cdot C_2 \cdot R^2}}{s^2 + s \cdot \frac{2}{R \cdot C_2} + \frac{1}{C_1 \cdot C_2 \cdot R^2}} \quad (6.5)$$

Når der arbejdes med et 2. ordens Butterworth filter, vides det at udsvinget  $\zeta$  skal have værdien 0.7 [10]. Under beregningerne, var der usikkerhed omkring, hvad værdien af  $\zeta$  skulle være. Da kredsløbet skulle realiseres og dokumenteres, blev det derfor overvejet at ændre samtlige komponentværdier så de passede med en  $\zeta = 1$ . Inden dette blev gjort blev det dokumenteret at det gælder at  $\zeta$  skal have en værdi på 0.7.

Den sidste overføringsfunktion sammenlignes med den generelle for 2. ordens systemer. Det gælder at  $C_2 = 680 \cdot 10^{-9}\text{nF}$ . Det er muligt at isolere forskellige led. Først isoleres der for modstanden

(midterste led i nævneren) og den udregnes til  $R = 6687\Omega$ .

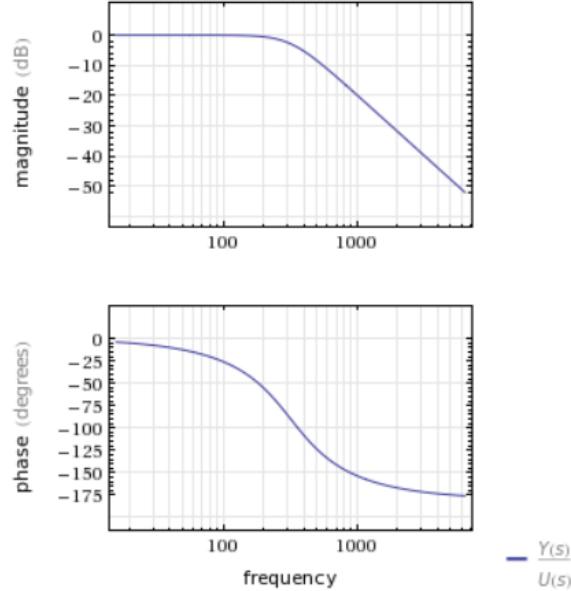
Nu kan det tredje led i nævneren isoleres og kondensatoren  $C1$  udregnes til  $C1 = 333 \cdot 10^{-9}nF$ .

Nærmere beregninger kan ses under Systemarkitektur i Dokumentationen.

Alle komponentværdierne til lavpasfilteret er fundet og det kan realiseres.

Under udviklingen af lavpasfilteret er komponentstørrelserne blevet ændret for at kunne realisere det. De brugte komponentværdier er:  $R = 6.6k\Omega$ ,  $C1 = 330 \cdot 10^{-9}nF$  og  $C2 = 680 \cdot 10^{-9}nF$ . For at være sikker på at filteret har de ønskede karakteristika, laves et bodeplot for den endelig overføringsfunktion:

$$H(z) = \frac{62500000000}{610929 \cdot \left( s^2 + \frac{250000}{561} \cdot s + \frac{62500000000}{610929} \right)} \quad (6.6)$$



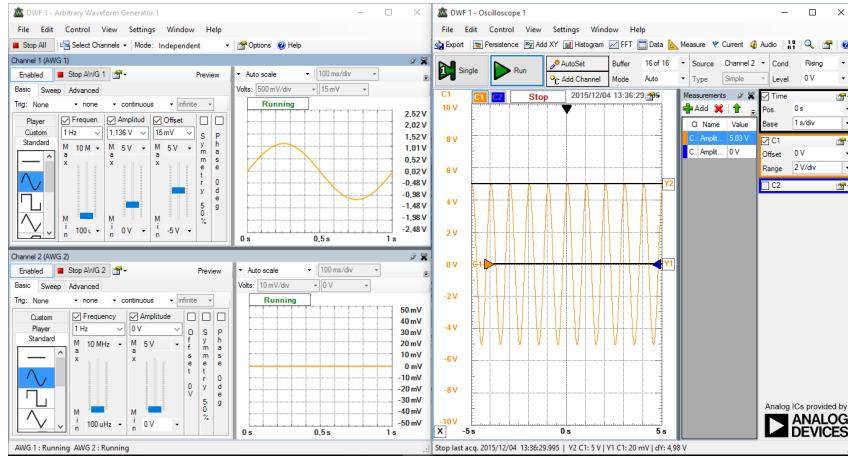
Figur 6.4: Bodeplot

Ud fra den nye overføringsfunktion udregnes en nu  $\zeta$  for at kontrollere at værdien ikke har ændret sig. Denne udregnes til  $\zeta = 0.709$ . Derfor kan det konkluderes at filteret stadig har den ønskede funktionalitet.

### 6.5.3 Test

#### Forstærkning

For at teste forstærkningen, sendes et differentieret signal ind vha. Analog Discovery. Her observeres det hvor meget signalet bliver forstærket. På figur 6.5 ses det signal, som sendes ind i Forstærker-blokken og det, der måles på udgangen af blokken.



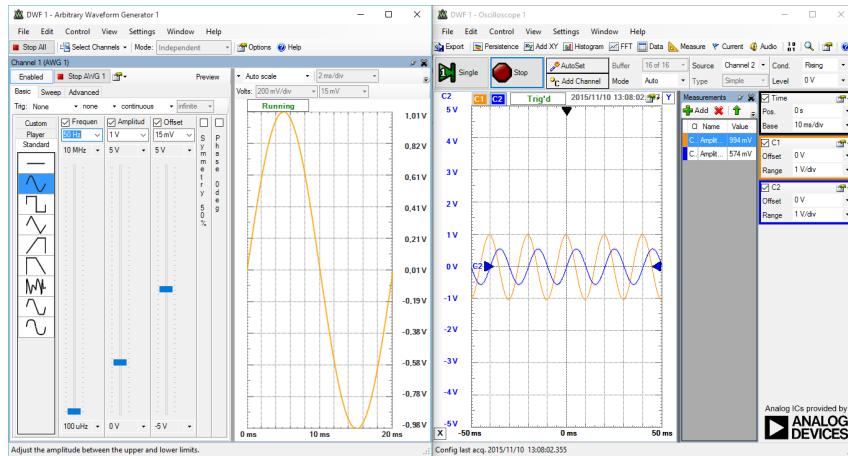
Figur 6.5: Forstærkningsblok

Der sendes et differentieret signal ind i INA114. På udgangen ses det at signalet er blevet forstærket op til 5 V DC. Herved er maksimumsinput til Forstærker-blokken blevet forstærket så det passer med maksimumsinput til DAQ'en. Signalet bliver ikke ændret på andre måder i denne blok.

### Lavpas

For at teste lavpasfilteret foretages målinger med en sinus, hvor frekvensen varierer for hver maling. Derved aflæses fasen mellem ind- og udgangssignal, samt amplituden for hver maling. Der laves flere målinger både før, på og efter knækfrekvensen. Ved knækfrekvensen skal fasedrejningen være 90°. Dette kan aflæses på figur 6.6.

Se Hardware, Modultest i Dokumentation for mere information.

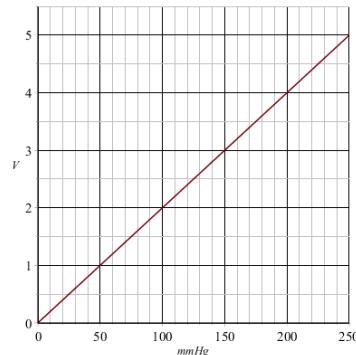


Figur 6.6: Måling for 50 Hz

### Kalibrering med væskesøjle

Efter forstærkning og lavpasfilteret er blevet testet hver for sig, udføres en kalibrering af systemet vha. en væskesøjle. Her bruges en udleveret væskesøjle med tre målepunkter, hvor det er angivet hvor højt trykket (målt i millimeter kviksølv, mmHg) er ved hvert af disse punkter. Derved kan det testes om hardwaren måler den rigtige spænding i forhold til mmHg.

Ud fra den maksimale spænding (målt i Volt, V) og mmHg kan det udregnes, hvad hardware skal vise ved 100 mmHg, se figur 6.7.



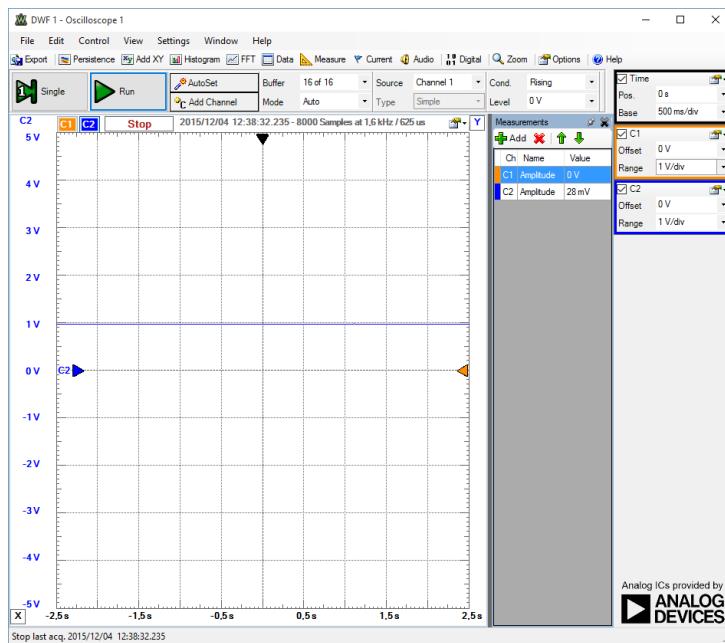
Figur 6.7: Graf til kalibrering, fra udregninger

Testen udføres ved at fylde vand i søjlen til markeringen. Transduceren skal være tilkoblet et målepunkt, mens de andre er lukket til. Transduceren er sat til hardwaren, der hvor Analog Discovery tidligere har været sat til. Transduceren er tilkoblet 9 V ved batterierne. På samme måde, som ved simuleringen, aflæses målingen på computeren ved hjælp af programmet WaveForms. Da det vides hvilken trykændring der måles på, vides det fra grafen til kalibreringen, hvilken spænding den skal vise. Dette foretages for de tre målepunkter på væskesøjlen, hvor hver måling sammenlignes med den udregnede graf. For hver måling, skal transduceren flyttes til et af de andre målepunkter.



Figur 6.8: Opstilling

Ud fra grafen i figur 6.7 vides, hvad svaret til hver måling skal være. På figur 6.9 ses målingen, da transduceren var tilkoblet målepunktet for 50 mmHg. Ud fra figur 6.7 vides det at målingen skal vise 1 V DC.



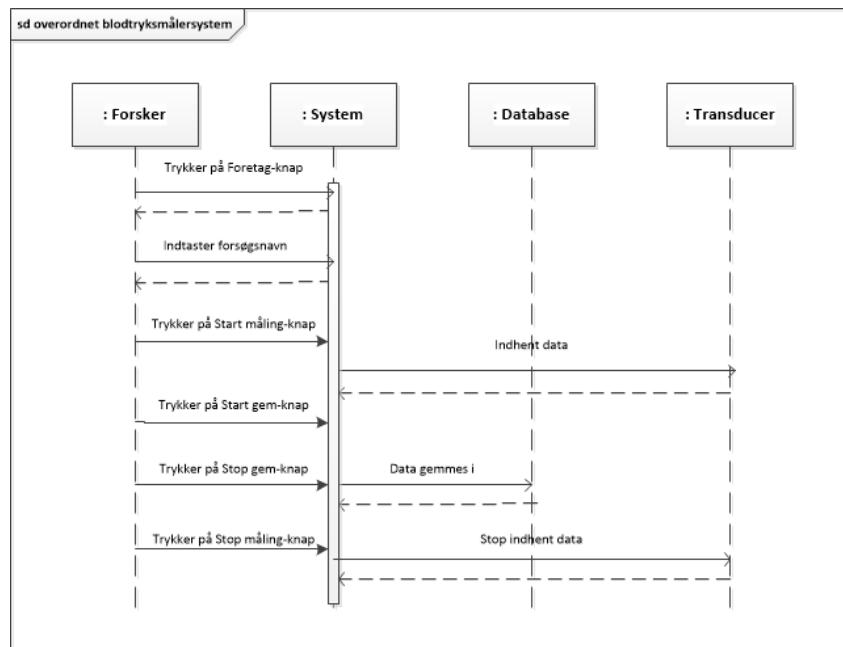
Figur 6.9: Måling ved 50 mmHg

Ud fra figur 6.7 kendes udgangsspændingen også for de to andre målepunkter. Se under Hardware, Modeltest i Dokumentation for billeder af målingerne ved 10 mmHg og 100 mmHg. Målingen for 50 mmHg er valgt ud, da denne måling ligger til baggrund for kalibreringen i softwaren.

## 6.6 Software

### 6.6.1 Design

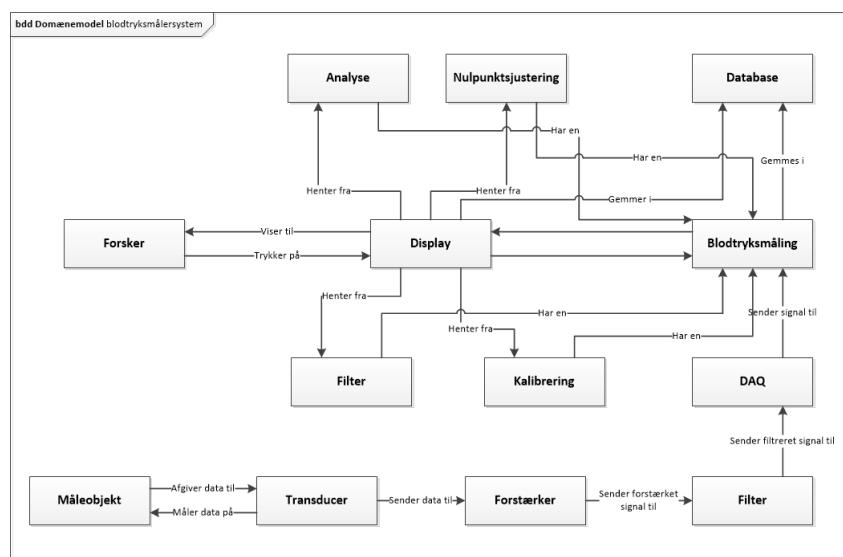
Overordnet set ønskes det at udvikle et system, der kan interagerer med en Forsker. Diagrammet herunder viser at Forskerens opgave består i at starte en måling, foretage en nulpunktsjustering og gemme de ønskede rådata, samt uafhængigt af systemet at bestemme en kalibreringskoefficient, hvis det ønskes. Diagrammet er en simpel illustration, som viser systemets adfærd gennem alle fem Use Cases. For yderligere præcisering af Use Cases se Kravspecifikation i Dokumentation. Formålet med dette diagram er at skabe et overblik over det samlede system. I de efterfølgende diagrammer dækker Transducer-blokken over alt hardware og DAQ'en.



Figur 6.10: Overordnet sekvensdiagram for systemet

### Problemidentifikation

Første step i softwaredesignet er at klarlægge, hvilke klasser systemet skal bestå af. Til dette er en domænemodel udarbejdet med udgangspunkt i de fem Use Cases. Modellen har til formål at vise, hvilke dele systemet skal holde styr på. Systemet vil primært centrere sig omkring blodtryksmålingen, da det er den, der skal analyseres og bearbejdes, samt omkring display hvorfra Forskeren får vist målingerne. Dette kan ses på figur 6.11.

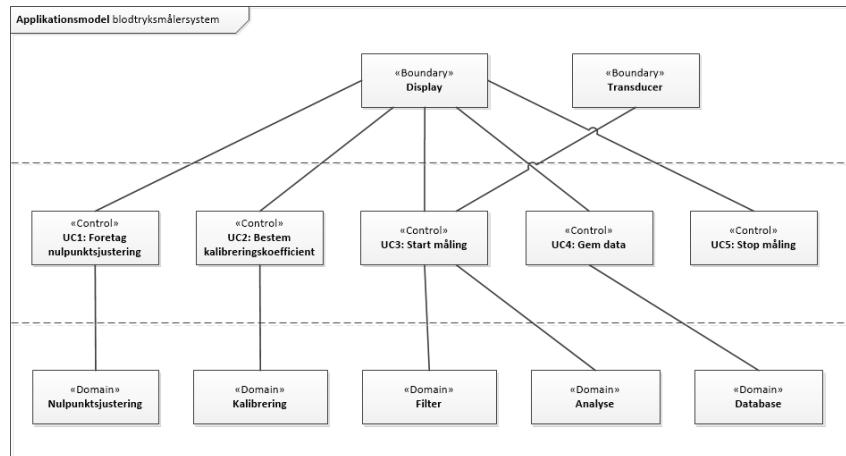


Figur 6.11: Domænemodel

Analyse dækker over bestemmelse af systoliske, diastoliske og puls værdier. Hardwarekomponenterne er medtaget for at vise signalets vej fra måleobjekt til systemet.

## Klasseidentifikation

Domænemodel leder hen til udarbejdelse af en applikationsmodel. Denne model har til formål at vise hver enkelt klasses individuelle formål, samt hvilke domæne og boundary klasser der kommer i spil ved den pågældende Use Case.



Figur 6.12: Applikationsmodel for software

## Metodeidentifikation

Klasserne i applikationsmodellen er med til at definere, hvilke blokke sekvensdiagrammerne må indeholde. Dermed også med til at definere hvilke metoder i softwaresystemet, der er nødvendige for at få udført de ønskede handlinger mellem blokkene. Sekvensdiagrammerne for hver enkel Use Case kan ses i Dokumentation under Software, Design.

### 6.6.2 Implementering

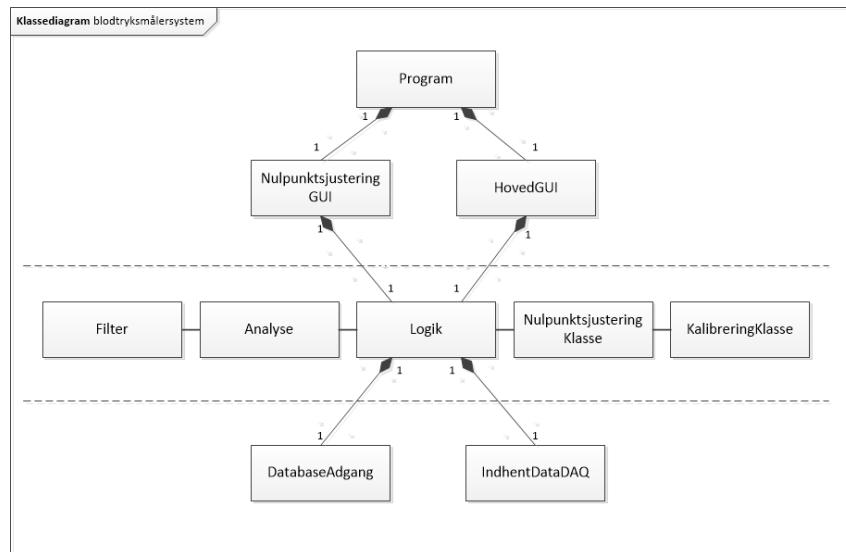
På baggrund af designfasen for softwaren kan implementeringen påbegyndes. Softwaredesignet viser at systemet skal implementeres med en GUI-applikation, hvor aktøren kan interagere med systemet. Derudover er det kendt at softwaren skal indeholde en række klasser, hvori funktionaliteter som kalibrering, nulpunktsjustering, digitalt filter og indhentning af systolisk, diastoliske og puls værdier skal placeres. I det følgende beskrives de overvejelser, der er gjort i forhold til implementering af disse funktionaliteter og softwaresystemet.

Implementeringen af softwaren sker i Visual Studio 2013 i sproget C#. Dette er valgt da programmet er et godt værktøj til at arbejde med GUI-applikationer, samt til håndtering af tråde og trådkommunikation. Tråde benyttes i softwaren da systemet, der skal implementeres, er et eventdrevet system. Det vil sige, at systemet skal kunne håndtere mange handlinger på en gang. Handlingerne igangsættes af events, der kommer af aktørens interaktion med systemet. Trådkommunikationen fungerer således, at en tråd kan sende et signal ud, som andre tråde kan reagere på.

## Klasse implementering

På baggrund af designmodellerne er det besluttet at opbygge systemkoden efter principperne i en trelagsmodel[11]. Trelagsmodellen indeholder et præsentations-lag, et logik-lag og et data-lag. Al kommunikation i trelagsmodellen skal gå igennem logik-laget. Fordelen ved trelagsmodellen er,

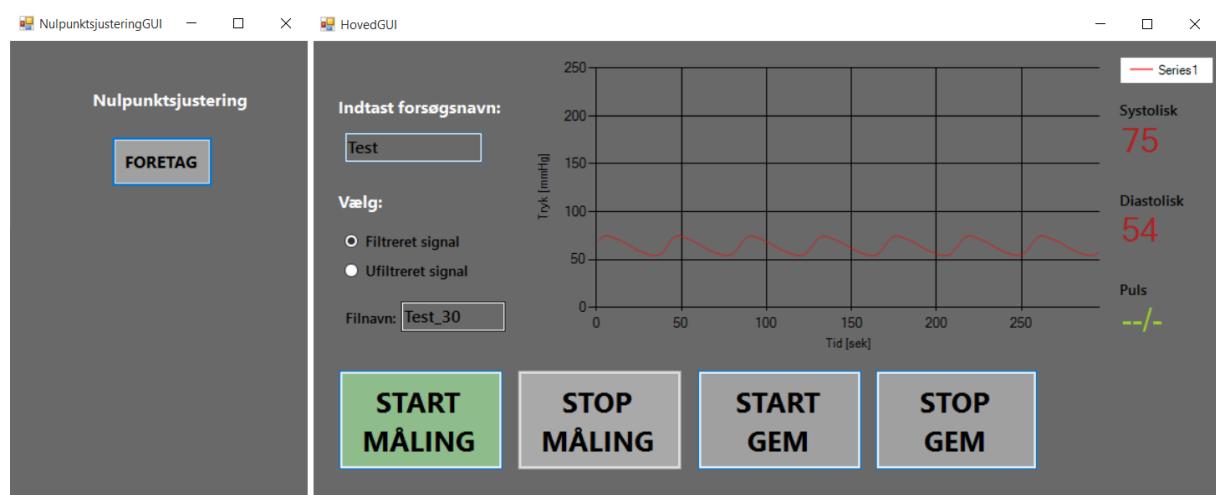
at den skaber et godt overblik i koden, og det fremstår tydeligt hvad hver enkel klasses specifikke ansvar er. Et overordnet klassediagram over systemet er udarbejdet, se figur 6.13. Hvilke metoder hver enkelt klasse indeholder kan ses i Bilag nr. 13.



Figur 6.13: Klassediagram

### Brugergrænseflade

GUI er Forskerens indgang til systemet. Derfor er det vigtigt at denne er opbygget efter Forskerens logik. Til at klarlægge dette er principperne om en god brugergrænseflade taget i mente. Det er et krav at Forsker indtaster et Forsøgsnavn inden en måling kan startes, derfor er komponenterne implementeres således, at knappen Start Måling først bliver aktiveret, når Forsøgsnavn er indtastet i den tilhørende tekstboks.



Figur 6.14: NulpunktsjusteringGUI og HovedGUI

Af figur 6.14 ses det, at grafen er en væsentlig del af displayets HovedGUI. Det vælges at få vist signalet i grafen, som en kurve. Førsteaksen indstilles til samples fra 0-300, og andenaksen til værdier fra 0-250 mmHg, hvilket er givet i Kravspecifikationen.

## Observer - Strategy

Observer og Strategy er to programmeringsmønstre, det er valgt at opbygge softwarekoden efter. Grundet at de i samarbejde med hinanden er gode til, at håndtere overførsel af data fra et lag til et andet. Observer definerer et "en til mange" relation mellem objekter således, at en ændring i et objekts tilstand medfører, at de mange objekter informeres om ændringer og dermed opdateres automatisk. Dette implementeres med to interfaces `IObserver` og `ISubject`. I disse interface er generelle metoder `Notify()`, `Attach()` og `Gennemsnit()` placeret, hvis ansvar er at flytte data fra en klasse til en anden, når metoden kaldes. Mønstret opbygges som en push.

Strategy mønstret indkapsler algoritmer og gør at metoder oprettet i interface kan overskrives i klasser der arver fra interfacet. Således at den nødvendige funktion kan tilføjes i den enkelt klasse.

Hvordan mønstrene er implementeret med relevante metoder kan skematisk ses i Dokumentation figur 3.24 Observer mønstre. I projektet blev mønstrene i første omgang benyttet fra logik-laget til præsentations-laget i forbindelse med at overføre data til visning i graf. Men undervejs viste det sig nødvendigt også at implementere mønstrene fra data-laget til logik-laget, således at det kan kontrolleres, hvor stor en mængde data, der sendes op af gangen.

## Samplefrekvens

Samplefrekvensen er som krav givet til 1000 Hz. Hvilket svarer til at systemet modtager 1000 samples i sekunder. Varigheden af en sample er givet ved:

$$\frac{1}{f_s} = \frac{1}{1000} = 0.001 \text{ sek} \quad (6.7)$$

Det har vist sig under arbejdet med softwaren, at grafen ikke kan følge med, når den modtager så mange målinger i sekundet. Derfor er det valgt at skære i antallet af målinger pr. sekund, der skal viderebearbejdes i logik-laget og udskrives i præsentations-laget. Antallet skæres ned til 50 målinger pr. sekund. Dette gøres ved at gennemsnittet af 20 målinger efter hinanden bestemmes, hvorefter gennemsnitsværdien returneres og gemmes i en liste, der sendes videre i systemet. Herefter findes gennemsnittet af de næste 20 målinger og således fortsætter det.

## Nulpunktsjustering

Formålet med en nulpunktsjustering er at flytte signalets offset op eller ned, så det atmosfæriske tryk altid er placeret ved 0 V på outputsignalet. Justeringsfaktoren er givet ved, hvor  $x$  er det målte atmosfæriske tryk i Volt modtaget gennem DAQ'en:

$$faktor_{jus} = 0 - (x) \quad (6.8)$$

Systemet ønskes nulpunktsjusteret for at sikre, at alle de målte blodtrykssignaler har samme udgangspunkt. Hvilket gør at målingerne kan sammenlignes. Systemet foretager nulpunktsjusteringen ved at returnerer gennemsnittet af de første 20 målinger fra DAQ'en, såfremt den tilsluttede transducer er åbnet så atmosfærisk tryk måles.

## Kalibrering

I dette projekt betyder kalibreringen, at kalibreringskoefficienten fra volt til millimeter kvisksølv bestemmes. Denne bestemmes ved at tilkoble en væskesøjle til systemet. Væskesøjlen fyldes med

vand til markering, så den vil give et kendt tryk (mmHg). Herefter kan output i volt fra hardwaren måles. Kalibreringskoefficienten er givet ved:

$$koeff = \frac{x[\text{mmHg}]}{y[\text{Volt}]} \quad (6.9)$$

x angiver trykket fra væskesøjlen, denne hardcodes til 50 mmHg. Værdien for y angiver det målte spændingsoutput på hardwaren. Optimalt set er kalibreringskoefficienten 50.

Kalibreringen implementeres i softwaren ved brug af konfiguration. Forskeren beregner omsætningskoefficienten ud fra ligning 6.9. Resultatet af denne beregning indtaster Forsker i konfigurations XML-filen under App.settings. XML-filen kan tilgås uden opstart af systemet, derfor bliver kalibreringen uafhængig af, hvornår systemet kører og kalibreringen kan dermed foretages på et vilkårligt tidspunkt.

Det er vigtigt at pointere, at nulpunktsjusteringsfaktoren lægges til samtlige værdier i signalet før kalibreringskoefficienten ganges på. Dette udføres i kodens logik-lag.

## Digitalt Filter

Formålet med implementering af et digitalt filter[12] er at fjerne støj fra det indhentede signal. Dette gøres ved at udglatte signalet. I projektet er det valgt at implementere et glidende middelværdifilter (moving average filter).

Fordelen ved dette filter er, at det er simpelt at forstå og det er optimalt at bruge på signaler i tidsdomænet.

Det glidende middelværdifilter fungerer ved midling af en række punkter fra inputsignalet for, at frembringe hvert punkt i outputsignalet. Hvilke punkter, der tages fra inputsignalet, vil flytte sig én plads for hvert beregnet outputpunkt, heraf kommer den glidende effekt. Matematisk er filteret givet ved:

$$y[i] = \frac{1}{M} \cdot \sum_{j=0}^{M-1} x[i+j] \quad (6.10)$$

Hvor  $x[]$  er inputsignalet,  $y[]$  er outputsignalet og M er antallet af punkter, der benyttes i det glidende middelværdifilter. Denne beregning benytter sig udelukkende af punkter placeret på den samme side af output samplenummeret, hvilket vil føre til en relativ forskydning mellem input og output. M sættes til en længde på fem. Implementeringen af filteret er vist i et aktivitetsdiagram i Dokumentation under Software, Implementering.

Filteret er implementeret således, at der minimum skal være 5 målinger i listen der skal filtreres før, filteret vil starte filtreringen. Dette er en begrænsning, der ikke er optimal, men accepteres. Da den ikke vil være en begrænsning, der vil påvirke Forskerens resultater i betydende grad.

Systemet gør det muligt for Forsker at vælge på GUI om det filtrerede eller ufiltrerede signal ønskes udskrevet.

## Analyse

Analyse dækker over bestemmelsen af de systoliske, diastoliske og puls værdier ud fra blodtrykssignalet. Dette er implementeret i en klasse kaldet Analyse.

Der er implementeret metoder hvori den maksimale værdi og den mindste værdi bestemmes. Værdierne bestemmes ud fra listen, der vises på grafen. Værdierne udskrives i labels på GUI. I

præsentations-laget er implementeret en timer[13], der håndterer at de systoliske og diastoliske værdier på GUI opdateres hvert 3. sekund. Intervallet på 3 sekunder er valgt, da det er passende tid til at aflæse den pågældende værdi.

I forhold til implementering af puls er der gjort en række overvejelser om mulige løsninger. Puls er defineret ved slag pr. minut og på en pulsperiode vil der være en systole og diastole. Pulsen må derfor kunne bestemmes ved at tælle antallet af systoliske værdier på 6 sekunder. Antallet ganges med 10 for at få den rette enhed. En anden mulighed er også at bestemme pulsen ved at finde antallet af samples mellem to systoliske værdier. Omregnes samples til sekunder og ganges op til et minut, må dette være lig med Måleobjektets øjeblikkelige puls. Det er ikke lykkedes at implementere puls ved projektets aflevering.

## Database

I systemet er der implementeret en lokal Database. Databasen er oprettet gennem hosten webhotel10.ihb.dk. Formålet med Databasen er at lagre det målte blodtrykssignals rådata. Det er valgt at implementere Databasen af typen SQL, da denne database-type indeholder de funktioner, som er nødvendige for dette system. Data gemmes i tabeller i Databasen. Indledningsvis for at oprette den nødvendige tabel defineres en type til hver værdi. SQL-koden til oprettelse af tabel er vist på figur 6.15.

```

1  CREATE TABLE [db_owner].[SEMPRJ3] (
2      [Forsøgsnavn]      NVARCHAR (20)    NOT NULL,
3      [Id]                BIGINT          IDENTITY (1, 1) NOT NULL,
4      [Datostempel]       DATETIME        NOT NULL,
5      [Blodtryksmåling]  VARBINARY (MAX) NOT NULL,
6      PRIMARY KEY CLUSTERED ([Id] ASC)
7  );

```

Figur 6.15: SQL-kode til oprettelse af tabeller i database

For yderligere forklaring til de valgte SQL-typer, se Dokumentation under Software, Implementering.

### 6.6.3 Test

I teorien burde man, for at teste, skrive et testprogram, som tester én metode af gangen. Dette er gruppen ikke i stand til, derfor er det valgt at gøre test af softwaren an vha. debugging. Et modul er forsøgt færdiggjort og testet, før det er blevet sat sammen med andre moduler. På den måde er der blevet testet løbende gennem udviklingsprocessen.

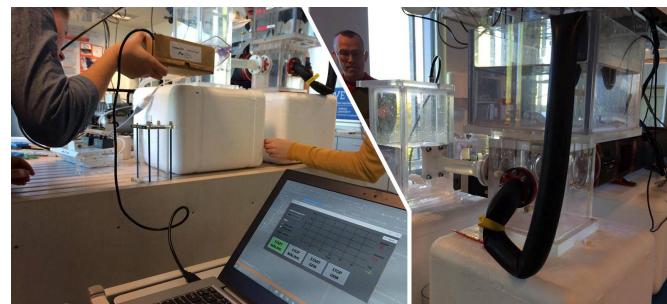
Der bliver vist en eksempel, hvor der debugges igennem funktionen for fra der trykkes på Start Måling-knap til blodtrykssignalet vises i grafen på GUI. Se dette i Dokumentation under Software, Modultest.

#### 6.6.4 Integrationstest

Til sidst i projektforløbet blev en integrationstest[14] udført. En integrationstest laves primært for at teste om softwaren fungerer korrekt, og om enhederne/modulerne deri anvender hinanden. Testen retter sig mod afprøvning af det komplette program, med de eksterne systemer, i dette tilfælde sammen med hardwaren.

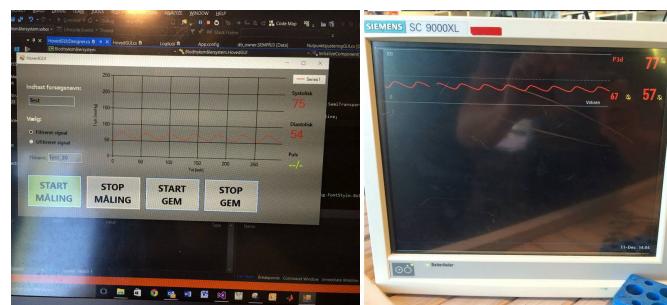
Softwareen er blevet sammensat af de forskellige enheder som fremvisning af graf, indhentning af data, kalibrering, nulpunktsjustering, digitalt filter og at gemme. Hver gang én enhed er færdig, er den blevet testet, hvorefter en ny færdig enhed er blevet sat på osv. Tilsidst er der blevet udført en test hvor alle enheder sættes sammen, software og hardware, og der testes derpå. Testen kan sammenlignes med en "Big Bang" test, da det var første kan vi satte den færdige software sammen med hardwaren.

I dette tilfælde blev In Vitro maskinen i Cave Lab brugt til at simulere et blodtrykssignal. Dette blev sendt ind i systemet og sammenlignet med en anden "rigtig" blodtryksmåler, som var tilkoblet samtidigt. In Vitro maskinen danner et tryk, som efterligner et hjerteslag. Trykket bliver skabt i vand, som presses igennem en falsk hjerteklap, derved er der opbygget en model af et hjerte, som kan give et blodtrykssignal.



Figur 6.16: Opstilling til Integrationstest

Der startes med at lave en nulpunktsjustering på systemet, som bagefter viser et korrekt signal, som lå så konstant, at det var svært at se cursoren. Samtidig stod systolisk og diastolisk tryk i tal på GUI. Her viste det sig, at systemet konstant afveg fra den anden blodtryksmåler med en værdi på 3-4, både på systolisk og diastolisk tryk. Da den konstant afveg, vurderes det at afvigelsen skyldes at systemet ikke var kalibreret eller at den anden blodtryksmåler ikke var nulpunktjusteret. Der kunne ikke udføres en kalibrering, da væskesøjlen var gået i stykker. Det blev også forsøgt at nulpunktsjustere den anden blodtryksmåler, men heller ikke dette kunne udføres. Derfor blev resultatet af testen, at systemet virker som det skal, efter de opstillede krav.



Figur 6.17: Målinger under Integrationstest

## 6.7 Udviklingsværktøjer

Gennem projektarbejdet er der anvendt en række forskellige værktøjer til udvikling af blodtryksmåler-systemet. Disse er yderligere uddybet herunder.

### Visual Studio 2013

Softwaredelen af projektets programmering er skrevet i sproget C#. Her er Visual Studio 2013 anvendt som kompiler, da programmet gør det nemt at omskrive tekst til kode. Visual Studio 2013 indeholder også funktionen Windows Form Application, der visuelt kan fremstille de ønskede resultater i form af knapper, grafer og labels mv. i en samlet brugergrænseflade, som aktøren interagerer med.

### Microsoft Visio 2010

Microsoft Visio er et tegne værktøj, der i dette projekt er anvendt til at designe både SysML og UML diagrammer, som benyttes ved organisering af hardware og software design. Microsoft Visio er det oplagte valg, da diagrammer lavet i programmet får et enkelt og overskueligt udseende, og dermed fremstår det tydeligt for læseren hvad diagrammet vil vise.

### Analog Discovery og WaveForms fra Digilent

Analog Discovery og WaveForms er i projektet benyttet som signal generator under testfasen. Her fungerer Analog Discovery som en WaveForms generator, så et analogt signal kan sendes videre ind i lavpasfiltret, forstærkeren og derefter ind i DAQ'en. I den endelig implementering erstattes Analog Discovery og WaveForms med transduceren.

WaveForms er under test og implementering også brugt som oscillator.

### DAQ

DAQ er et værktøj udarbejdet af National Instruments, som anvendes til at omforme det indkomne analoge signal fra transduceren (Analog Discovery) til et digital signal. DAQ'ens output består af samples, disse værdier kan bearbejdes i softwarekoden. For at kunne benyttet DAQ'en skal et ekstra program fra National Instruments installeres.

### LATEX

LATEX er anvendt i projektet til design og opsætning af projektrapport og projektdokumentation. LATEX er god til tekstformatering, hvor opsætning og strukturer defineres samlet for hele rapport, samt god til versionsstyring. Til at skrive selve koden benyttes programmet TeX-maker som kompiler.

## 6.8 Resultater og diskussion

Resultatet af dette projektforløb er blevet en prototype af en hardware- og en softwaredel, hvis egenskab er at detektere den systoliske og diastoliske værdi ud fra et blodtrykssignal. Denne prototype er blevet udviklet til forskningsbrug. Det kunne være en mulighed at videreudvikle systemet således, at det kan detektere hypotension og hypertension og så alarmere. På denne måde kunne systemet altså bruges til patientbrug, på hospitaler og lignende.

I projektoplægget var der stillet krav til både hardware- og softwaredelen, som alle er blevet op-

fyldt. Derudover er der stillet krav fra vejleder, som er forsøgt løst. Nogle er lykkes, andre er ikke (se problemrapport). Nogle af punkterne i accepttesten var ikke testbare, hvilket bør opfyldes før det ville være muligt at videreudvikle systemet.

I de ikke-funktionelle krav, var målsætningen en MTTR (Mean Time To Restore) på maksimalt 5 timer. Dette var ikke testbart, da systemet stadig var prototype og ikke et færdigt produkt. Det tænkes ydermere, at det endelige produkts hardwaredele forefindes som en sammenkobling af alle dele, så Forskeren kun skal styre en enkelt sammenkoblet hardware del, i stedet for fire dele (DAQ, Veroboard og to batterier). Det ville gøre det lettere for Forskeren at styrer og transportere hardwaren. Hvis nogle af delene i hardware skulle gå i stykker, ville det være nemt for Forskeren blot at udskifte hele hardwaren, i stedet for at skulle fejlfinde og udskifte den bestemte del. Dette er under forudsætningen af tilstedeværende reservedele - forsker vil altså kunne gendanne systemet indenfor 5 timer.

Da systemet stadig er en prototype, bestående af flere løse komponenter, der let kan frakobles, er risikoen for nedbrud relativt høj, hvilket ikke er optimalt for et færdigt produkt. En sammenkobling af alle løse komponenter, som nævnt før, skal sikre en længere Mean Time Between Failure.

Som Projektformulering beskriver, er der udviklet en hardware- og en softwaredel, der sammen kan detektere et blodtryk. Prototypen kan på under tre sekunder vise systolisk og diastolisk blodtryk via en graf. Formålet og Projektformulering er dermed blevet indfriet.

Der er blevet arbejdet og designet en GUI, der i sidste ende har fået et relativt professionelt design, som vil være let, for en Forsker, at benytte. Til videreudvikling vil det være nemt at implementere systemet, på f.eks. et sygehus, da de, som skal benytte systemet uden problemer vil kunne sætte sig ind i hvordan systemet håndteres.

Skal der påpeges en negativ ting om projektet og den endelige rapport, så må det siges at der har været en del forvirring omkring zeta. Zeta ønskes, for det meste, at være 1, hvilket der er blevet arbejdet med i starten af projektet. Derfor var størrelserne på alle komponenterne bestemt ud fra en zeta på 1. Fordi der arbejdes med et 2. ordens butterworth sallen-key lavpasfilter, så er zeta ikke 1, men 0,7. Dette var forvirrende og tog en del tid at forstå, hvorefter alle komponentstørrelserne skulle omregnes, så de passede til en zeta på 0,7. Ved hardwaren har der derudover været en manglende forståelse for de signaler, som sendes ind og måles ved udgangen. Det har givet en stor usikkerhed for, hvornår de forskellige blokke har virket korrekt. Derfor er der blevet spildt tid på at tro at systemet ikke virkede, mens det i virkeligheden gjorde som det skulle.

Ved softwaren er det specielt teorien bag tråde og derfor forståelsen derom, som har manglet og derved skabt flere problemer. Teorien for tråde har været i kurset IT3, dette har dog ikke været fyldestgørende i forhold til brugen i projektet.

Kravene om kalibrering og nulpunktsjustering er blevet opfyldt, selvom det voldte mange problemer. Teorierne bag kalibrering og nulpunktsjustering var forståelig, men hvordan det skulle omsættes til kode, var der ingen anelse om. Så det var problematisk at komme i gang med kodningen af disse.

I løbet af projektet er tidsplanen skredet adskillige gange. Blandt andet pga. mange uforudsete problemstillinger og udefrakommende faktorer, som f.eks. DSB-miniprojekter og en KSS-eksamen, der lå midt i forløbet.

## 6.9 Opnåede erfaringer

I dette projekt handlede det om, at integrere en hardware- og softwaredel. Derfor blev projektgruppen fra starten inddelt i en software- og hardwaregruppe, med indflydelse på hvilket område man ønskede at arbejde med. Ud over dette blev det aftalt hvem, der stod for de praktiske ting, såsom mødeindkaldelser, mødereferater og tidsplan. Det gjorde at man i projektgruppen altid vidste, hvad der var blevet lavet og hvad de fremadrettede opgaver var. Det har gjort at tidsplanen er skrevet af en og at der derfor er en let forståeligt rød tråd igennem udviklingen af tidsplanen. At mødeindkaldelserne kun er skrevet af én gør, at der har været en præcis skabelon, formatet og skriftsprøg er det samme. Det samme kan siges om mødereferaterne. Det giver en bedre sammenhæng og en rød tråd igennem afsnittene. Selvom projektgruppen var delt i to, har grupperne altid sidset og arbejdet sammen. På denne måde fik hele projektet en sammenhæng og en rød tråd igennem rapporten.

ISE blev brugt til udarbejdelse af dokumentation ved hjælp af diagrammer. Igennem projektet blev der i begge grupper, løbende lavet modultests og integrationstests i forbindelse med prototypen.

I hardwaren kunne der bruges erfaringerne fra fagene ASB og KVI, om hvordan forstærkning og filtre fungerer og bygges til det ønskede projekt. I faget KVI blev instrumentationsforstærkeren og strain gauge introduceret, hvorpå der kunne arbejdes videre med metoderne fra faget, hvilket har givet erfaringer til at bruge teorien i praksis, i form af projektet.

I softwaren blev der brugt erfaringerne fra 2. semesterprojekt ved bl.a. brug af trelagsmodellen og oprettelse af en Database. Disse erfaringer er nu blevet udvidet til at bruge tråde ved indsamling af data og en kontinuerlig visning i en graf. Der er et digitalt filter i softwaren, hvorfra viden og erfaringer fra DSB er blevet brugt til at opsætte sampling, aliasering og filter.

Underviserne har været behjælpelige, hvilket har aflastet projektvejlederen. Dette har gjort at møderne kunne blive effektive, da det ikke kan forventes af vejlederen, kan have styr på det hele. Dog har viden om specifikke dele af projektet været centreret på enkelte projektvejledere. Dette har gjort, at det har været svært at få hjælp, da der har været meget pres på disse vejledere.

I dette semester blev det valgt at skrive rapporten og dokumentationen i L<sup>A</sup>T<sub>E</sub>X, som var nyt for de fleste. Dette har voldt mange problemer og frustrationer, men alle har kunne se fordelene ved at benytte L<sup>A</sup>T<sub>E</sub>X som skriveprogram. Det ser professionelt ud og det er rare, at billeder og figurer ikke hopper rundt, som de f.eks. gør i Word. Derfor har alle i projektgruppen været glade for valget af L<sup>A</sup>T<sub>E</sub>X.

## 6.10 Perspektivering - Fremtidigt arbejde

I fremtiden vil blodtryksmåleren kunne udvides gennem flere muligheder. Da blodtryksmåleren er lavet til forskningsbrug, har der i projektforløbet ikke været nogle idéer til at udvikle den til patientbrug. Det er dog muligt, da alle grundprincipperne omkring en god blodtryksmåler er opfyldt, og der ville kunne arbejdes på usability til sygeplejerskerne. Softwaren skal også opdateres til patientjournaler og lignende, men igen, er det muligt at optimere blodtryksmåleren til patientbrug. Det vil her være en god idé at lave en alarm til blodtryksmåleren, som viser når blodtrykket er for lavt/højt. En anden forlængelse af systemet kunne være en metode, som skal kunne vise gemte målinger. Dette vil være nyttigt for både forskere og til en udvidet patientblodtryksmåler, da det skaber overblik over målingerne.

Et log-in vindue er en anden ting, som kunne forbedre systemet, for på den måde at skabe større sikkerhed for forskeren og data. Et log-in vindue vil gøre, at en forsker kan være sikker på, at målingerne og forskningen ikke kan tilgås af andre. Dette kræver en større udvidelse, hvor der skal laves et log-in vindue og en database, hvor password og brugernavn gemmes.

Generelt skal de standarder, som findes for blodtryksmålere undersøges grundigere. Specielt brugergrænsefladen, men også resten af systemet som enheder og visning af graf skal rettes til, efter de passende standarder.

Hvis systemet ydeligere skulle tilpasses forskning, kunne det gøres gennem en bedre navngivning af data i tabellen eller et bedre overblik over, hvordan data bliver gemt f.eks. gennem en liste for de gemte målinger. På den måde vil det blive nemmere for forskeren at finde frem til gamle målinger. I fremtiden vil det være oplagt at filteret bliver implementeret således at hver enkel sampleværdi vil blive vægtet i forhold til dens betydning for outputpunktet. Eksempelvis ved støj, vil filteret være i stand til at kompensere for lave og høje sampleværdier.

I forhold til hardware er målet, at det hele skal samles i en kasse. Så det på den måde ikke er muligt at ændre det. Derved skal filteret og forstærkningen laves på en printplade. Samtidigt skal der i kassen være plads til batterierne, hvor det er muligt at kunne skifte dem, når nødvendigt. Derved fås et samlet system som nemt kan flyttes rundt på, og som ikke er i fare for at gå i stykker.

# Konklusion 7

---

Formålet med projektet var, at udvikle en blodtryksmåler, som består af en hardware- og en softwaredel. Hardwaredelen består af en Forstærker-blok og et lavpasfilter. Hardwaren modtager et differentieret signal fra transduceren, som forstærkes og filtreres inden det sendes gennem en DAQ og videre ind i softwaren. Softwaren vil gennem algoritmer, kunne detektere systoliske og diastoliske værdier.

Da projektet blev lavet til forskningsbrug skulle Forskeren kunne gemme sine målinger i en Database. Herudover var formålet med projektet, at blodtryksmåleren skulle kunne blive kalibreret når det ønskes og nulpunktsjusteres ved opstart.

I dette projekt er det lykkedes, at udvikle en repræsentativ prototype, der opfylder de overordnede formål. Systemet viser den systoliske og diastoliske værdi.

Selvom dette kun er en prototype, så er hardwaredelen blevet udviklet til at være brugervenlig. Kredsløbet er blevet loddet på et Veroboard og lagt i en kasse. Batterierne holdes udenfor kassen, da det på denne måde vil være ubesværet for Forskeren, at skifte batterierne. På kassen vil en lille diode lyse, når batterierne er sat til.

På trods af et funktionsdygtigt slutprodukt, må det konkluderes, at arbejdsprocessen blev stressende. Både hardware- og softwaredelen tog længere tid end forventet. Hensigten med den første tidsplan var god, men pga. DSB-miniprojekter og en KSS-eksamen, blev tidsplanen skubbet. Især softwaredelen blev presset hen mod slutningen. Derfor er det ikke lykkedes at opfylde alle punkter i Kravspecifikationen (Se Problemrapporten i Dokumentation).

Alt i alt, er der blevet udviklet en fornuftig prototype, der med en smule videreudvikling, ville kunne bruges af en Forsker.

# Bilag 8

---

Herunder findes en liste over bilagene.

1. Tidsplan
2. Samarbejds aftale
3. Development Process
4. NI-6009 DAQ Datasheet
5. INA114 Datasheet
6. Instrumentationsforstærkeren
7. Transducer Datasheet
8. OP27 Datasheet
9. Beregninger af overføringsfunktion
10. HW-diagram
11. HW komponentliste
12. Trelagsmodel
13. Klassediagram
14. Softwarekode pdf
15. Softwarekode Visual Studio
16. NI-DAQ kode
17. Mødeindkaldelser
18. Logbog
19. Mødereferater

# Litteratur

---

- [1] Egil Haug og Jan G. Bjålie Olav Sand, Øystein V. Sjaastad. *Menneskets anatomi og fysiologi*. Gads forlag, 2008.
- [2] Egil Haug og Jan G. Bjålie Olav Sand, Øystein V. Sjaastad. *Menneskets anatomi og fysiologi*. Gads forlag, 2008.
- [3] Hanne Ramløv Ivarsen og Ahmed Aziz. *Sygdomslære for sundhedsprofessionelle*. Gads forlag, 2013.
- [4] Hypotension. <http://prodoktor.dk/lavt-blodtryk/>. Online, accessed 26.11.
- [5] Kim Bjerge. *Development Processes*. IHA, 2015.
- [6] Richard G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, 2011.
- [7] Peter Johansen. *Instrumentationsforstærkeren*. IHA, 2014.
- [8] *INA114 Datasheet*.
- [9] Sallen key. [https://en.wikipedia.org/wiki/Sallen%E2%80%93Key\\_topology](https://en.wikipedia.org/wiki/Sallen%E2%80%93Key_topology). Online, accessed 15.12.
- [10] Gregory J. Toussaint Rolande E. Thomas, Albert J. Rose. *The Analysis and Design of Linear Circuits*. Wiley, 2012.
- [11] Lene Hauser. *Trelagsmodel*. IHA, 2015.
- [12] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.
- [13] *Testen af URL i referencer*. Online, accessed 03.12.
- [14] Kurt Nørmark. Objekt-orienteret programmering introduktion, integrationstest. <http://people.cs.aau.dk/~normark/prog1-01/html/noter/test-dokumentation-note-integrationstest.html>, 2001. Online, accessed 14.12.

# Figurer

---

4.1 Skitse af signalændring . . . . .	8
5.1 Use Case diagram . . . . .	10
6.1 ASE-modellen . . . . .	13
6.2 V-model . . . . .	14
6.3 Diagram over HW . . . . .	18
6.4 Bodeplot . . . . .	19
6.5 Forstærkningsblok . . . . .	20
6.6 Måling for 50 Hz . . . . .	20
6.7 Graf til kalibrering, fra udregninger . . . . .	21
6.8 Opstilling . . . . .	21
6.9 Måling ved 50 mmHg . . . . .	22
6.10 Overordnet sekvensdiagram for systemet . . . . .	23
6.11 Domænemodel . . . . .	23
6.12 Applikationsmodel for software . . . . .	24
6.13 Klassediagram . . . . .	25
6.14 NulpunktsjusteringGUI og HovedGUI . . . . .	25
6.15 SQL-kode til oprettelse af tabeller i database . . . . .	28
6.16 Opstilling til Integrationstest . . . . .	29
6.17 Målinger under Integrationstest . . . . .	29



## AARHUS SCHOOL OF ENGINEERING

SUNDHEDSTEKNOLOGI  
3. SEMESTERPROJEKT

---

## Dokumentation

---

### *Gruppe 2*

Anne Bundgaard Hoelgaard	(201404492)
Mette Hammer Nielsen-Kudsk	(201408391)
Ditte Heebøll Callesen	(201408392)
Martin Banasik	(201408398)
Albert Jakob Fredshavn	(201408425)
Johan Mathias Munk	(201408450)

### *Vejleder:*

Studentervejleder  
Peter Johansen  
Aarhus Universitet

16. december 2015

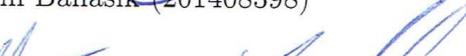
## Gruppemedlemmer

Anne Bundgaard Hoelgaard (201404492) 16/12-2015

Mette Hammer N-k 16/12-2015  
Mette Hammer Nielsen-Kudsk (201408391) Dato

Ditte Heebøll Callesen 16/12-2015  
Ditte Heebøll Callesen (201408392) Dato

Benif 16/12-15  
Martin Banasik (201408398) Dato

Martin Banasik (201408398) Dato  
 16/12-2015

Albert Jakob Fredshavn (201408425) Dato  
 16/12-2015

Mads Munk 16/12-15

Vejleder  
  
Peter Johansen

16/12/15

Dato

# Ordliste

---

Ord	Forklaring
(F)URPS+	Et akronym, der repræsenterer en model til klassificering af softwareens kvalitet
GUI	Graphical User Interface (Grafisk brugergrænseflade)
VPN	Virtual Private Network
DAQ	Data acquisition, NI USB-6009 DAQ MX
SysML	Systems Modeling Language – sprog til visuel fremstilling af systemer
UML	Unified modelling language – sprog til oversigtsfremstilling af klasser i programmering
BDD	Block Definition Diagram
IBD	Intern Block Diagram
SD	Sekvensdiagram
MTTR	Mean Time To Restore
MTBF	Mean Time Between Failure
Veroboard	Printplade med isoleret kobberører
CMRR	Common Mode Rejection Ratio
In Vitro	Maskine til simulering af tryk, i dette tilfælde et blodtryk
KS	Kravspecifikation

# Indholdsfortegnelse

---

<b>Ordliste</b>	<b>ii</b>
<b>Kapitel 1 Indledning</b>	<b>1</b>
<b>Kapitel 2 Kravspecifikation</b>	<b>2</b>
2.1 Versionshistorik . . . . .	2
2.2 Godkendelsesformular . . . . .	3
2.3 Indledning . . . . .	4
2.4 Systembeskrivelse . . . . .	4
2.5 Funktionelle krav . . . . .	4
2.5.1 Aktør-kontekstdiagram . . . . .	4
2.5.2 Aktørbeskrivelse . . . . .	5
2.5.3 Use case-diagram . . . . .	5
2.5.4 Use Cases . . . . .	6
2.6 Ikke-funktionelle krav . . . . .	9
2.6.1 (F)URPS+ . . . . .	9
<b>Kapitel 3 Systemarkitektur</b>	<b>12</b>
3.1 Hardware . . . . .	13
3.1.1 Design . . . . .	13
3.1.2 Implementering . . . . .	14
3.1.3 Modultest . . . . .	18
3.2 Software . . . . .	24
3.2.1 Design . . . . .	24
3.2.2 Implementering . . . . .	30
3.2.3 Modultest . . . . .	39
3.3 Integrationstest . . . . .	44
<b>Kapitel 4 Accepttest</b>	<b>46</b>
4.1 Accepttest af Use Cases . . . . .	46
4.2 Indledning . . . . .	46
4.2.1 Use Case 1 . . . . .	47
4.2.2 Use Case 2 . . . . .	47
4.2.3 Use Case 3 . . . . .	48
4.2.4 Use Case 4 . . . . .	49
4.2.5 Use Case 5 . . . . .	51
4.3 Accepttest af ikke-funktionelle krav . . . . .	52
4.4 Godkendelsesformular . . . . .	58
4.5 Problemrapport . . . . .	59
<b>Litteratur</b>	<b>60</b>



# Indledning 1

---

I dag bruges blodtryksmålere mange steder, både på hospitalet og i hjemmet. Blodtryksmålere kan måle en persons blodtryk, hvor den viser puls, samt diastoliske- og systoliske tryk i numerisk form og afbilledet i en graf.

Vi har valgt at arbejde ud fra, at blodtryksmåleren skal bruges til forskning. Derfor skal systemet gemme samtlige målinger, således at en forsker senere kan tilgå dem. Samtidig skal puls og tryk vises på en graf, som skal være nem at aflæse. Brugeren vil kunne benytte måleren gennem et interface, hvor han kan starte og gemme målinger. Det er også her grafen vises.

Der var fra start givet en række krav til systemet, samtidig har gruppen valgt at tilføje nogle flere for at få de ting løst, gruppen synes var vigtigt. Disse kan findes under Kravspecifikationen.

Nærmere informationer om opbygning af hardware og software kan findes under Systemarkitektur, som er delt ind efter Hardware og Software. Her under findes også Modultest. Under Modultest kan det læses, hvordan vi har testet systemet samlet og enkeltvis for hardware og specifikke softwaredele. Under Accepttest ses det, om systemet opfylder kravene der blev sat.

## Initialer:

Albert Jakob Fredshavn	AJF
Martin Banasik	MBA
Mette Hammer Nielsen-Kudsk	MHNK
Ditte Heebøll Callesen	DHC
Johan Mathias Munk	JMM
Anne Bundgaard Hoelgaard	ABH

# Kravspecifikation 2

---

## 2.1 Versionshistorik

Version	Dato	Ansvarlig	Beskrivelse
0.1	21-09-2015	MHNK og MBA	Oprettelse og udfyldning af kravspecifikation
0.2	24-09-2015	DHC og ABH	Omskrivning af UC1 - UC5
0.3	28-09-2015	ABH	Ikke-funktionelle krav
0.4	08-10-2015	Alle	Tilrette efter review med Grp. 1
0.5	15-10-2015	MBA	Indskrevet i LaTex
0.6	11-11-2015	ABH	Ændre Use Case 1 og 2 efter review med Grp. 4
0.7	20-10-2015	MHNK	Tilretning
0.8	26-11-2015	MHNK	Retning af hele kravspec.
0.9	09-12-2015	DHC	Rettelser ift. software
1.0	09-12-2015	MHNK	Rettelse af afsnit i rapport og dokumentation
1.1	10-12-2015	DHC, ABH	Rettelser i forhold til slutprodukt
1.2	13-12-2015	MHNK	Udfyldelse af accepttest efter gennemgang og godkendelse
1.3	14-12-2015	Alle	Korrekturlæsning

## 2.2 Godkendelsesformular

Antal sider: 62

Forfattere Anne Hoelgaard, Ditte Heebøll Callesen, Martin Banasik, Albert Fredshavn, Mathias Munk og Mette Hammer Nielsen-Kudsk

Godkendes af Peter Johansen

Kunde IHA

Ved underskrivelse af dette dokument accepteres det af begge parter, som værende kravene til udviklingen af det ønskede system.

<u>Aarhus</u>	<u>11/12/15</u>
Sted	Dato
	
Kundens underskrift	Leverandørens underskrift

Figur 2.1: Use Case-diagram

## 2.3 Indledning

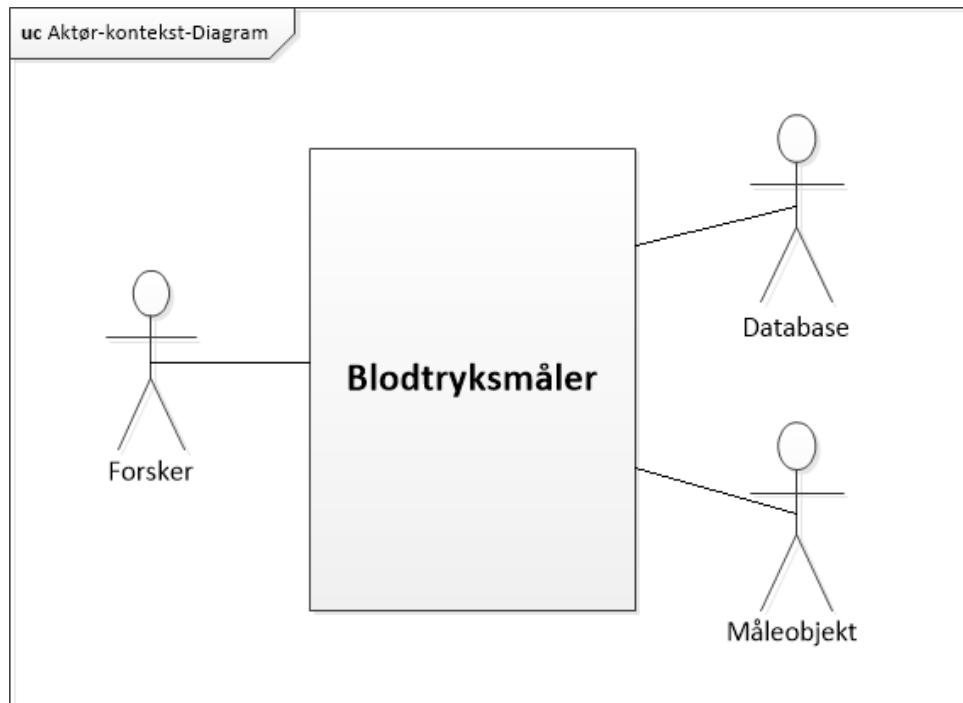
På baggrund af krav fra kunden, samt hvad leverandøren finder muligt, er denne kravspecifikation blevet udarbejdet. Kravspecifikationen har til formål at specificere kravene til produktet. Dette projekt tager udgangspunkt i en blodtryksmåler, hvortil der er en række aktører, som interagerer med systemet. Dette er beskrevet yderligere nedenfor.

## 2.4 Systembeskrivelse

Blodtryksmålersystemet ønskes udviklet således, at systolisk og diastolisk blodtryk, samt puls kan bestemmes ud fra en invasiv arteriel blodtryksmåling. Der udvikles instrumentering til den udleverede transducer, som er hardware og et softwareprogram til kontinuerligt visning af et målt blodtryk. Dette softwareprogram vil blive brugt til udskrivelse af løbende systoliske, diastoliske og puls værdier. Disse to dele udgør tilsammen systemet.

## 2.5 Funktionelle krav

### 2.5.1 Aktør-kontekstdiagram



Figur 2.2: Aktør-kontekstdiagram

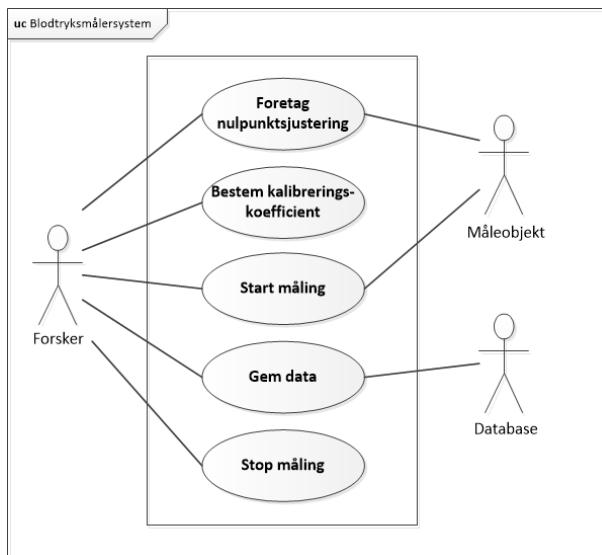
På figur 2.2 ses aktørerne til at være: Forsker, Måleobjekt og Database. Herunder er der en detaljeret beskrivelse af hver aktør.

### 2.5.2 Aktørbeskrivelse

Aktørnavn	Type	Beskrivelse
Forsker	Primær	Forskeren starter måling, giver besked om at data ønskes gemt, navngiver målingen, samt bestemmer kalibreringskoefficient
Database	Sekundær	Databasen er hvor rådata bliver gemt
Måleobjekt	Sekundær	Måleobjektet, hvorfra blodtrykssignalet indhentes. Måleobjektet er tilkoblet transduceren. I den endelige version er måleobjektet In Vitro maskinen, som findes i Cave Lab Under løbende test i udviklingsprocessen benyttes Analog Discovery og Waveform

Tabel 2.2: Aktørbeskrivelse

### 2.5.3 Use case-diagram



Figur 2.3: Use Case-diagram

Diagrammet ovenfor viser systemets fem Use Cases: Foretag nulpunktsjustering, Bestem kalibreringskoefficient, Start måling, Gem data og Stop måling. Herunder følger en nærmere beskrivelse af de enkelte Use Cases, gennem et fully-dressed Use Case skema.

Systemet består af en softwaredel, en DAQ og en transducer med tilhørende hardware. Systemet gør det muligt at foretage en blodtryksmåling på et måleobjekt, hvor transduceren er tilsluttet. Den sender rådata ind i systemet via DAQ'en, hvor signalet vises. Det ønskede stykke af blodtrykssignalet gemmes i databasen.

I softwaren benyttes der algoritmer til at analysere signalet, så systolisk, diastolisk og puls værdier udregnes og vises. Disse algoritmer undersøger signalet for, hvor signalets bølgetoppe og -bunde er placeret. Da toppen(maksimum) er signalets systoliske værdi og bund(minimum) er den diastoliske værdi. Puls bestemmes ved at tælle antallet af blodtryksperioder pr. minut.

Brugergrænsefladen er det som forskeren initierer med, altså hvorfra systemet aktiveres.

### 2.5.4 Use Cases

#### Use Case 1

---

Scenarie	Hovedscenarie
Navn	Foretag nulpunktsjustering
Mål	At få foretaget en nulpunktsjustering
Initiering	Startes af Forsker
Aktører	Forsker (primær), Måleobjekt (sekundær)
Referencer	
Samtidige forekomster	Én nulpunktsjustering pr. kørsel
Forudsætninger	Alle systemer er ledige og operationelle
Resultat	Nulpunktsjustering er blevet foretaget efter ønske
Hovedscenarie	<ol style="list-style-type: none"> <li>1. Pop-up vindue for nulpunktsjustering er åben</li> <li>2. Forsker trykker på Foretag-knap:</li> <li>3. Systemet foretager nulpunktsjustering og vinduet lukker ned.</li> </ol>
Undtagelser	-

---

Tabel 2.3: Fully dressed Use Case 1

#### Use Case 2

---

Scenarie	Hovedscenarie
Navn	Bestem kalibreringskoefficient
Mål	At få bestemt kalibreringskoefficienten
Initiering	Startes af Forsker
Aktører	Forsker (primær)
Referencer	Ingen
Samtidige forekomster	Én kalibrering pr. måling
Forudsætninger	Alle systemer er ledige og operationelle. Væskesøjle og computer med en WaveForm er tilgængeligt. Væskesøjlen er fyldt op med vand, transducer er tilkoblet målepunkt for 50 mmHg
Resultat	Kalibreringskoefficienten er blevet indtastet i XML-fil

---

Hovedscenarie	<ol style="list-style-type: none"> <li>1. Forsker tilslutter WaveForm og væskesøje ved 50 mmHg til systemets hardware</li> <li>2. Output spænding fra hardware aflæses i WaveForm</li> <li>3. Beregning foretages</li> <li>4. Forsker indtaster beregnet kalibreringskoefficient i konfigurations XML-fil</li> <li>5. Kalibreringskoefficienten tilgås af systemet</li> </ol>
---------------	---

Undtagelser	-
-------------	---

Tabel 2.4: Fully dressed Use Case 2

### Use Case 3

Scenarie	Hovedscenarie
Navn	Start Måling
Mål	At få foretaget en blodtryksmåling
Initiering	Startes af Forsker
Aktører	Forsker (primær), Måleobjekt (sekundær)
Referencer	Use Case 1
Samtidige forekomster	Et signal pr. måling
Forudsætninger	Use Case 1 er kørt succesfuldt, samt alle systemer kører og er klar til at foretage en måling
Resultat	Systolisk og diastolisk blodtryk, puls og blodtryksgraf bliver vist på GUI

Hovedscenarie	<ol style="list-style-type: none"> <li>1. Forsker indtaster Forsøgsnavn</li> <li>2. Filtreret signal er valgt per default af systemet</li> <li>3. Forsker trykker på Start-knap på GUI</li> <li>4. Signal for blodtryk vises på GUI</li> <li>5. Systolisk og diastolisk blodtryk, samt puls bliver vist i bokse på GUI</li> </ol>
---------------	---

[Udvidelse 1:] Forsker vælger filtreret/ufiltreret signal

Undtagelser og udvidelser	<p>[Udvidelse 1:] Forsker vælger filtreret/ufiltreret signal</p> <ol style="list-style-type: none"> <li>a. Forsker vælger ufiltreret signal</li> </ol>
---------------------------	--

- b. Det viste signal er nu ufiltreret
  - c. Forsker vælger filtreret signal
  - d. Det viste signal er nu filtreret
- 

*Tabel 2.5: Fully dressed Use Case 3***Use Case 4**

Scenarie	Hovedscenarie
Navn	Gem data
Mål	At gemme rådata i Databasen
Initiering	Startes af Forsker
Aktører	Forsker (primær), Database (sekundær)
Referencer	Use Case 1 og Use Case 3
Samtidige forekomster	Ét signal pr. måling
Forudsætninger	Use Case 1 er kørt succesfuldt, Use Case 3 kører. VPN er tilsluttet
Resultat	Signalets rådata er blevet gemt i en Database under Forsøgsnavn og et autogenereret Id
Hovedscenarie	<ol style="list-style-type: none"> <li>1. Forsker trykker på Start Gem-knap</li> <li>2. Systemet gemmer det fremadrettede signals rådata i Databasen</li> <li>3. Forsker trykker på Stop Gem-knap for at stoppe med at gemme [<i>Undtagelse 1:</i>] Forsker trykker på Stop Måling-knap</li> <li>4. Det vises at rådata er gemt ved at filnavnet (Forsøgsnavn og Id) for målingen vises på GUI</li> </ol>
Undtagelser	<p>[<i>Undtagelse 1:</i>] Forsker trykker på Stop Måling-knap</p> <ol style="list-style-type: none"> <li>a. Systemet gemmer ikke målingen og blodtryksgrafen fastholdes</li> </ol>

---

*Tabel 2.6: Fully dressed Use Case 4*

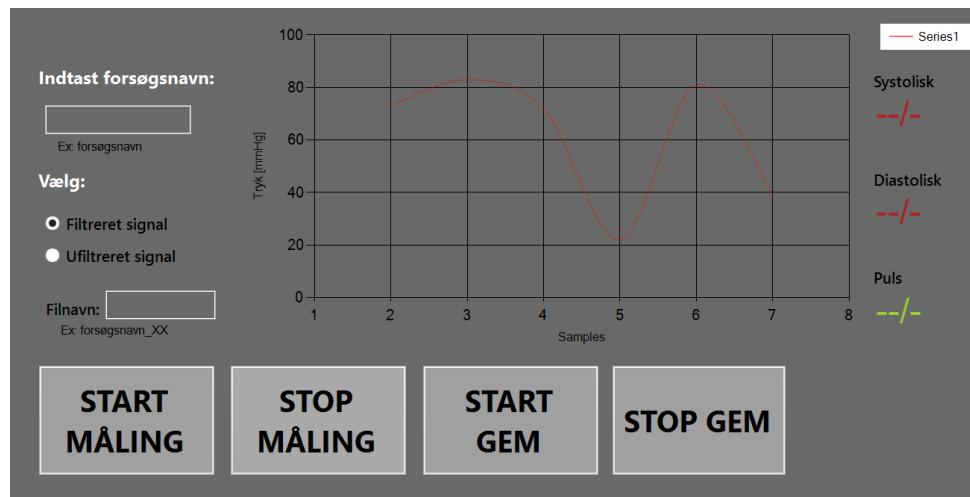
**Use Case 5**

Scenarie	Hovedscenarie
Navn	Stop måling
Mål	At stoppe målingen af blodtryk
Initiering	Startes af Forsker
Aktører	Forsker (primær)
Referencer	Use Case 1 og 3
Samtidige forekomster	Et signal pr. måling
Forudsætninger	Use Case 1 er kørt succesfuldt, Use Case 3 kører
Resultat	Måling af blevet stoppet
Hovedscenarie	<ol style="list-style-type: none"> <li>1. Forsker trykker på Stop Måling-knap</li> <li>2. Målingen stopper og blodtryksgrafen fastholdes</li> </ol>
Undtagelser	-

*Tabel 2.7: Fully dressed Use Case 5***2.6 Ikke-funktionelle krav****2.6.1 (F)URPS+****Functionality**

1. Blodtryksmåleren skal indeholde en Start Måling-knap til at igangsætte målingerne
2. Blodtryksmåleren skal indeholde en Stop Måling-knap, hvorfra måling kan stoppes
3. Blodtryksmåleren skal indeholde en Start Gem-knap til påbegyndelses af at gemme måling i Database
4. Blodtryksmåleren skal indeholde en Stop Gem-knap til afslutning af at gemme måling i Database
5. Blodtryksmåleren skal indeholde en tekstboks til forsøgsnavn, hvori forsker indtaster det pågældende forsøgsnavn
6. Blodtryksmåleren skal indeholde radiobutton til filtreret signal, denne er default
7. Blodtryksmåleren skal indeholde radiobutton til ufiltreret signal
8. Blodtryksmåleren skal indeholde tekstbokse til puls, systolisk og diastolisk blodtryk som vises med op til tre cifre

9. Blodtryksmåleren skal indeholde en tekstboks, som viser filnavn (forsøgsnavn og id) på målingen, efter måling er gemt
10. GUI'en skal se ud som vist på figur 2.4:



Figur 2.4: Skitse af GUI

## Usability

1. Forskeren skal kunne starte en default-måling maksimalt 30 sekunder efter systemet er startet

## Reliability

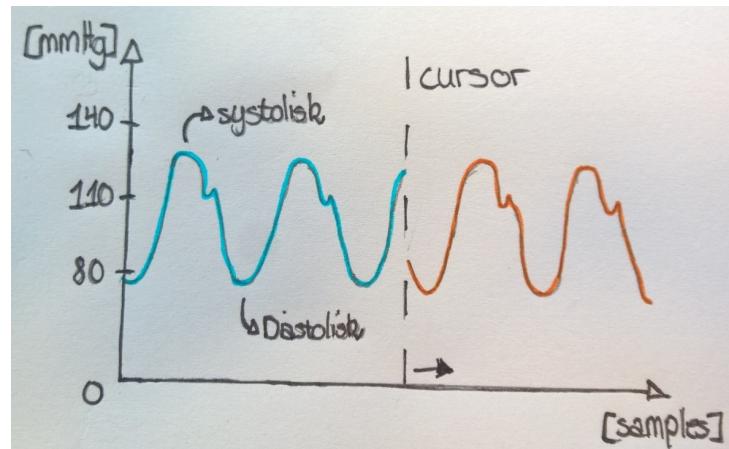
1. Det skal maksimalt tage 5 timer at gendanne systemet (MTTR - Mean Time To Restore)
2. Systemet skal have en oppeid uden nedbrud på minimum 1 måned (720 timer) (MTBF - Mean Time Between Failure)
3. Systemet skal have en oppeid/køretid på:

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \cdot 100 = \frac{720}{720 + 5} \cdot 100 = 99,31\% \quad (2.1)$$

## Performance

1. Blodtryksmåleren skal, indenfor 3 sekunder, kunne vise systolisk og diastolisk blodtryk via grafen. Dette accepteres med en tolerance på +/- 15 %
2. Blodtryksmåleren skal, indenfor 5 sekunder fra der er trykket på Stop Gem-knap, have gemt målingerne i Databasen. Dette accepteres med en tolerance på +/- 15 %
3. Grafen vises i ét vindue, hvor y-aksen måles i mmHg (millimeter kvisksølv) og x-aksen i tid pr. sekund

4. Hvert 3. sekund skal værdier for systolisk og diastolisk blodtryk, samt puls opdateres. Dette accepteres med en tolerance på +/- 15 %
5. Grafen for blodtryk skal køre kontinuerligt i GUI efter følgende princip (figur 2.5), hvor det blå signal erstatter det orange signal ved, at den seneste måling altid sættes ved cursorens placering



Figur 2.5: Graf for blodtryk

6. Når der trykkes på Stop Gem-knap gemmes signals rådata under det indtastede Forsøgsnavn og et autogenereret Id. "Forsøgsnavn\_Id"
7. Systemet skal kunne mæle blodtryksværdier fra 0 til 250 mmHg

### Supportability

1. Forskeren skal kunne udskifte batterierne til hardwaren inden for 2 minutter
2. Softwaren skal opbygges med lav kobling

# Systemarkitektur 3

---

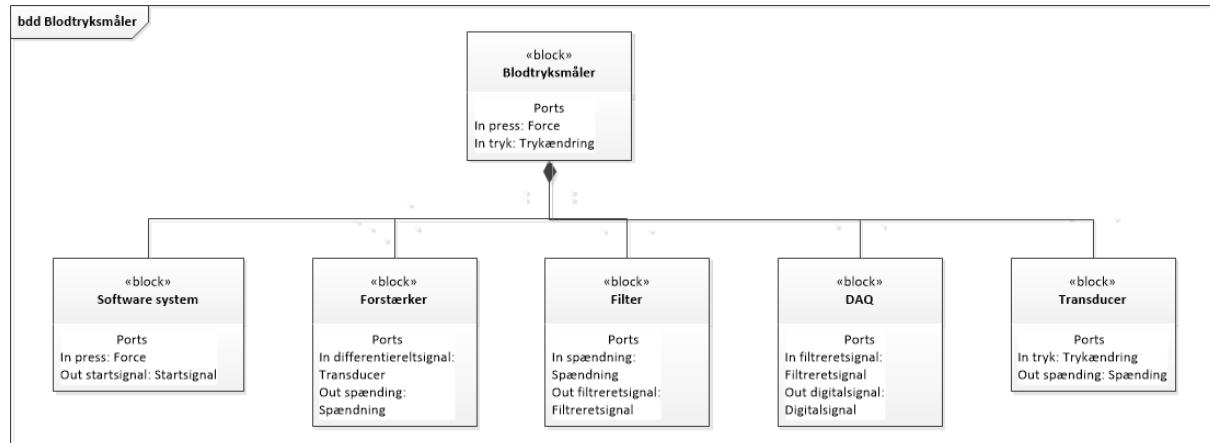
Version	Dato	Ansvarlig	Beskrivelse
0.1	03-11-2015	MBA	Oprettelse
0.2	10-11-2015	DHC, MBA	HW Start af skrivning, indsætning af billeder
0.3	10-11-2015	ABH	SW Start på design, indsætning af diagrammer
0.4	11-11-2015	DHC	HW Design Forstrækning
0.5	13-11-2015	ABH	SW Design klasse- og metodeidentifikation
0.6	18-11-2015	ABH	HW Rettelse af diagrammer
0.7	18-11-2015	DHC, AJF	HW Implementering Forstrækning, Modultest Lavpas
0.8	18-11-2015	MHNK, JMM	SW Design, Rettelse af domænemodel
0.9	18-11-2015	ABH	SW Design, Mere metodeidentifikation
1.0	20-11-2015	MHNK	SW Indskrivning af alle sekvensdiagrammer
1.1	26-11-2015	DHC	HW Modultest, Kalibrering ved vandsøje
1.2	26-11-2015	DHC, AJF	HW Design Lavpas
1.3	02-12-2015	DHC	HW Referencer
1.4	02-12-2015	MHNK	HW Rettelser i tekst
1.5	02-12-2015	DHC, MBA	HW Modultest
1.6	04-12-2015	ABH	SW Implementering, Generelt, Analyse og Digitalt filter
1.7	06-12-2015	ABH	SW Implementering, Kalibrering og nulpunktsjustering
1.8	09-12-2015	DHC	Rettelser i tekst
1.9	09-12-2015	ABH, JMM	SW Implementering Observer-Strategy, Analyse og Digital Filter
2.0	14-12-2015	ALLE	Korrekturlæsning

I det følgende beskrives arkitekturen for systemet. Systemarkitekturen er udviklingsramme for den videreudvikling af design og implementering af blodtrykssystemet. Designet af systemet er grebet an således at, der først kigges på det overordnede system, hvorefter systemet arbejdes ned i mindre brudstykker. Dette gøres ved at benytte diagrammer med tilhørende beskrivelser.

### 3.1 Hardware

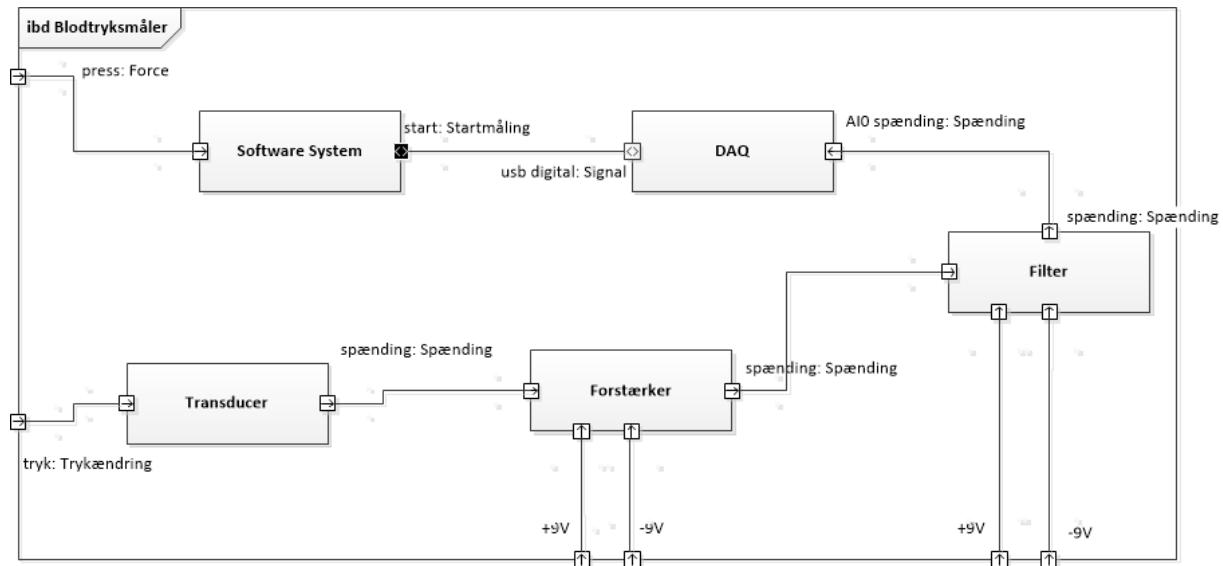
### 3.1.1 Design

Systemets hardware kan illustreres i et BDD. Det ses på figur 3.1 at systemet består af fem hardware blokke: softwaresystem, forstærker, filter, DAQ og transducer. Disse fem blokke udgør tilsammen blodtryksmåleren.



*Figur 3.1: Block Definition Diagram for hardware*

Ovenstående BDD-diagram fører videre til udarbejdelsen af IBD for hardware komponenterne. I IBD diagrammet vises koblingen mellem de forskellige blokke gennem port forbindelser. Det ses at signalet starter ved transduceren, hvorefter det bliver behandlet gennem forstærker, filter og DAQ. Til sidst sendes det ind i softwaresystemet, som bliver påvirket af tryk på knapper på GUI.



*Figur 3.2: Internal Block Diagram for hardware*

## Forstærkning

For at kunne måle fysiske parametre er der benyttet en strain gauge[1], en passiv transducer, der omsætter fysisk tryk til en spænding, som er proportional med trykket. En strain gauge er en resistiv enhed, med en tynd fysisk film af strømledende materiale, afsat på isolerende substrat, der bliver påvirket af et tryk. Strain gauge måler de små mekaniske ændringer af filmen, når den udsættes for et tryk. Jo større tryk, jo mere bliver filmen presset sammen og jo højere spænding bliver genereret. Dog er ændringsafstanden for filmen minimal på systemet og der bliver derfor brugt en Wheatstone[1] bro til at forstærke de detekterede ændringer i strain gauge.

Transduceren udsender derfor et differentieret signal, som sendes ind i Forstærker-blokken. Da signalet fra transduceren er en lav spænding, skal det forstærkes op, for at passe med DAQ'ens input. Denne forstærkning udregnes ud fra det maksimale output fra transduceren og det maksimale input til DAQ'en. Se beregningerne under Implementering.

Under simuleringen bruges Analog Discovery som en funktionsgenerator, der simulerer det differentieret signal. Analog Discovery har en usikkerhed, når der arbejdes med små spændinger. Dette kan modarbejdes vha. spændingsdelerprincippet. Dette gør at Analog Discovery kan sende en højere spænding ind i systemet, så usikkerheden mindskes. Dette bruges kun under simulering og test af hardwaren.

## Lavpas

I projektet skal der laves et 2. ordens lavpasfilter. Filteret skal laves for at sikre, at der ikke opstår aliasering og for at fjerne støj.

Aliasering [2] er, hvor signalet bliver gentaget. Når signalet er i det digitale domæne, bliver spektret for signalet en periodisk funktion. Det vil sige, at den gentager sig selv, efter et bestemt stykke tid.

Det skal sikres, at der ikke kommer overlap mellem signalet og et alias. Da det ellers kan give anledning til misforståelser. Derfor laves et lavpasfilter, som sikre at der ikke ligger noget signal ved den halve samplingsfrekvens. Signalet her kan med fordel gøres så lille at DAQ'en ikke kan læse det, dvs. signalet skal være mindre end  $1/2 \cdot LSB$  (Least Significant Bit).

Lavpasfilteret skal være et Sallen-Key Butterworth-filter med en knækfrekvens på 50 Hz og en samplingsfrekvens på 1kHz. Ud fra oplysninger givet til projektet, vides det at filteret skal dæmpe signalet med 20 dB, under antagelse af at, den forekommende støj er mindre end signalet, også når støjen forekommer over knækfrekvensen.

Ved en typisk blodtryksmåling forekommer der ikke signaler over 50 Hz, samtidigt er signalet her aftaget med ca. 70 dB. For at få signalet, ved den halve samplingsfrekvens til at være  $1/2 \cdot LSB$ , skal det ydeligere dæmpes 20 dB. Derfor oplyses filterets knækfrekvens til at være 50 Hz, da dette giver en minimum dæmpning på 20 dB pr. dekade.

### 3.1.2 Implementering

#### Forstærkning

For at få den rette forstærkning er det blevet valgt, at benytte instrumentationsforstærkeren INA114. Her kan transduceren sættes på med det differentierede signal. INA114 er valgt da følgende gælder[3] for instrumentationsforstærkeren:

- Differentielt input - single ended output
- Gain justering med ændring af kun én modstand
- Meget høj indgangsimpedans
- Stor Common Mode Rejection Ratio(CMRR)

Under opbygning og modultestning vil det differentierede signal blive simuleret af Analog Discovery.

For at udregne den korrekte forstærkning, bruges følsomheden fra transduceren og eksistations-spændingen, som kommer fra to 9 V batterier. Først udregnes det maksimale output fra transduceren:

$$9V \cdot 250mmHg \cdot 5\mu \cdot 10^{-5}uV/V/mmHg = 11.25mV \quad (3.1)$$

Da det er besluttet at det maksimale input til DAQ'en [2] er 5V, kan forstærkningen (Gain) nu udregnes:

$$\begin{aligned} 5V &= 11.25mV \cdot G \\ G &= 444.44 \end{aligned} \quad (3.2)$$

[4] For at få den rette forstærkning udregnes den eksterne modstand ( $R_g$ ) til INA114. INA114's forstærkning afhænger af størrelsen på  $R_g$ , hvis modstanden er stor, er forstærkningen lille og omvendt.  $R_g$  udregnes ved formlen:

$$\begin{aligned} G &= 1 + \frac{50k\Omega}{R_g} \\ 444.44 &= 1 + \frac{50k\Omega}{R_g} \Rightarrow R_g = 112.75\Omega \end{aligned} \quad (3.3)$$

Derved fås en værdi for den eksterne modstand til INA114, som skaber den ønskede forstærkning. Det skal nu sikres at dette kan lade sig gøre. Derfor sikres det, at den ønskede forstærkning kan ske ved båndbredden. Dette kan undersøges da produktet af forstærkning og båndbredde er en konstant. Konstanten aflæses i databladet for INA114[4].

$$\begin{aligned} 1000000Hz &= G \cdot BW \\ BW &= 2250Hz \end{aligned} \quad (3.4)$$

Da båndbredden ligger over knækfrekvensen for lavpasfiltret, er dette godkendt. Hvis båndbredden havde ligget under knækfrekvensen vil operationsforstærkeren ikke have kunnet arbejde med de ønskede frekvenser. Derfor er det vigtigt at båndbredden er bred nok til at kunne indeholde frekvenser på begge side af knækfrekvensen.

For at imødekommme usikkerheden ved Analog Discovery med lave spændinger, laves et kredsløb efter spændingsdelerprincippet. Signalerne fra Analog Discovery skal sendes igennem dette kredsløb, hvor de efter spændingsdelerprincippet gøres mindre. I kredsløbet benyttes to modstande pr. indgang, hvis værdier er  $R_1 = 100k\Omega$  og  $R_2 = 1k\Omega$ . Da vi kender signalet som skal ind i INA114 og modstandene i kredsløbet, kan størrelsen af den spænding, som skal sendes fra Analog Discovery, findes:

$$\begin{aligned} U_{INA} &= U_{analog} \cdot \frac{R_2}{R_1 + R_2} \\ 11.25mV &= U_{analog} \cdot \frac{1k\Omega}{100k\Omega + 1k\Omega} \Rightarrow U_{analog} = 1.1362V \end{aligned} \quad (3.5)$$

Derved kan Analog Discovery sende signaler med en højere spænding ud og usikkerheden for lave spændinger mindskes. Der er taget højde for, at hvis modstandene i kredsløbet bliver for store, vil det skabe en termisk usikkerhed. Derfor er modstandene valgt som de er. Dette bruges kun under simulering. Når transduceren benyttes, bruges spændingsdeleren ikke.

### Lavpas

For at opnå den ønskede effekt i lavpasfilteret, blev det oplyst at  $f_c = 50 \text{ Hz}$ ,  $f_s = 1\text{kHz}$ ,  $R_1 = R_2$  og  $C_2 = 680\text{nF}$ . Ud fra disse værdier, udregnes de resterende komponentværdier for filteret.

Overføringsfunktionen for et 2. ordens filter er:

$$H(z) = \frac{\omega_n^2}{(s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2)} \quad (3.6)$$

For at finde overføringsfunktionen for det gældende system, vides det at følgende ligninger gælder [5]:

$$\begin{aligned} \omega_n &= 2 \cdot \pi \cdot 50 = \frac{1}{\sqrt{R_1 \cdot R_2 \cdot C_1 \cdot C_2}} \\ 2 \cdot \zeta \cdot \omega_n &= \frac{1}{C_2} \cdot \left( \frac{R_1 + R_2}{R_1 \cdot R_2} \right) \end{aligned} \quad (3.7)$$

Derved fås en overføringsfunktion, som hedder:

$$H(z) = \frac{\left( \frac{1}{\sqrt{R_1 \cdot R_2 \cdot C_1 \cdot C_2}} \right)^2}{s^2 + \left( \frac{1}{C_2} \cdot \left( \frac{R_1 + R_2}{R_1 \cdot R_2} \right) \cdot s \right) + \left( \frac{1}{\sqrt{R_1 \cdot R_2 \cdot C_1 \cdot C_2}} \right)^2} \quad (3.8)$$

Da det bliver oplyst at  $R_1 = R_2$ , kan funktionen reduceres. Den kan samtidig simplificeres. I sidste ende fås følgende overføringsfunktion, se Bilag nr. 9 for nærmere udregninger.

$$H(z) = \frac{\frac{1}{C_1 \cdot C_2 \cdot R^2}}{s^2 + s \cdot \frac{2}{R \cdot C_2} + \frac{1}{C_1 \cdot C_2 \cdot R^2}} \quad (3.9)$$

Da der arbejdes med et 2. ordens Butterworth filter, vides det at udsvinget  $\zeta$  skal være 0.7 [6]. Den sidste overføringsfunktion sammenlignes med den generelle for 2. ordens systemer. Det gælder at  $C_2 = 680 \cdot 10^{-9}\text{nF}$ . Det er muligt at isolere forskellige led. Først isoleres der for modstanden:

$$\begin{aligned} \frac{2}{R \cdot C_2} &= 2 \cdot \zeta \cdot \omega_n \\ \frac{2}{R \cdot 680 \cdot 10^{-9}} &= 2 \cdot 0.7 \cdot (2 \cdot \pi \cdot 50) \\ &\Downarrow \\ R &= 6687\Omega \end{aligned} \quad (3.10)$$

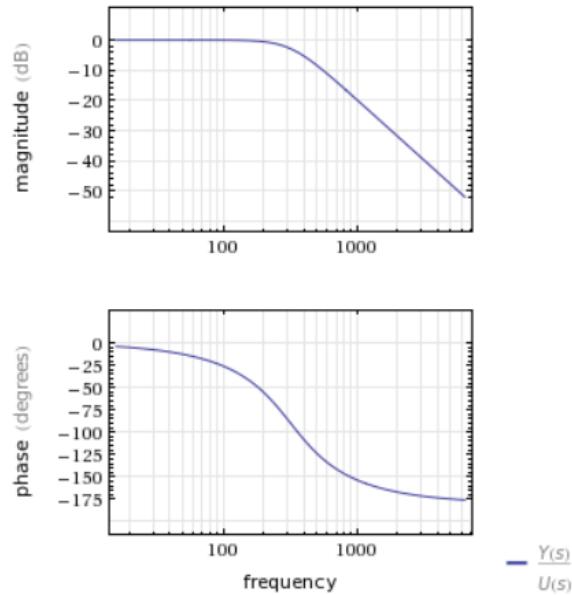
Derved er modstandene udregnet til  $R = 6687\Omega$ . Nu kan der isoleres for kondensator  $C_1$ :

$$\begin{aligned} \frac{1}{C_1 \cdot C_2 \cdot R^2} &= \omega_n^2 \\ \frac{1}{C_1 \cdot 680 \cdot 10^{-9} \cdot 6687^2} &= (2 \cdot \pi \cdot 50)^2 \\ &\Downarrow \\ C_1 &= 333 \cdot 10^{-9}\text{nF} \end{aligned} \quad (3.11)$$

Dette betyder, at  $C1 = 333 \cdot 10^{-9}nF$  og  $C2 = 680 \cdot 10^{-9}nF$ . Derved er alle komponentværdierne til lavpasfilteret fundet og det kan nu realiseres.

Under udviklingen af lavpasfilteret er komponentstørrelserne blevet ændret for at kunne realisere det. De er blevet ændret da det ikke var muligt at realiserer de udregnede størrelser. De brugte komponentstørrelser er:  $R = 6.6k\Omega$ ,  $C1 = 330 \cdot 10^{-9}nF$  og  $C2 = 680 \cdot 10^{-9}nF$ . For at være sikker på at filteret har de ønskede karakteristika, laves et bodeplot for den endelig overføringsfunktion:

$$H(z) = \frac{62500000000}{610929 \cdot \left( s^2 + \frac{250000}{561} \cdot s + \frac{62500000000}{610929} \right)} \quad (3.12)$$

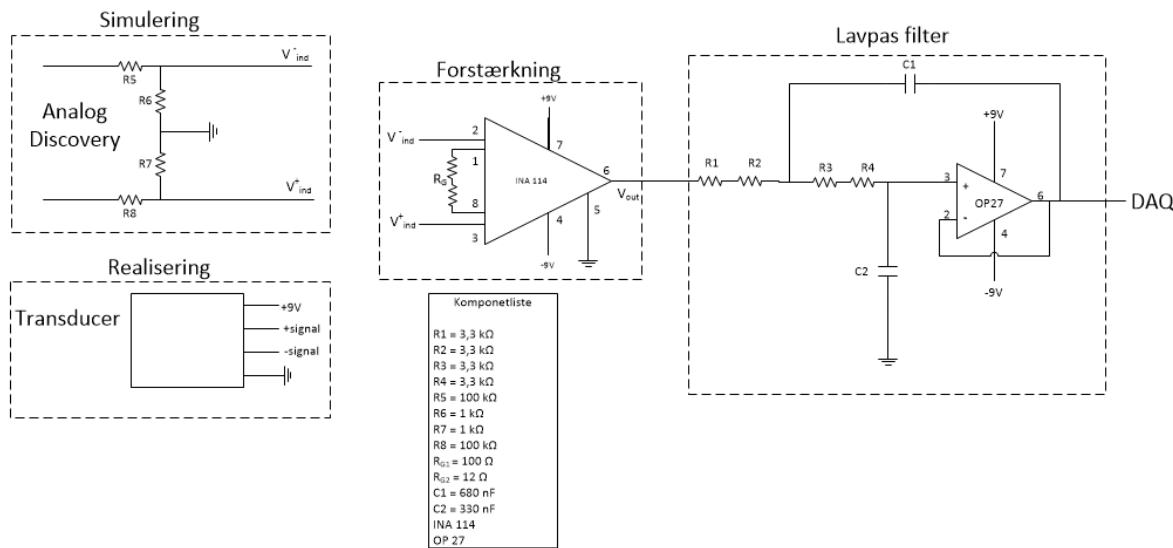


Figur 3.3: Bodeplot

Udregning af det præcise oversving  $\zeta$  ud fra de benyttet komponentværdier:

$$\begin{aligned} \frac{2}{R \cdot C1} &= 2 \cdot \zeta \cdot \omega_n \\ \frac{2}{6600 \cdot 680 \cdot 10^{-9}} &= 2 \cdot \zeta \cdot (2 \cdot \pi \cdot 50) \\ \downarrow \\ \zeta &= 0.709 \end{aligned} \quad (3.13)$$

Dvs. de små ændringer i komponentværdierne har ikke haft betydende indflydelse på værdien for  $\zeta$ .



Figur 3.4: Diagram over HW

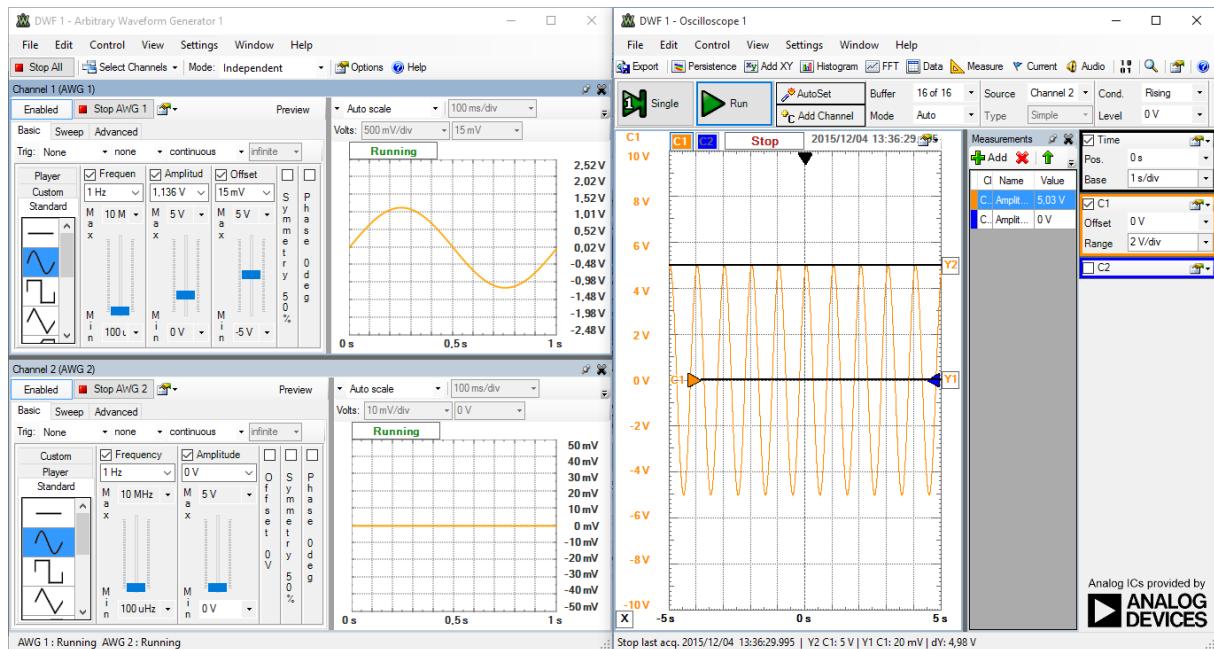
På figur 3.4 ses et diagram over, hvordan kredsløbet er opbygget. Her ses kredsløbet for realiseringen med transduceren og for simuleringen med Analog Discovery.

### 3.1.3 Modultest

#### Forstærkning

For at teste forstærkningen sendes et differentieret signal ind vha. Analog Discovery. Signalet måles ved udgangen og der ses på, hvor meget signalet er blevet forstærket.

På figur 3.5 ses det signal, som sendes ind i Forstærker-blokken og det, der måles på udgangen af blokken.

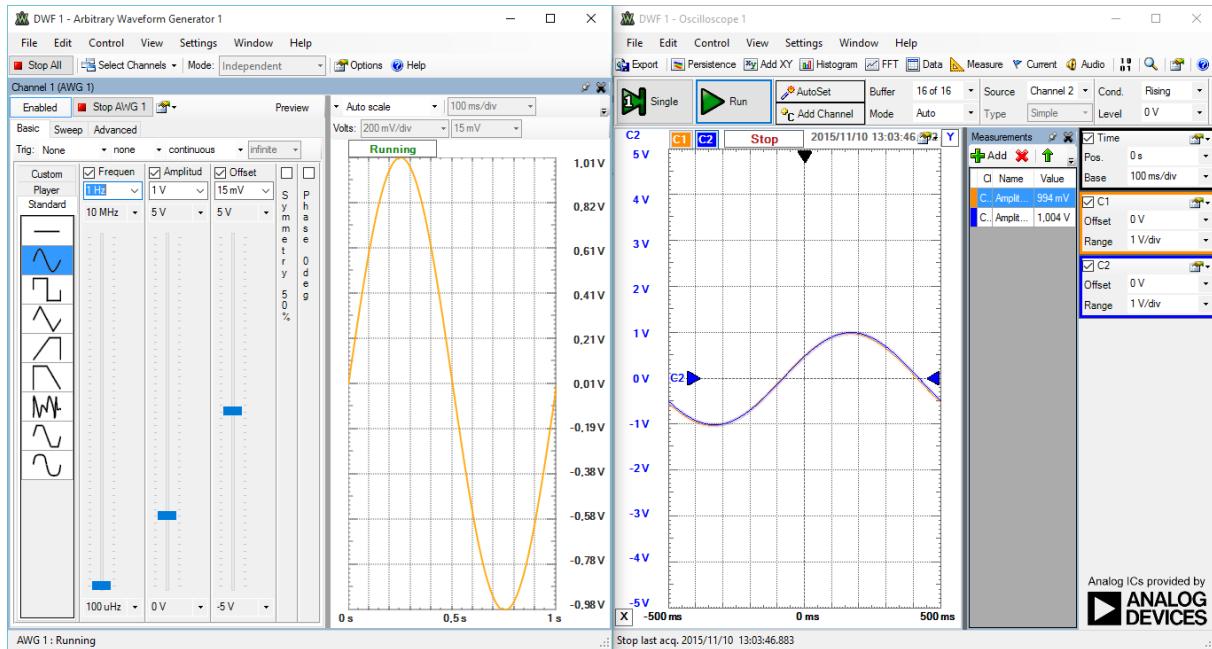


Figur 3.5: Forstærknings blok

På udgangen ses det, at signalet er blevet forstærket op til 5 V DC. Herved er det maksimale output fra transduceren blevet forstærket så det passer med det maksimale input til DAQ'en. Signalet bliver ikke ændret på andre måder i Forstærker-blokken.

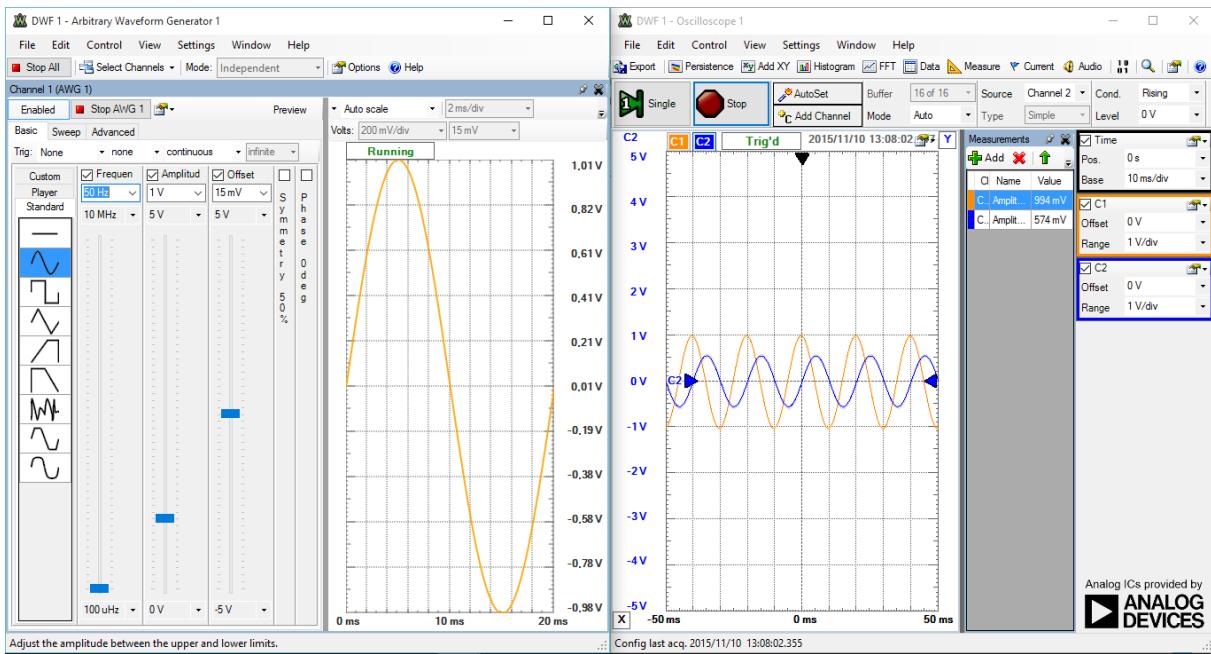
### Lavpas

For at teste lavpasfilteret foretages målinger med en sinus, hvor frekvensen varierer for hver måling. Fasen aflæses mellem indgang- og udgangssignalet. Amplituden aflæses ligeledes for hver måling. Ved knækfrekvensen skal fasedrejningen være  $90^\circ$ . Dette kan aflæses på figur 3.7. Efter knækfrekvensen skal amplituden gå mod nul. Ved målingen for 60 Hz på figur 3.8, kan det ses, hvordan amplituden er faldet drastisk efter knækfrekvensen.

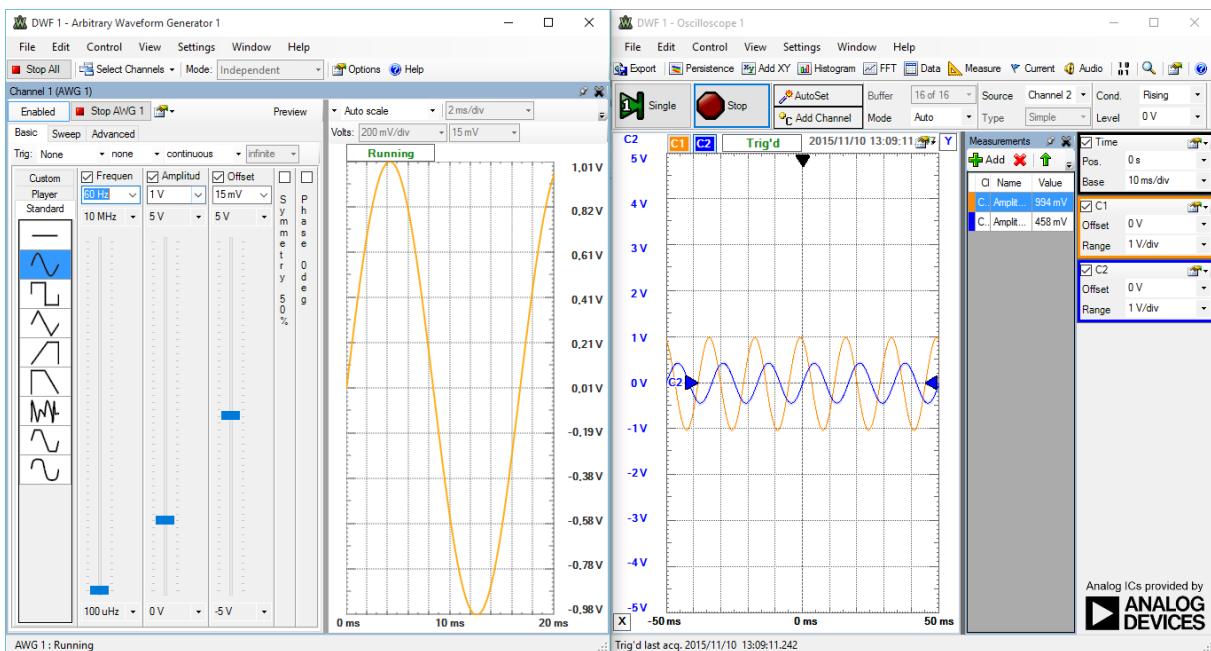


Figur 3.6: Måling for 10 Hz

### 3.1. Hardware



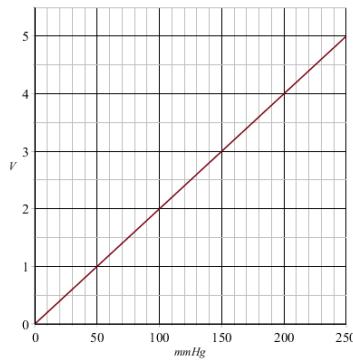
Figur 3.7: Måling for 50 Hz



Figur 3.8: Måling for 60 Hz

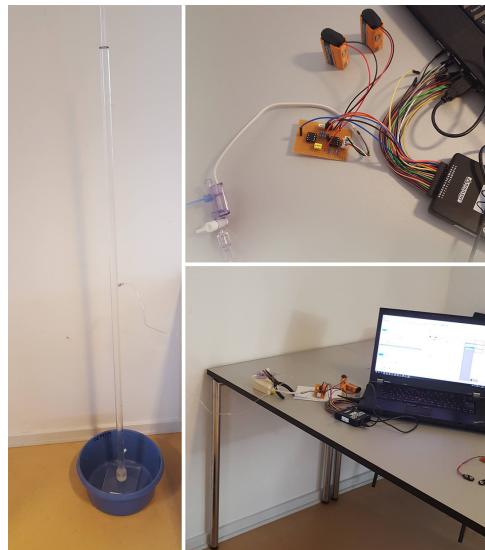
### Kalibrering med væskesøjle

Efter forstærkning og lavpasfilteret er blevet testet hver for sig, udføres en kalibrering af systemet vha. en væskesøjle. Her bruges en udleveret væskesøjle med tre målepunkter, hvor det er angivet, hvor højt trykket(mmHg) er ved hvert af disse punkter. Derved kan det testes om hardwaren måler den rigtige spænding i forhold til millimeter kvisksølv(mmHg). Ud fra den maksimale spænding (V) og millimeter kvisksølv(mmHg) kan det udregnes, hvad hardwaren skal vise ved 100 mmHg.



Figur 3.9: Graf til kalibrering, fra udregninger

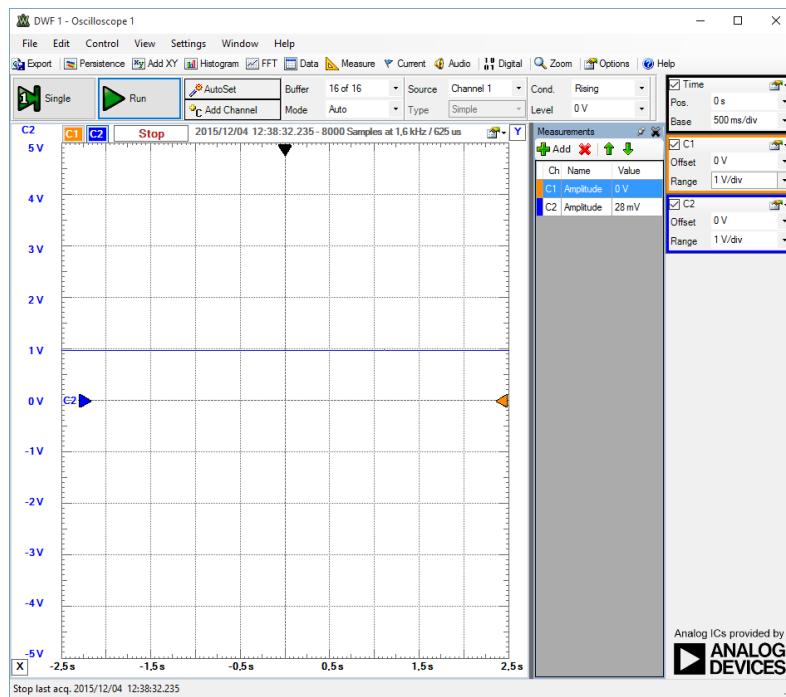
Testen udføres ved, at fyde vand i søjlen til markeringen. Transduceren skal være tilkoblet et af de tre målepunkter, mens de andre er lukket til. Transduceren er sat til forstærkningen, hvor Analog Discovery tidligere har været sat til. Transduceren er tilkoblet 9 V ved batterierne. På samme måde som ved simuleringen aflæses målingen på computeren ved hjælp af programmet WaveForms. Da det vides, hvilken trykændring der måles på, ved vi fra kalibreringsgrafen, hvilken spænding den skal vise. Dette foretages for de tre målepunkter på væskesøjlen, hvor hver måling sammenlignes med den udregnede graf. For hver måling, skal transduceren flyttes til et af de andre målepunkter.



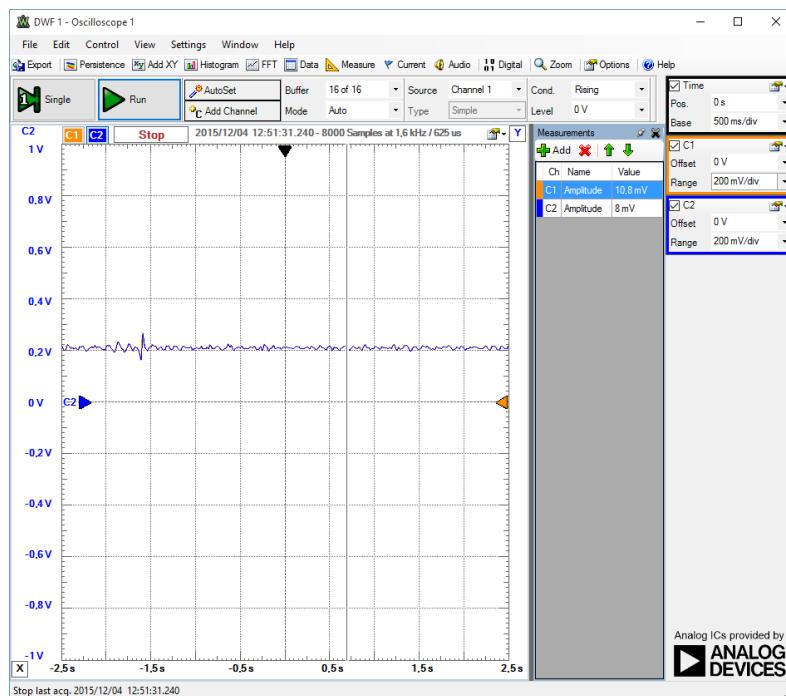
Figur 3.10: Opstilling

Opstillingen er gjort klar og der hentes ekstra vand under testen. Vandet skal bruges til at fyde væskesøjlen op mellem hver måling.

Ud fra grafen i figur 3.9 vides der, hvad svaret på hver måling skal være. På figur 3.11 ses målingen fra det tidspunkt hvor transduceren var tilkoblet målepunktet for 50 mmHg. Ud fra figur 3.9 ses det at målingen skal vise 1 V DC.

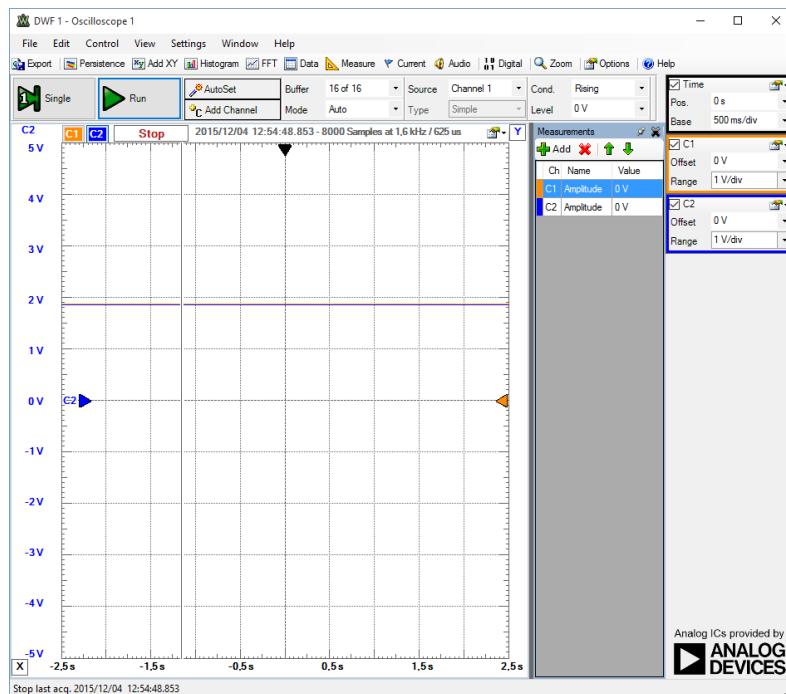


Figur 3.11: Måling ved 50 mmHg



Figur 3.12: Måling ved 10mmHg

På målingen for 10 mmHg ses en del rystelser(udsving på signalet). Som det ses på figur 3.12 ligger signalet ikke præcist på 0.2 V, dette kan skyldes at under testen, skal transduceren være i højde med målepunktet. Pga. korte ledninger, blev det under testen derfor nødvendigt at løfte og holde transduceren, Veroboard og Analog Discovery i højde med målepunktet.



Figur 3.13: Måling ved 100mmHg

Ved målingen for 100 mmHg skulle der måles en spænding på 2 V. Som det ses på figur 3.13 ligger den ikke præcis på 2 V. Som under målingen for 10 mmHg skal transduceren være i samme højde som målepunktet. Her er målepunktet lavt, men det skaber stadig en del usikkerhed.

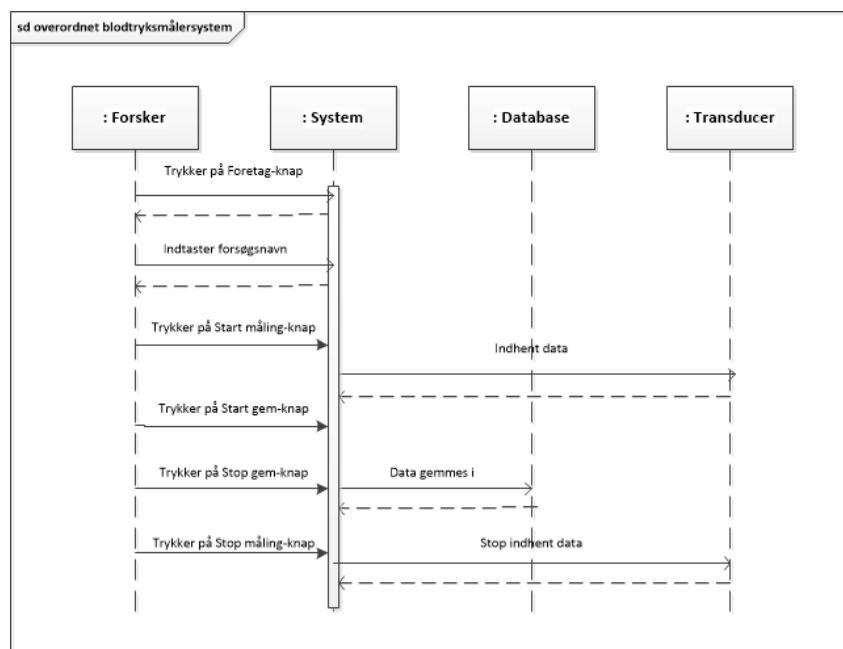
## 3.2 Software

### 3.2.1 Design

I dette afsnit beskrives systemets softwaredesign på baggrund af systembeskrivelsen og kravspecifikationen. De overvejelser, som er gjort i forbindelse med design af software vil blive præsenteret i dette afsnit.

#### Overordnet sekvensdiagram

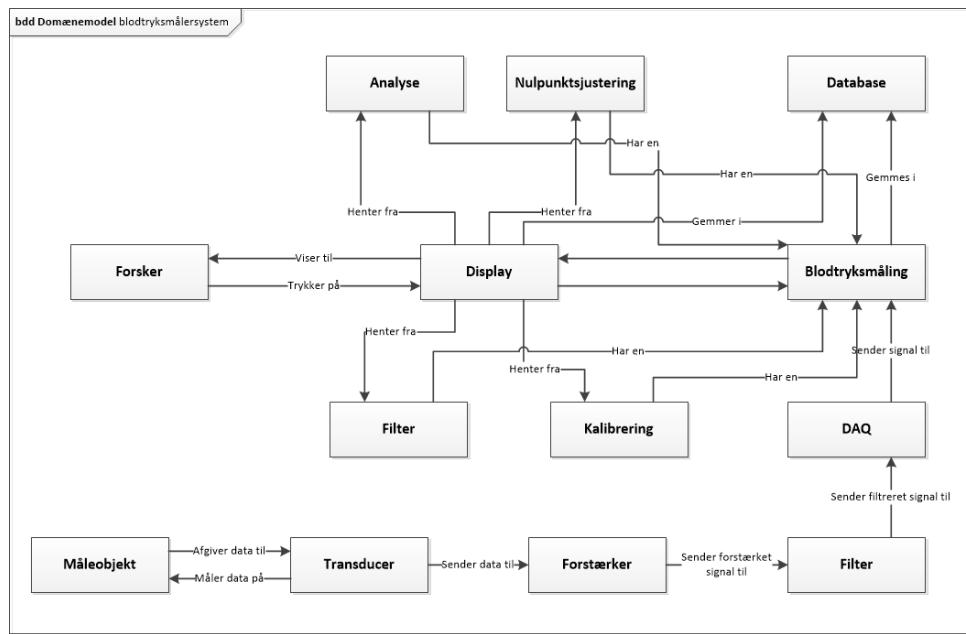
Overordnet set ønskes det at udvikle et system, der kan interagerer med en forsker. Diagrammet herunder viser at forskerens opgave består i at starte en måling, foretage en nulpunktsjustering og gemme de ønskede rådata, samt uafhængigt af systemet at bestemme en kalibreringskoefficient, hvis det ønskes. Diagrammet er en simpel illustration, som viser systemets adfærd gennem alle fem Use Cases. Formålet med dette diagram er at skabe et overblik over det samlede system.



Figur 3.14: Overordnet sekvensdiagram for systemet

#### Problemidentifikation

Første step i softwaredesignet er at klarlægge, hvilke klasser systemet skal bestå af. Til dette er en domænemodel udarbejdet med udgangspunkt i de fem Use Cases. I de fem Use Cases er de konceptuelle klasser blevet identificeret, og derefter indført som klasser i nedenstående domænemodel. Modellen har til formål at vise, hvilke dele systemet skal holde styr på.

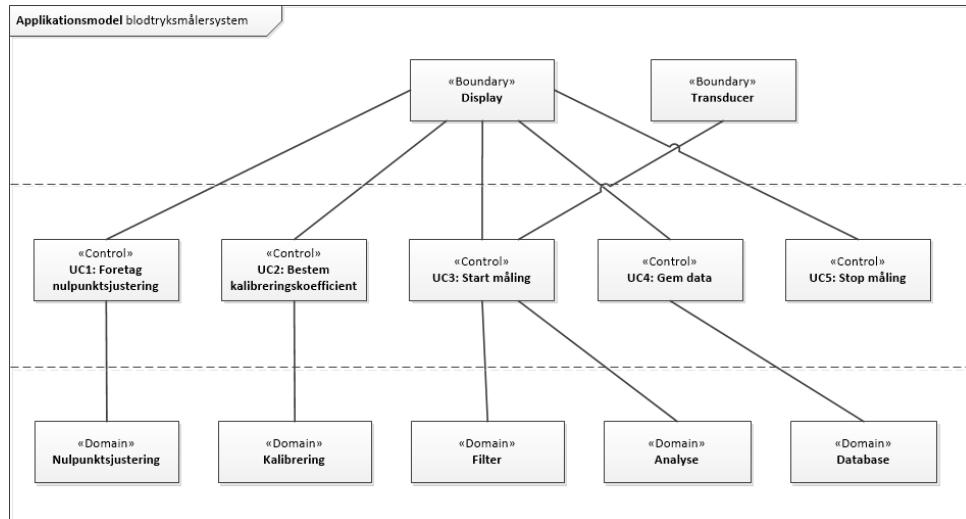


Figur 3.15: Domænemodel

Diagrammet viser forskerens interaktion med displayet, samt hvilke handlinger interaktionen starter i systemet. Hardwarekomponenterne er medtaget for at vise signalets vej fra måleobjekt til systemet.

### Klasseidentifikation

Ud fra domænemodellen kan en applikationsmodel udarbejdes, dette diagram tager også udgangspunkt i de fem Use Cases. Hensigten med et klassediagram er at klarlægge hver klasses individuelle formål.



Figur 3.16: Applikationsmodel for software

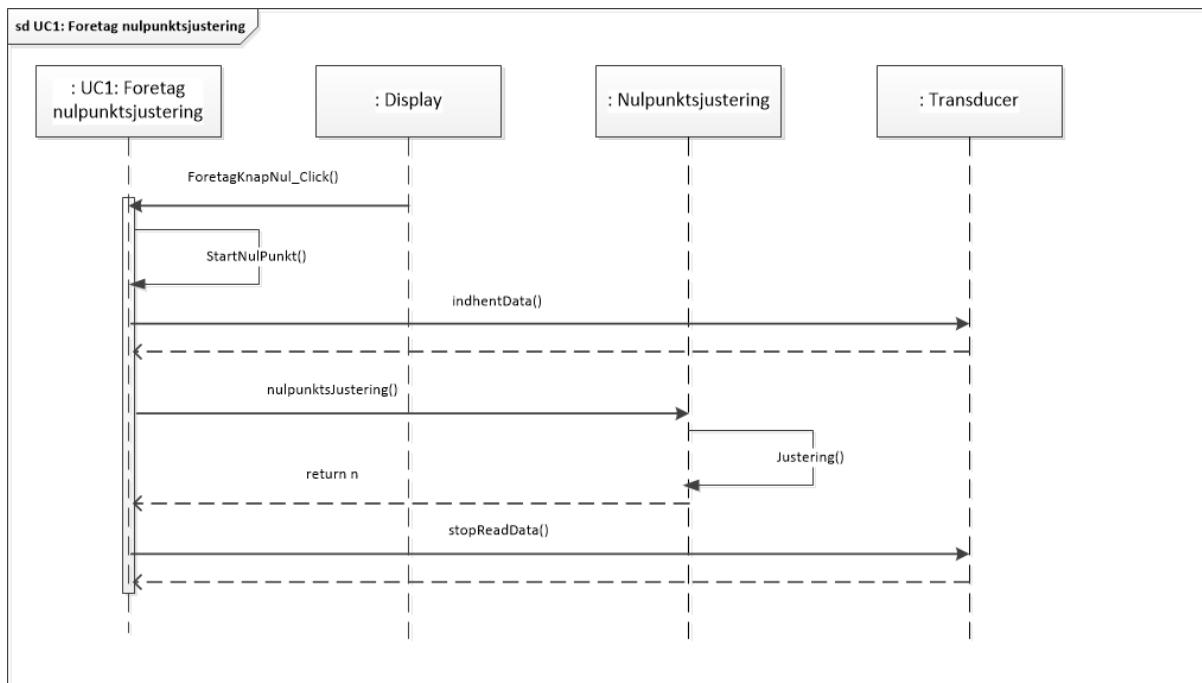
På figur 3.16 ses det at denne model er delt op i tre niveauer:

1. Grænsefladeklasse
  - a) Transducer - Indhentet data fra måleobjekt
  - b) Display - Brugergrænseflade til forsker
2. Kontrolklasse
  - a) UC1: Foretag nulpunktsjustering
  - b) UC2: Bestem kalibreringskoefficient
  - c) UC3: Start måling
  - d) UC4: Gem data
  - e) UC5: Stop måling
3. Domæneklasses
  - a) Database
  - b) Nulpunktsjustering - Bestemmer nulpunktsjusteringsværdi
  - c) Kalibrering - Bestemmer kalibreringskoefficient
  - d) Filter - Indholder det digitale filter
  - e) Analyse - Bestemmer systole, diastole og puls

### **Metodeidentifikation**

Klasserne i ovenstående klassediagram er med til at definere, hvilke blokke de følgende sekvensdiagrammer må indeholde. Det er yderst vigtigt, at der er en sammenhæng mellem klasserne i klassediagrammet og blokkene i sekvensdiagrammet. Det er blevet valgt at udarbejde et sekvensdiagram for hver enkelt Use Case, hvori systemets interne kommunikation beskrives, når normalforløbet og udvidelser gennemløbes. I alle diagrammerne beskrives forløbet via de metodekald, der er nødvendige for at få de ønskede handlinger mellem blokkene udført. I de efterfølgende diagrammer dækker Transducer-blokken over alt hardware og DAQ'en.

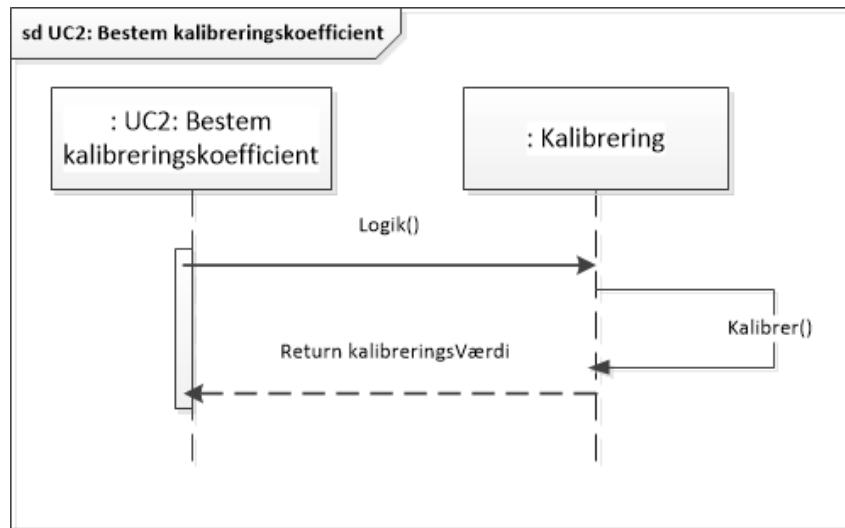
### Use Case 1



Figur 3.17: Sekvensdiagram for Use Case 1

Det ses af ovenstående sekvensdiagram at Forsker interagerer med display ved tryk på en knap. Denne interaktion skal igangsætte en nulpunktsjustering, som systemet udfører ved at læse gennemsnittet af de første 20 indhentede værdi fra transduceren.

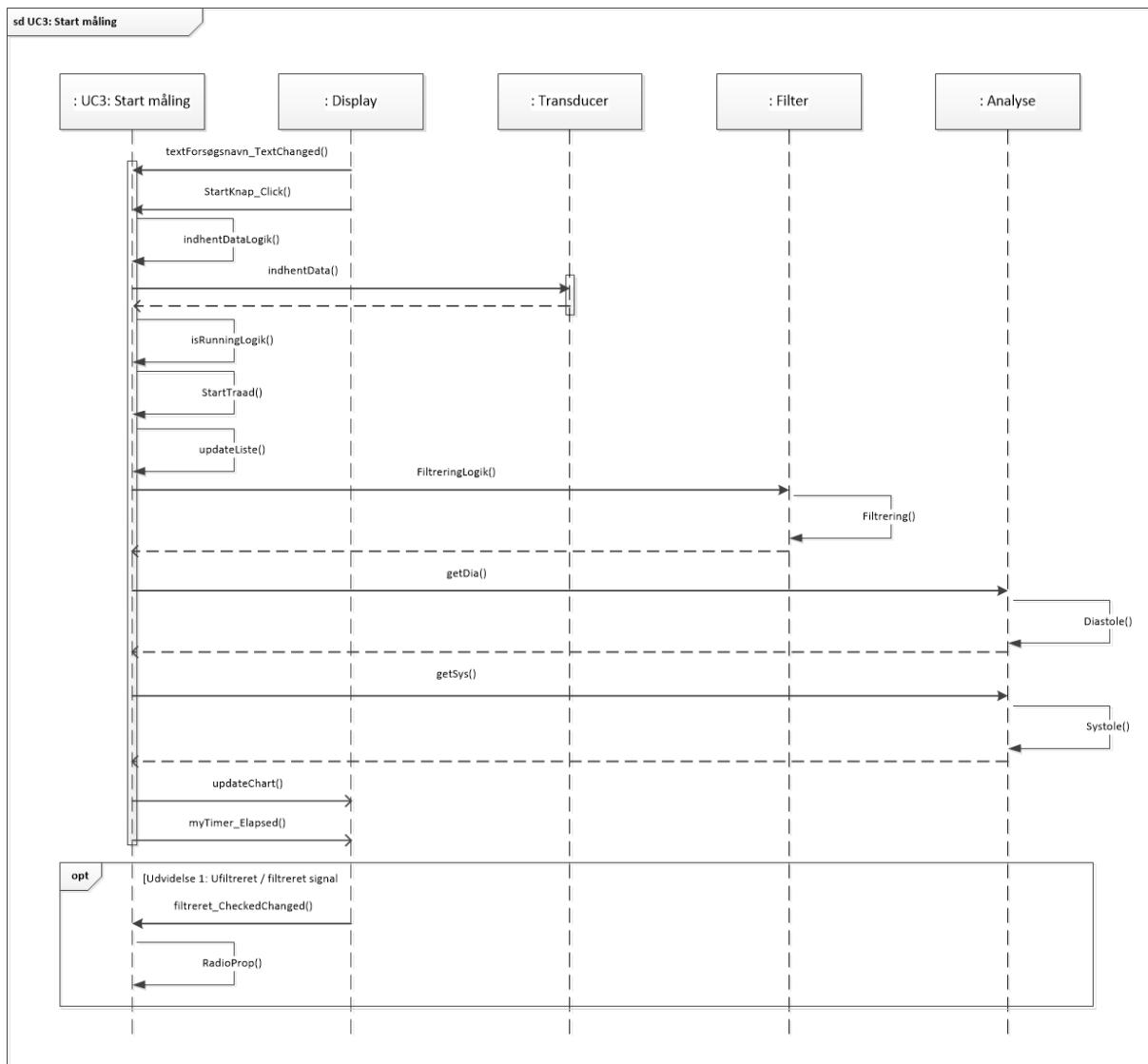
### Use Case 2



Figur 3.18: Sekvensdiagram for Use Case 2

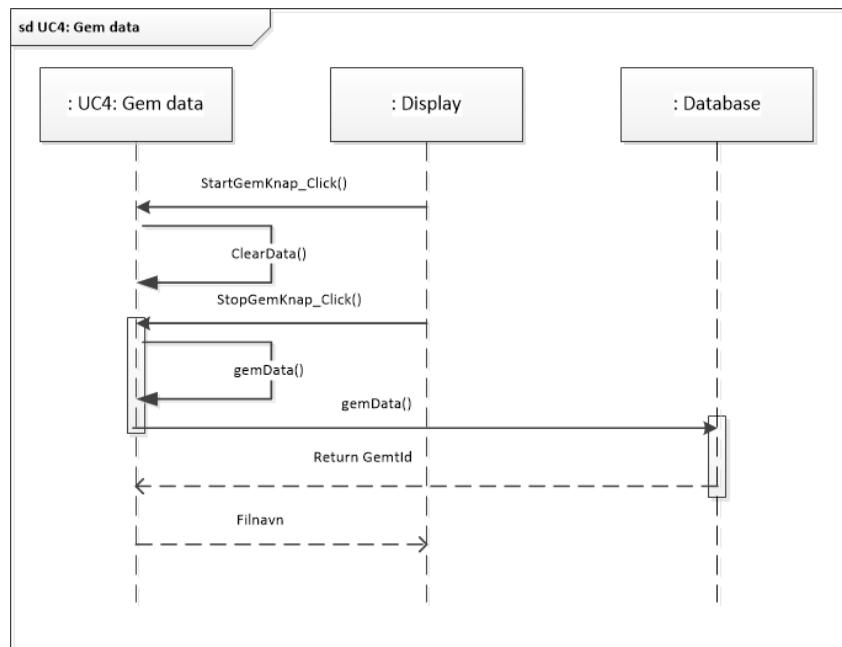
Af diagrammet på figur 3.18 ses det at kalibreringen skal implementeres simpelt i softwaren, hvor kaliberingskoefficienten hentes frem så den kan ganges på samtlige indhentede samples.

### Use Case 3



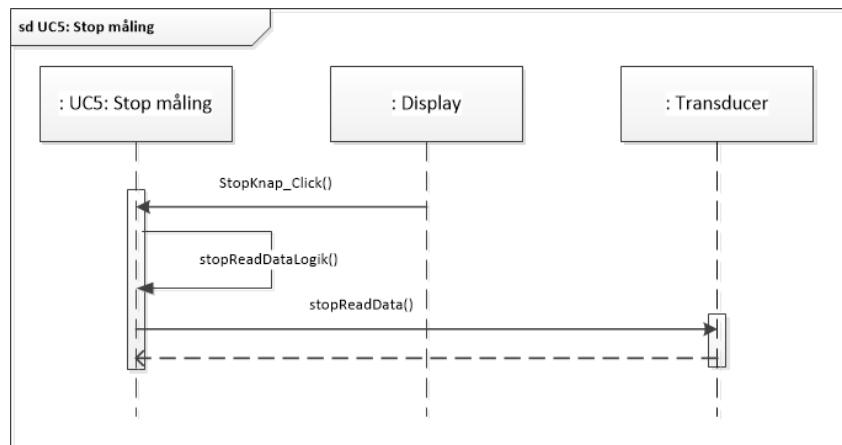
Figur 3.19: Sekvensdiagram for Use Case 3

På diagrammet i figur 3.19 ses det at display-, transducer-, analyse- og filterklassen vil komme i spil. Her modtages der besked ved indtastning af Forsøgsnavn og tryk på Start Måling-knap på display om, at signaldata fra transduceren skal hentes ind i systemet. Herefter foretages filtrering af signalet, samt visning af signal i graf, systoliske, diastoliske og puls værdier på displayet. Use Casen indeholder en udvidelse hvor filtrering af signal ikke ønskes foretaget, dette er vist ved en Optional nederst på figur 3.19.

**Use Case 4**

Figur 3.20: Sekvensdiagram for Use Case 4

Ovenstående diagram viser at det kræver, at der trykkes på Start Gem-knap på display, for at få gemt signalets rådata. Hvorefter systemet skal gemme det fremadrettede rådata indtil der trykkes på Stop Gem-knap.

**Use Case 5**

Figur 3.21: Sekvensdiagram for Use Case 5

Ved stop af en måling ses det at Forsker trykker på Stop Måling-knap på display, hvorefter indhentning af data fra Transducer-blokken stoppes.

### 3.2.2 Implementering

#### Indledende implementeringsovervejelser

På baggrund af designfasen for softwaren kan implementeringen påbegyndes. Softwaredesignet viser at systemet skal implementeres med en GUI-applikation, hvor aktøren kan interagere med systemet. Derudover er det kendt at softwaren skal indeholde en række klasser, hvori funktionaliteter som kalibrering, nulpunktsjustering, digitalt filter og indhentning af systolisk, diastoliske og puls værdier skal placeres. I det følgende beskrives de overvejelser, der er gjort i forhold til implementering af disse funktionaliteter og softwaresystemet.

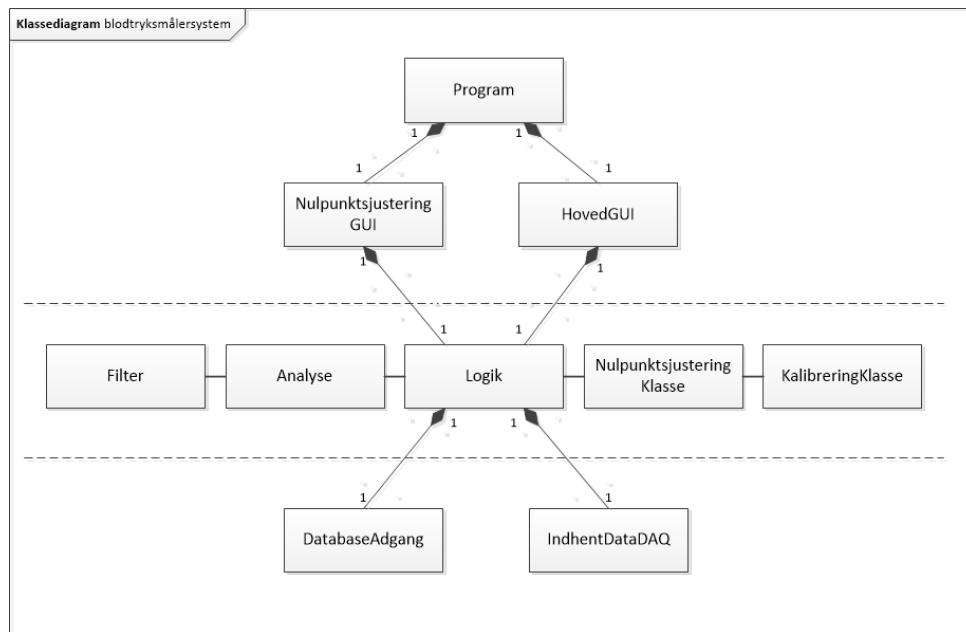
Implementeringen af softwaren sker i Visual Studio 2013 i sproget C#. Dette er valgt da programmet er et godt værktøj til at arbejde med GUI-applikationer, samt til håndtering af tråde og trådkommunikation. Tråde benyttes i softwaren da systemet, der skal implementeres, er et eventdrevet system. Det vil sige, at systemet skal kunne håndtere mange handlinger på en gang. Handlingerne igangsættes af events, der kommer af aktørens interaktion med systemet. Trådkommunikationen fungerer således, at en tråd kan sende et signal ud, som andre tråde kan reagere på.

Det er valgt kun at udarbejde aktivitetsdiagrammer for metoder, hvor det menes at skabe et bedre overblik. Flere af metoderne er simple og derfor er et aktivitetsdiagram ikke nødvendigt.

#### Klasse implementering

På baggrund af designmodellerne er det besluttet at opbygge systemkoden efter principperne i en trelagsmodel[7]. Trelagsmodellen indeholder et præsentations-lag, et logik-lag og et data-lag. Præsentations-laget består af de klasser som systemets aktører har tilgang til. Logik-laget er det analyserende lag. Det er i dette lag at signalet behandles. Logik-laget har tilgang til de andre lag som det eneste. Det betyder at præsentations-laget og data-laget ikke kan kommunikere sammen, derved skal denne kommunikation foregå gennem logik-laget. Data-laget er tilgangen til den implementerede database og til indhentning af blodtrykssignalet fra hardwaren.

Fordelen ved trelagsmodellen er, at den skaber et godt overblik i koden, og giver en kode med lav kobling, da hver enkelt klasse har hvert sit specifikke ansvar. Hvilket gør at koden er let at vedligeholde og ændre, hvis funktionaliteter ønskes opbygget anderledes. Et overordnet klassediagram over systemet er udarbejdet på baggrund af præcisering af applikationsmodellen, se figur 3.22. Hvilke metoder hver enkelt klasse indeholder kan ses i Bilag nr. 13.

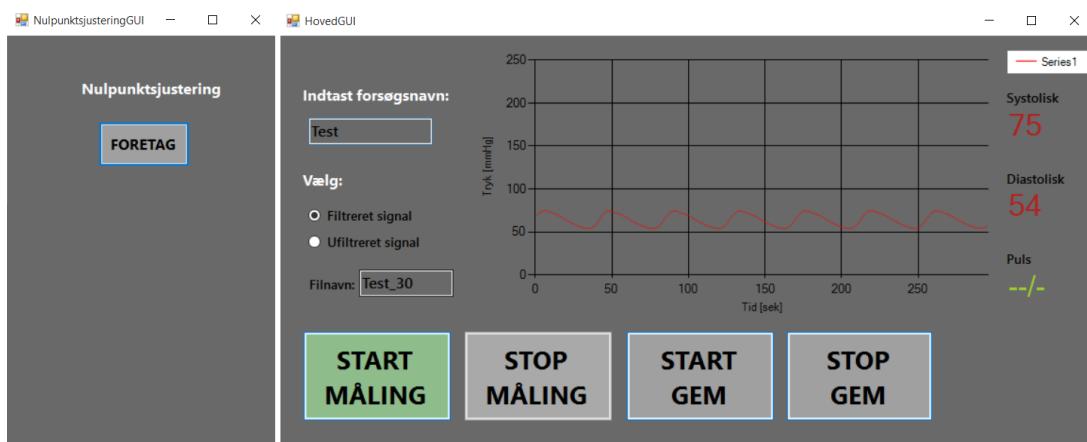


Figur 3.22: Klassediagram

### Brugergrænseflade

Displayet (GUI) er Forskerens indgang til systemet. Derfor er det vigtigt at den er opbygget efter forskerens logik. Til at klarlægge dette er principperne om en god brugergrænseflade taget i mente. Brugen af disse kommer til udtryk ved, at det tydeligt fremgår af hver knap eller label, hvad dens formål er, samt at størrelsen af det enkelte komponent er tilstrækkelig stor til at det ikke er til at overse. Komponenterne på displayet er logisk placeret, det vil sige, at de dele som forsker først skal forholde sig til og eventuelt udfyldte er placeret i venstre side af display.

Det er et krav at Forsker indtaster et Forsøgsnavn inden en måling kan startes, derfor er komponenterne implementeres således at knappen "Start Måling" først bliver aktiveret, når der er indtastet noget i tekstdoboksen til Forsøgsnavn. Systoliske, diastoliske og puls værdi er placeret efter hvilken rækkefølge der typisk ses på standard blodtryksapparater.



Figur 3.23: NulpunktsjusteringGUI og HovedGUI

Af figur 3.23 ses det, at grafen er en væsentlig del af displays brugergrænseflade. Grafen

implementeres som en Windows Form komponent. Det vælges at få vist signalet som en kurve, og førsteaksen indstilles til samples fra 0-300, og andenaksen til værdier fra 0 til 250 mmHg, hvilket er givet i kravspecifikationen.

### **Observer - Strategy**

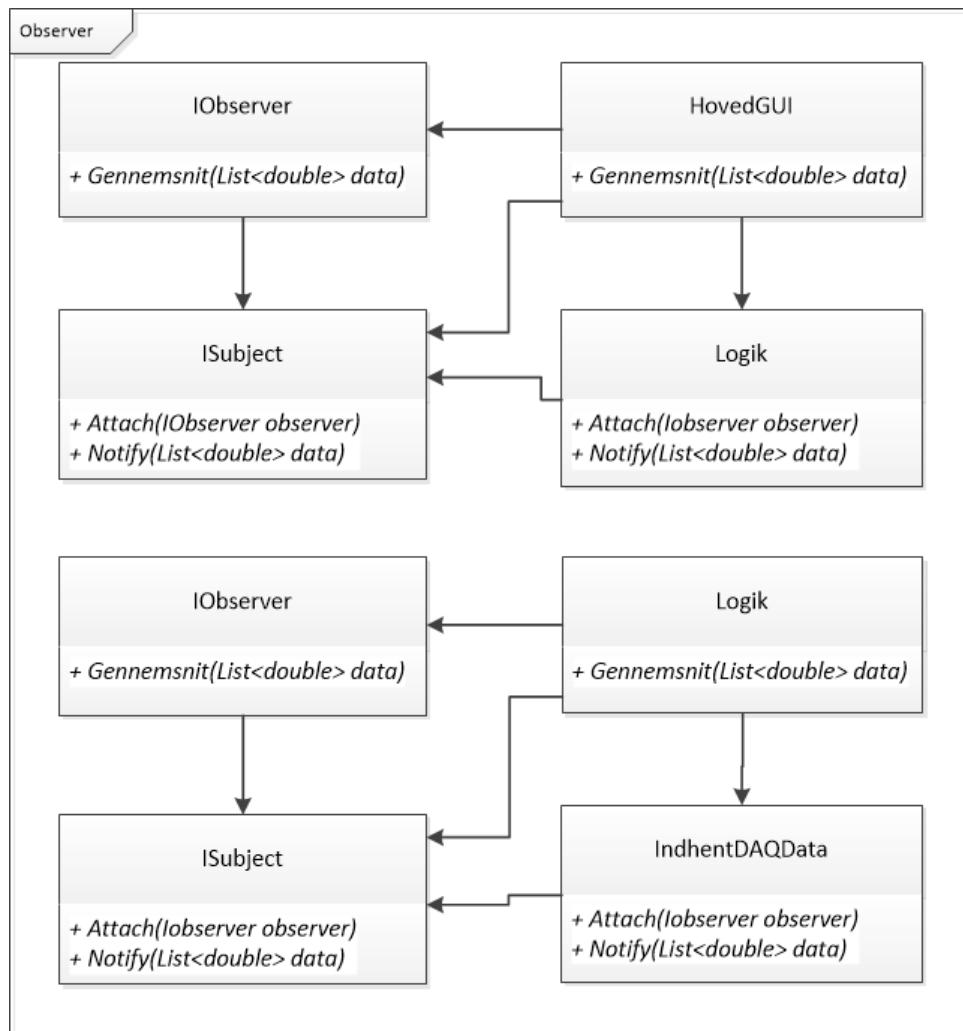
Observer og Strategy er to programmeringsmønstre. Der i samarbejde med hinanden er gode til at håndtere en overførelse af data fra et lag til et andet. Det er valgt at opbygge softwarekoden efter disse to mønstre. Observer definerer et "en til mange" relation mellem objekter således, at en ændring i et objekts tilstand medfører, at de mange objekter informeres om ændringer og dermed opdateres automatisk.

Dette implementeres ved at oprette to interfaces IObserver og ISubject. Disse interfaces placeres i deres eget namespace, som alle lag kan tilgå, samt gør det muligt for de nødvendige klasser at arve fra disse interfaces.

I ISubject placeres de generelle metoder Notify() og Attach(), hvis ansvar er at informere og flytte data fra en klasse til en anden klasse, når de kaldes i en Subject-klasse som fx logikklassen. IObserver indeholder metoden, der kaldes i Observer når en Notify() fra Subject og ISubject modtages.

Mønstret benyttes både mellem data-laget og logik-laget, hvor data-laget fungerer som Subject og logik-laget er Observer, samt mellem logik-laget og præsentations-laget, hvor logik-laget er Subject og præsentations-laget er Observer.

Mønstret opbygges som en push, hvilket vil sige, at når Subject har ny data klar til at sende op til Observer, kaldes metoden Notify() med data'en som parameter og dette sendes op til Observer, via ISubject og IObserver. Således fortsætter koden med at arbejde så længe ny data ønskes flyttet op. Skematisk er det, i dette projekt givet ved, hvor de relevante metoder i forhold til mønstret er medtaget, se figur 3.24.



Figur 3.24: Observer mønstre

Strategy mønstret indkapsler algoritmer og gør dem udskiftelige med hinanden. Det vil sige at en metode oprettes i et interface. Klasser vil arve fra dette interface. Afhængig af hvem, der bruger metoden vil den blive overskrevet i klassen og den nødvendige funktion tilføjet. I samarbejde med Observer-mønstret bruges det ved, at Subject arver fra ISubject, og Observer arver fra IObserver. I projektet blev mønstrene i første omgang benyttet fra logik-laget til præsentationslaget i forbindelse med at overføre data til visning i graf. Men undervejs viste det sig nødvendigt også at implementere mønstrene fra data-laget til logik-laget, således at det kan kontrolleres, hvor stor en mængde data, der sendes op af gangen.

### Samplefrekvens

Samplefrekvensen er som krav givet til 1000 Hertz. Hvilket svarer til at systemet modtager 1000 samples i sekunder. Varigheden af en sample er givet ved:

$$\frac{1}{f_s} = \frac{1}{1000} = 0.001 \text{ sek} \quad (3.14)$$

Det har vist sig under arbejdet med softwaren, at grafen ikke kan følge med, når den modtager så mange målinger i sekundet. Derfor er det valgt at skære i antallet af målinger pr. sekund, der

skal viderebearbejdes i logik-laget og udskrives i præsentations-laget. Antallet skæres ned til 50 målinger pr. sekund. Dette gøres ved at gennemsnittet af 20 målinger efter hinanden bestemmes, hvorefter gennemsnitsværdien returneres og gemmes i en liste, der sendes videre i systemet. Herefter findes gennemsnittet af de næste 20 målinger og således fortsætter det.

### Nulpunktsjustering

Formålet med en nulpunktsjustering er at flytte signalets offset op eller ned, så det atmosfæriske tryk altid er placeret ved 0 V på outputsignalet. Dette gøres ved at åbne for den tilsluttede transducer, så det atmosfæriske tryk måles. Ud fra denne værdi kan justeringsfaktoren bestemmes, hvor  $x$  er det målte atmosfæriske tryk i Volt modtaget gennem DAQ'en:

$$faktor_{jus} = 0 - (x) \quad (3.15)$$

Af ligningen ses det, at justeringsfaktoren både vil kunne blive positiv og negativ, afhængig af om offsetværdien skal rykkes op eller ned for at blive placeret i nul. Optimalt set vil det atmosfæriske tryk være en konstant værdi ved den samme måling, men det opleves, at der er en smule støj på signalet og derfor vil den målte værdi være en tilnærmelse af det atmosfæriske tryk. Systemet ønskes nulpunktsjusteret for at sikre, at alle de målte blodtrykssignaler har samme udgangspunkt. Hvilket gør at målingerne kan sammenlignes.

Systemet foretager nulpunktsjusteringen ved at retunerer gennemsnittet af de første 20 målinger fra DAQ'en, når der trykkes på knappen Foretag. Denne værdi er justeringsfaktoren, der lægges til samtlige samples i det indhentede blodtrykssignal.

### Kalibrering

Ved kalibrering ønskes det at bestemme hardwarens visningsfejl. I dette projekt betyder det, at kalibreringskoefficienten fra volt til millimeter kvisksølv bestemmes. Denne bestemmes ved at tilkoble en væskesøjle til systemet. Væskesøjlen fyldes med vand til markering, så den vil give et kendt tryk (mmHg). Herefter kan output i volt fra hardwaren måles. Kalibreringskoefficienten er givet ved:

$$koeff = \frac{x[\text{mmHg}]}{y[\text{Volt}]} \quad (3.16)$$

$x$  angiver trykket fra væskesøjlen, denne hardcodes til 50 mmHg. Værdien for  $y$  angiver det målte spændingsoutput på hardwaren. Optimalt set er kalibreringskoefficienten givet ved:

$$\frac{250[\text{mmHg}]}{5[\text{V}]} = 50 \quad (3.17)$$

hvor 250 mmHg er det maksimale blodtrykssystemet kan måle og 5 V er maks spændingen i volt. Grafisk vil det se ud som vist på figur 3.9 under Modultest for hardware. Af figur 3.9 kan det aflæses at den optimale outputspænding ved 50 mmHg er 1 V. Kalibreringskoefficienten skal ganges på samtlige sampleværdier, der kommer fra DAQ'en, som ønskes udskrevet på graf i display.

Kalibreringen implementeres i softwaren ved brug af konfiguration. Forskeren beregner omsætningsværdien ud fra ligning 3.16. Resultatet af denne beregning indtaster Forsker i konfigurations XML-filen under App.settings. XML-filen kan tilgås uden opstart af systemet, derfor bliver kalibreringen uafhængig af, hvornår systemet kører og kalibreringen kan dermed foretages på et vilkårligt tidspunkt. Værdien der ændres i XML-filen er den tilhørende "Value" til "KalibreringsKoefficient". Den er markeret med en grøn firkant på figur 3.25.

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <configuration>
3   <startup>
4     <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
5   </startup>
6   <appSettings>
7     <add key="KalibreringsKoefficient" value="50"/>
8   </appSettings>
9 </configuration>

```

Figur 3.25: Konfigurations XML-fil

Metoden Kalibrering() i Kalibreringsklassen, som er en del af logik-laget, læser "KalibreringsKoefficienten" fra konfigurations-filen, hver gang kalibreringskoefficienten skal ganges på et signal.

Det er vigtigt at pointere, at nulpunktsjusteringsfaktoren lægges til samtlige værdier i signalet før kalibreringskoefficienten ganges på. Dette udføres i kodens logik-lag.

### Digitalt Filter

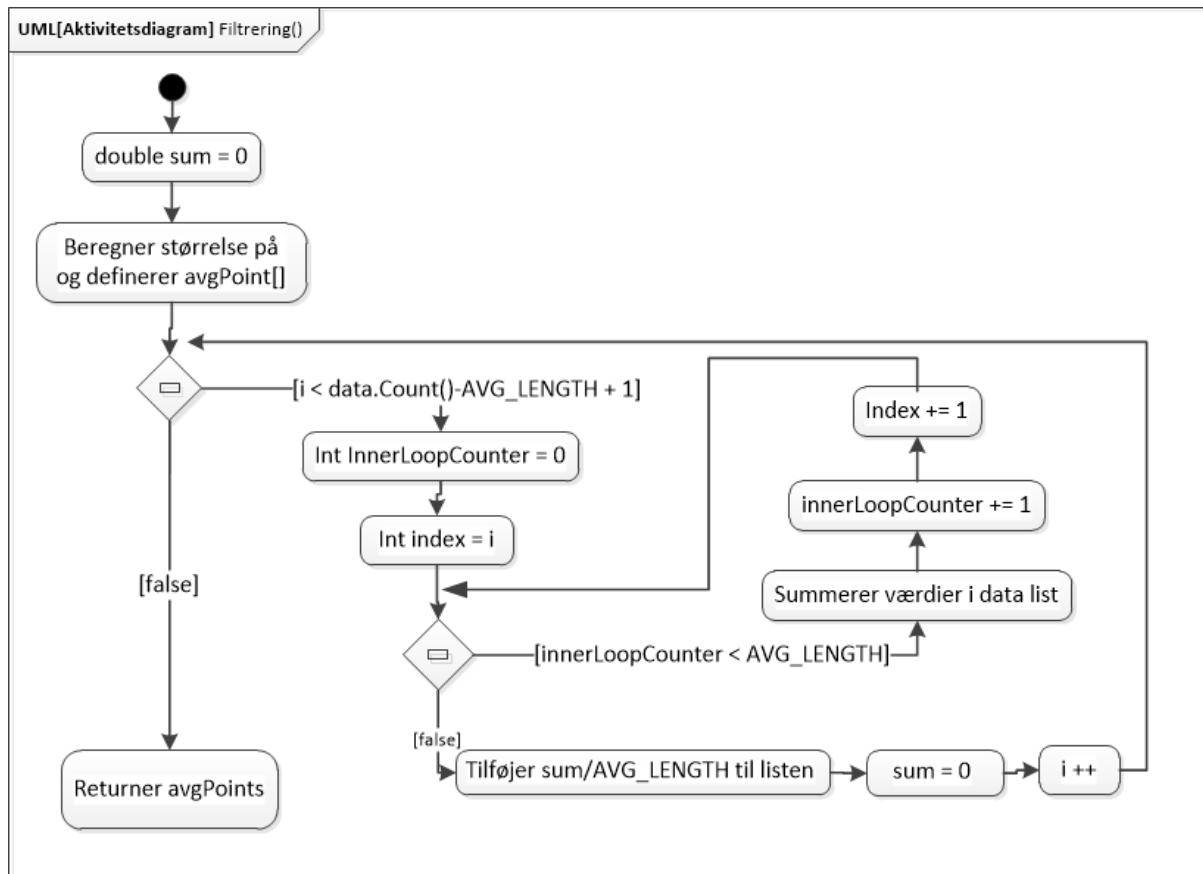
[8] Formålet med implementering af et digitalt filter er at fjerne støj fra det indhentede signal. Dette gøres ved at udglatte signalet. Til dette kan en række forskellige filtre benyttes. I projektet er det valgt at implementere et glidende middelværdifilter (moving average filter).

Fordelen ved dette filter er, at det er simpelt at forstå og det er optimalt at bruge på signaler i tidsdomænet. Skulle signalet være vist i frekvensdomænet ville valget have faldet på et andet filter.

Det glidende middelværdifilter fungerer ved midling af en række punkter fra inputsignalet for, at frembringe hvert punkt i outputsignalet. Hvilke punkter, der tages fra inputsignalet, vil flytte sig én plads for hvert beregnet outputpunkt, heraf kommer den glidende effekt. Matematisk er filteret givet ved:

$$y[i] = \frac{1}{M} \cdot \sum_{j=0}^{M-1} x[i+j] \quad (3.18)$$

Hvor  $x[]$  er inputsignalet,  $y[]$  er outputsignalet og  $M$  er antallet af punkter, der benyttes i det glidende middelværdifilter. Denne beregning benytter sig udelukkende af punkter placeret på den samme side af output samplenummeret, hvilket vil føre til en relativ forskydning mellem input og output.  $M$  sættes til en længde på fem. Implementeringen af filteret er vist i et aktivitetsdiagram på figur 3.26.



Figur 3.26: Aktivitetsdiagram af metoden *Filtrering()*

Måden hvorpå filtret er implementeret gør, at der ikke sker en filtrering af de første fire samples, det ses i følgende kodeudsnit. `AVG_LENGTH` er defineret til fem, og er mængden af punkter, der benyttes i filteret. I ligning 3.18 svarer `AVG_LENGTH` til  $M$ .

```

1 reference
public List<double> Filtrering(List<double> data)
{
    double sum = 0;
    List<double> avgPoints = new List<double>();
    for (int i = 0; i < data.Count() - AVG_LENGTH + 1; i++)
    {

```

Figur 3.27: Udsnit af koden til det glidende middelværdifilter

Det ses, at der skal være minimum fem samples i `data.Count` først at listen `avgPoints` oprettes. Det er en begrænsning, som er vigtig at være opmærksom på, men som accepteres da de første fire samples ved visning i graf er kørt så hurtigt igennem. Det skaber ikke en begrænsning for Forsker. Optimalt set vil der sættes en begrænsning på filtret således, at når første måling modtages vil gennemsnittet findes af en sample, dernæst af to samples, tre samples osv. Indtil der er fem samples og gennemsnittet vil altid bestemmes af de fem seneste samples.

Systemet gør det muligt for Forsker selv at vælge om signalet ønskes vist filtreret eller ufiltreret. Dette vælges på brugergrænsefladen. Vælges visning af det ufiltrede signal, sendes det indhentede

signal ikke gennem det digitale filter. Det er muligt at skifte mellem filtreret og ufiltreret signal, mens systemet kører. I det tilfælde skifter hele det viste signal til det valgte, da alt data i listen, der indhentes dermed skifter.

## Analyse

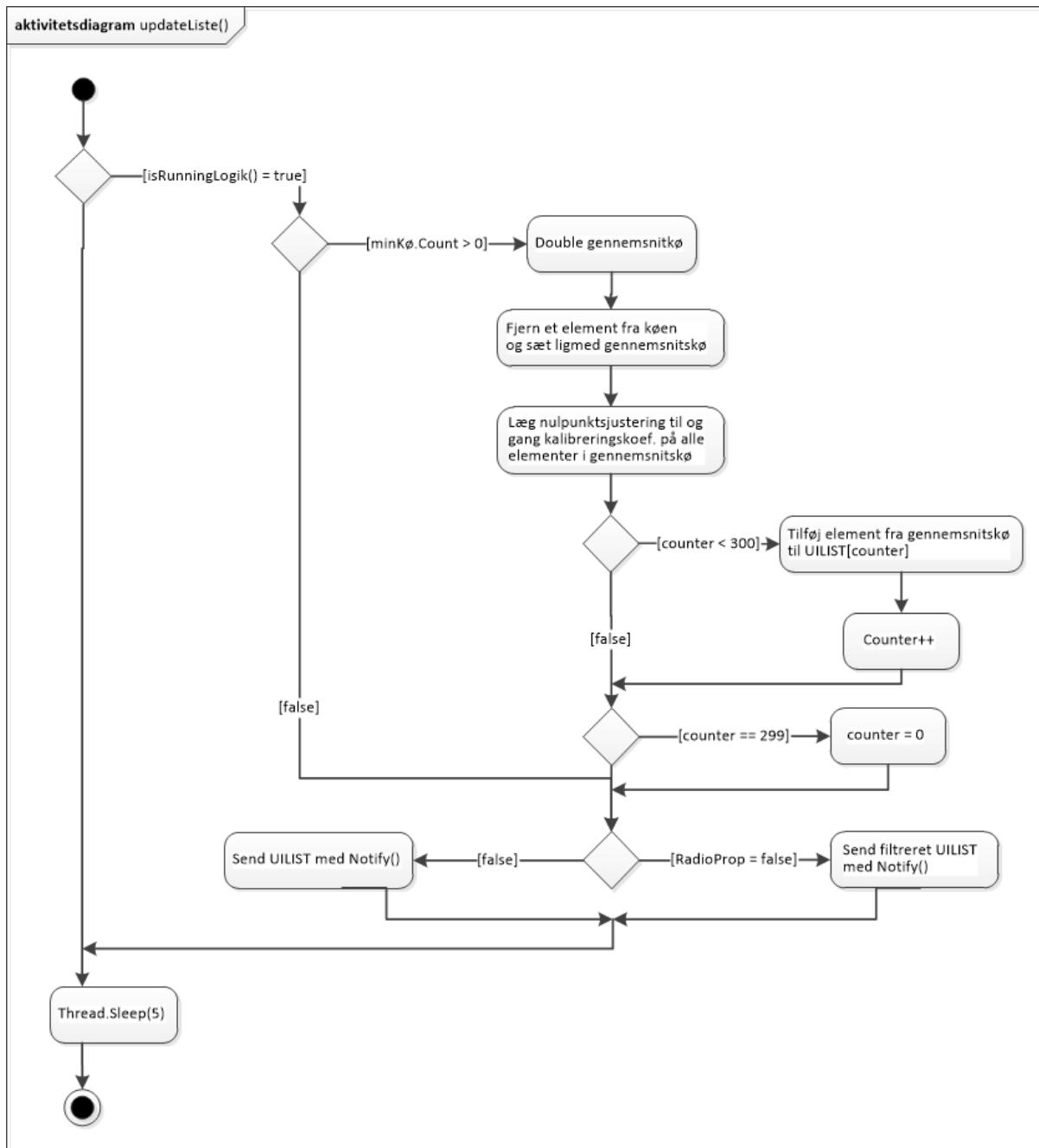
Analyse dækker over indhentningen af de systoliske, diastoliske og puls værdi ud fra blodtrykssignalet. Dette er implementeret i en klasse kaldet Analyse. Heri er placeret metoder for henholdsvis systole og diastole. I en blodtrykskurve er den systoliske værdi givet ved maksimum på kurven og den diastoliske er givet ved minimumsværdien på kurven.

Metoderne bestemmer derfor den maksimale værdi og den mindste værdi i listen, der medtages som parameter til metoderne. Listen, der bruges som parameter er UILIST, som indeholder 300 tal ad gangen. UILIST er listen, der sendes fra logik-laget til præsentations-laget med de behandlede data, som vises i grafen. I præsentations-laget er implementeret en timer[9], der håndterer de systoliske og diastoliske værdier i display skal opdateres hvert 3. sekund. I løbet af 3 sekunder vil der være gennemløbet 3-5 blodtryksperioder, afhængig af pulsfrekvensen. Dermed vil samtlige systoliske og diastoliske værdier ikke blive udskrevet. Intervallet på 3 sekunder er valgt, da det er passende tid til at aflæse den pågældende værdi.

I forhold til implementering af puls er der gjort en række overvejelser om mulige løsninger. Puls er defineret ved slag pr. minut og på en pulsperiode vil der være en systole og diastole. Pulsen må derfor kunne bestemmes ved at tælle antallet af systoliske værdier på 6 sekunder. Antallet ganges med 10 for at få den rette enhed. Udfordringer er dog opstået i forhold til at kunne bestemme præcist, hvornår der er gået 6 sekunder i programmet. En anden mulighed er også at bestemme pulsen ved at finde antallet af samples mellem to systoliske værdier. Omregnes samples til sekunder og ganges op til et minut, må dette være lig med Måleobjektets øjeblikkelige puls. Det er ikke lykkedes at omsætte overvejelserne til kode, og pulsen er dermed ikke blevet implementeret ved projekts aflevering.

## Beregnings metode

Metoden updateListe() er den vigtigste metode i logik-laget. Den er placeret i logik-klassen. Metoden har til ansvar at tage højde for nulpunktsjustering og kalibrering ved beregning. Derudover har den til ansvar at tilføje data til en liste med 300 pladser, der kan vises i grafen på displayet. Til sidst checker metoden om radiobutton for filtreret eller ufiltreret signal er valgt og sender listen gennem filteret, hvis filtreret signal er valgt inden listen med Notify() sendes til præsentationslaget. Metodens aktivitetsdiagram er givet ved:



Figur 3.28: Aktivitetsdiagram for metoden updateListe()

## Database

I systemet er der implementeret en lokal Database. Databasen er oprettet gennem hosten webhotel10.ihb.dk. Formålet med Databasen er at lagre det målte blodtrykssignals rådata. Det er valgt at implementere Databasen af typen SQL, da denne database-type indeholder de funktioner, som er nødvendige for dette system. Data gemmes i tabeller i Databasen. Indledningsvis for at oprette den nødvendige tabel defineres en type til hver værdi. SQL-koden til oprettelse af tabel er vist på figur 3.29.

```

1  CREATE TABLE [db_owner].[SEMPRJ3] (
2      [Forsøgsnavn]      NVARCHAR (20)    NOT NULL,
3      [Id]                BIGINT          IDENTITY (1, 1) NOT NULL,
4      [Datostempel]       DATETIME        NOT NULL,
5      [Blodtryksmåling]  VARBINARY (MAX) NOT NULL,
6      PRIMARY KEY CLUSTERED ([Id] ASC)
7  );

```

Figur 3.29: SQL-kode til oprettelse af tabeller i database

Forsøgsnavn referer til det Forsøgsnavn, der indtastes i GUI ved påbegyndelse af en ny måling. Dette er af typen NVARCHAR(20), hvilket betyder at forsøgsnavnet maksimalt kan være 20 tegn langt. Id er defineret som primær nøgle, det betyder at denne er unik for hver enkelt sekvens i Database, og Id der vil refereres til mellem tabeller i Databasen, hvis flere tabeller var nødvendigt. Et blodtrykssignal indeholder en stor mængde datapunkter, derfor gemmes signalet i en VARBINARY, hvor en række binære datapunkter gemmes som en enkelt enhed i Databasen. Dette er valgt for at spare på pladsen i Databasen. Denne type besværliggør at få vist, hvilke værdier blodtrykssignalets datapunkter består af.

Databasen er implementeret således, at flere sekvenser af den samme måling kan gemmes uden at systemet skal startes forfra. Dette er smart for Forsker, hvis der testet flere ting på det samme signal.

### 3.2.3 Modultest

I teorien burde man, for at teste, skrive et testprogram, som tester én metode af gangen. Dette er vi dog ikke i stand til, derfor er det valgt at gøre test af softwaren an vha. debugging. Et modul er forsøgt færdiggjort og testet, før det er blevet sat sammen med andre moduler. På den måde er der blevet testet løbende gennem udviklingsprocessen.

Koden, der bliver kigget igennem, er fra når der bliver trykket Start Måling på GUI og til at der udskrives en graf på charten. Der er brugt debugging i Visual Studio for at sikre koden bliver udført i korrekt rækkefølge. En stor del af koden i IndhentDAQData-klassen er taget fra et eksempel fra National Instruments. Se Bilag nr. 16.

Gennemgangen af hvad der sker i Datalaget, nærmere IndhentDAQData-klassen, hvor oprettelsen til DAQ'en, samt sampleliste sker, vil blive overfladisk. Koden i IndhentDAQData-klassen er tilpasset solution. Her er der tale om en samplerate og samples pr. channel.

Samples pr. channel er testet igennem ved at debugge på det kode, der tilføjer sampleværdier til råData-listen. Her kan der checkes om listens count er på 10 eller 20, hvis samples pr. channel er henholdsvis 10 eller 20.

### Fra DAQ til Logik

Programmet startes ved tryk på Start Måling-knappen på GUI. Det starter følgende event, figur 3.30.

```
1 reference | Matmero, 4 days ago | 2 authors, 3 changes
private void StartKnap_Click(object sender, EventArgs e)
{
    logik.indhentDataLogik();

    if (logik.isRunningLogik())
    {
        logik.StartTraad();
    }
    GemKnap.Enabled = true;
}
```

Figur 3.30: Kode for eventet StartKnapClick()

Følgende metoder bliver kaldt i opstillet rækkefølge:

```
1 reference | AnneBH, 30 days ago | 1 author, 1 change
public void indhentDataLogik()
{
    ≤1ms elapsed
    DAQdata.indhentData();
}
```

Figur 3.31: Kode for Logik metoden indhentDataLogik()

Herfra starter data acquisition i IndhentDAQData-klassen.

IndhentData() opretter en virtual channel, samt sørger for kald til InitializeDataTable() og håndterer tråde. Al kode, metoder og klasser kan ses i Bilag nr. 14 "Softwarekode pdf".

```
public void indhentData()...
```

Figur 3.32: Kode for IndhentDAQData metoden indhentData()

```
public void InitializeDataTable(AIChannelCollection channelCollection, ref DataTable data)...
```

Figur 3.33: Kode for InitializeDataTable()

```
// Denne metode kalder dataToDataTable og sender data og en reference med som parameter.
// Samt sker der asynkron callback, hvis DAQ'en kører.
1 reference
private void AnalogInCallback(IAsyncResult ar) //Asynkron callback...
```

Figur 3.34: Kode for IndhentDAQData metoden AnalogInCallBack()

```
// Denne metode tilføjer alle samples i sourceArray over i råData og kalder Notify der "skubber" sampleværdierne op til Logik laget.
1 reference
private void dataToDataTable(AnalogWaveform<double>[] sourceArray, ref DataTable dataTable)
{
    // Iterate over channels
    int currentLineIndex = 0;
    List<double> råData = new List<double>();

    foreach (AnalogWaveform<double> waveform in sourceArray)
    {
        for (int sample = 0; sample < waveform.Samples.Count; ++sample)
        {
            råData.Add(waveform.Samples[sample].Value);
        }

        Notify(råData);

        currentLineIndex++;
    }
}
```

Figur 3.35: Kode for IndhentDAQData metoden DataToDataTable()

I Logik-laget bliver der kaldt DAQData.Attach(this), derved bliver Logik objektet tilføjet som observer til subjectet IndhentDAQData-klassen.

```
public void Attach(IObserver observer)
{
    observers.Add(observer);
}
```

Figur 3.36: Kode for metoden Attach(), denne bruges på samme måde i klasserne IndhentDAQData som i Logik

Med Logik-objektet som liggende i observers-listen, kaldes der metoden Gennemsnit med graf som parameter.

```
//Her bruges der en "Push-Notify", som skubber en liste op til logiklaget.
5 references
public void Notify(List<double> graf)
{
    foreach (IObserver obs in observers)
    {
        obs.Gennemsnit(graf);
    }
}
```

Figur 3.37: Kode for metoden Notify() i IndhentDAQData klassen

Metoden Gennemsnit bliver kaldt i Logik-klasse. Her bliver listen lagt i en kø (queue), hvorfra metoden updateListe henter alle samples fra, med metoden queue.Dequeue(). Metoden updateListe() kører på en tråd for sig selv.

Her slutter gennemgangen af, hvordan sampleværdierne bliver skubbet op til Logik-klassen.

```
public void Gennemsnit(List<double> graf)
{
    ≤1ms elapsed
    databasetal = graf;
    minKØ.Enqueue(Convert.ToDouble(graf.Average()));
}
```

Figur 3.38: Kode for metoden Gennemsnit() i Logik-klassen

Fra Logik til GUI I starten blev metoden StartTraad() kaldt hvis IsRunning = true.

```
1 reference | Matimero, 4 days ago | 2 authors, 6 changes
public void StartTraad()
{
    updateUI.Start();
}
```

Figur 3.39: Kode for Logik metoden StartTraad()

Metoden updateListe() kører derved på tråden updateUI.

```
updateUI = new Thread(() => updateListe());
```

Figur 3.40: Kode for tråden updateUI

```
private void updateListe()
{
    while (isRunningLogik())
    {
        if (minKø.Count > 0)
        {
            double gennemsnitKø = minKø.Dequeue();
            gennemsnitKø = (gennemsnitKø + beregnetNværdi) * kalibreringKoef;

            if (counter < 300)
            {
                UILISTE[counter] = gennemsnitKø;
                counter++;
            }
            if (counter == 299)
            {
                counter = 0;
            }
        }
        if (RadioProp == false)
        {
            Notify(FiltreringLogik(UILISTE));
        }
        else
        {
            Notify(UILISTE);
        }
    }
    Thread.Sleep(5);
}
```

Figur 3.41: Kode for logik metoden updateListe()

Hvis der fra HovedGUI-klassen vælges true eller false til en Radiobutton, skal signalet være filtreret eller ufiltreret. I tilfælde af filtrering, sendes UILISTE først gennem Filter-klassen, som parameter, før den bliver returneret til Logik-klassen.

```

public List<double> FiltreringLogik(List<double> data)
{
    FiltreretListe = FilterObj.Filtrering(data);
    return FiltreretListe;
}

```

Figur 3.42: Kode for metoden *FiltreringLogik()*

```

class Filter
{
    private const int AVG_LENGTH = 5;
    1 reference | Matimero, 5 days ago | 1 author, 1 change
    public Filter()
    {

    }
    1 reference | Matimero, 5 days ago | 1 author, 1 change
    public List<double> Filtrering(List<double> data)
    {
        double sum = 0;
        List<double> avgPoints = new List<double>();
        for (int i = 0; i < data.Count() - AVG_LENGTH + 1; i++)
        {
            int innerLoopCounter = 0;
            int index = i;
            while (innerLoopCounter < AVG_LENGTH)
            {
                sum = sum + data[index];
                innerLoopCounter += 1;
                index += 1;

            }
            avgPoints.Add(sum / AVG_LENGTH);
            sum = 0;
        }
        return avgPoints;
    }
}

```

Figur 3.43: Kode for klassen *Filter()*. Her bliver summen af 5 værdier divideret med 5

Fra metoden *updateListe()* bliver *Notify()* kaldt, med *UILISTE* som parameter, der bliver sendt op til *HovedGUI*-klassen.

```

public void Notify(List<double> data)
{
    foreach (IObserver obs in observers)
    {
        obs.Gennemsnit(data);
    }
}

```

Figur 3.44: Kode for metoden *Notify()* i *Logik*-klassen

I *HovedGUI*-klassen bliver *Gennemsnits*-metoden kaldt via Observer mønsteret.

```

public void Gennemsnit(List<double> graf)
{
    guiliste = graf;

    updateChart();
}

```

Figur 3.45: Kode for metoden i GUI Laget Gennemsnit()

Til sidst bliver grafen (charten) i GUI opdateret ved at metoden updateChart() konstant kalder sig selv.

```

private delegate void UpdateUICallback();
2 references | Matimero, 4 days ago | 2 authors, 12 changes
private void updateChart()
{
    if (this.InvokeRequired)
    {
        UpdateUICallback d = new UpdateUICallback(updateChart);
        this.Invoke(d);
    }
    else
    {

    }
}

```

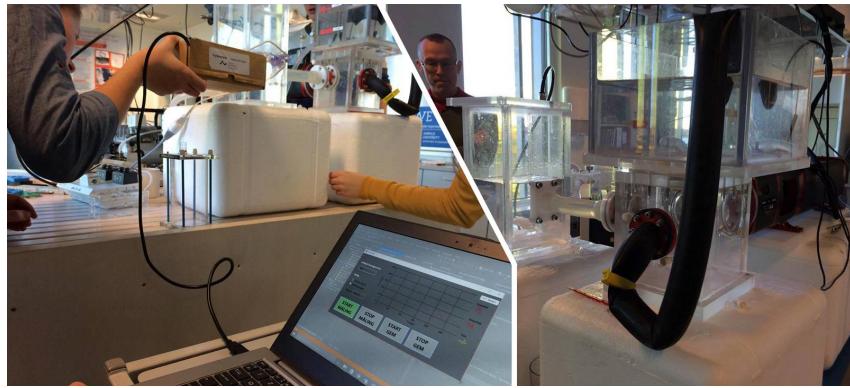
Figur 3.46: Kode for metoden i GUI Laget updateChart()

### 3.3 Integrationstest

Til sidst i projektforløbet blev en integrationstest[10] udført. En integrationstest laves primært for at teste om softwaren fungerer korrekt, og om enhederne/modulerne deri anvender hinanden. Testen retter sig mod afprøvning af det komplette program, med de eksterne systemer, i dette tilfælde sammen med hardwaren.

Softwareen er langsomt blevet sammensat af de forskellige enheder som fremvisning af graf, indhentning af data, kalibrering, nulpunktsjustering, digitalt filter og at gemme. Hver gang én enhed er færdig, er den blevet testet, hvorefter en ny færdig enhed er blevet sat på osv. Tilsidst er der blevet udført en test hvor alle enheder sættes sammen, software og hardware, og der testes derpå. Testen kan sammenlignes med en "Big Bang" test, da det var første kan vi satte den færdige software sammen med hardwaren.

I dette tilfælde blev In Vitro maskinen i Cave Lab brugt til at simulere et blodtrykssignal. Dette blev sendt ind i systemet og sammenlignet med en anden "rigtig" blodtryksmåler, som var tilkoblet samtidigt. In Vitro maskinen danner et tryk, som efterligner et hjerteslag. Trykket bliver skabt i vand, som presses igennem en falsk hjerteklap, derved er der opbygget en model af et hjerte, som kan give et blodtrykssignal.



Figur 3.47: Opstilling til Integrationstest

Der startes med at lave en nulpunktsjustering på systemet, som bagefter viser et korrekt signal, som lå så konstant, at det var svært at se cursoren. Samtidig stod systolisk og diastolisk tryk i tal på GUI. Her viste det sig, at systemet konstant afveg fra den anden blodtryksmåler med en værdi på 3-4, både på systolisk og diastolisk tryk. Da den konstant afveg, vurderes det at afvigelsen skyldes at systemet ikke var kalibreret eller at den anden blodtryksmåler ikke var nulpunktjusteret. Der kunne ikke udføres en kalibrering, da væskesøjlen var gået i stykker. Det blev også forsøgt at nulpunktsjustere den anden blodtryksmåler, men heller ikke dette kunne udføres. Derfor blev resultatet af testen, at systemet virker som det skal, efter de opstillede krav.



Figur 3.48: Målinger under Integrationstest

# Accepttest 4

---

Version	Dato	Ansvarlig	Beskrivelse
0.1	28-09-2015	MHNK og MBA	Oprettelse og udfyldelse af Accepttest
0.2	30-09-2015	ABH	Tilrette accepttest
0.3	08-10-2015	Alle	Tilrette efter review med Grp. 1
0.4	15-10-2015	MBA	Indskrevet i LaTex
0.5	20-10-2015	MHNK	Tilretning
0.6	26-11-2015	MHNK	Retning af hele accepttesten. Konsekvent med stavemåder
0.7	10-12-2015	DHC, ABH, AJF	Rettelser i forhold til slutprodukt
0.8	13-12-2015	MHNK	Færdiggørelse efter accepttest. Indskrivning af godkendelser. Oprettelse og skrivning af problemrapport

## 4.1 Accepttest af Use Cases

## 4.2 Indledning

Accepttesten skal vise om produktet lever op til de standarder, der er blevet sat op for, at den aktivt kan indgå i en forskningssituations. Accepttesten er en opfølging af kravspecifikation, som har til formål at sikre, at alle kravene er overholdt. Der vil blive testet både på hovedscenarier, samt undtagelser og udvidelser. Det er målsætningen, at disse test sikrer produktets kvalitet, idet produktet vil blive afprøvet før det tages i brug. Derfor er det accepttestens ansvarsfunktion, at godkende de opsatte delmål for produktet, hvad angår både funktionelle, samt ikke-funktionelle krav.

Data, der benyttes til målingerne fås fra In Vitro, der i form af tryk simulerer et fysiologisk blodtryk. Brugergrænsefladen er det som forskeren interagerer med, altså hvorfra systemet aktiveres. Den benyttede Database er en lokal database.

Når der i feltet Godkendt er et flueben, betyder det at testen er godkendt. Hvis der er et flueben i parenteser, betyder det at den er delvis godkendt. Hvis der er et kryds betyder det, at den ikke er godkendt.

### 4.2.1 Use Case 1

Det forventes for Use Case 1, at forskeren har fået tilsluttet transduceren og åbnet for tilgang af atmosfærisk tryk.

---

Test af Use Case 1	Foretag nulpunktsjustering
Scenarie	Hovedscenarie
Prækondition	Blodtryksmålesystemet er monteret korrekt. Forskeren har tændt for Blodtryksmåleren og pop-up vindue for nulpunktsjustering er åbent

---

Handling	Forventet observation/resultat	Faktisk observation/resultat	Godkendt
<i>Hovedscenarie</i>			
1. Forsker trykker på Foretag-knap	Systemet foretager nulpunktsjustering, hvorefter vinduet lukker	Som forventet foretager systemet nulpunktsjustering, hvorefter vinduet lukker, når forsker har trykket på Foretag-knap	✓

Tabel 4.3: Accepttest af Use Case 1

### 4.2.2 Use Case 2

---

Test af Use Case 2	Bestem kalibreringskoefficient
Scenarie	Hovedscenarie
Prækondition	Hardware er monteret ved 50 mmHg på væskesøjlen og er tilkoblet en computer med WaveForm

---

Handling	Forventet observation/resultat	Faktisk observation/resultat	Godkendt
<i>Hovedscenarie</i>			
1. Output spænding fra hardware aflæses i WaveForm	Output aflæses til 1 V +/- 30%	I WaveForm aflæses output til 1.04 V	✓

2.	Beregning foretages ud fra formlen $\frac{50}{output} = koefficient$	Koefficenten beregnes til 50 +/- 30%	Som forventet beregnes koefficenten til 48.1	✓
3.	Forsker indtaster beregnet kalibreringskoefficient i konfigurations XML-fil	Koefficenten står i XML-fil	Koefficenten står i XML-fil	✓
4.	Kalibreringskoefficient tilgås af systemet	Sammenlign værdierne i listen råData(findes i IndhentDAQData) med den viste graf på GUI	Værdierne overens	stemmer ✓

Tabel 4.5: Accepttest af Use Case 2

#### 4.2.3 Use Case 3

Test af Use Case 3	Start måling
Scenarie	Hovedscenarie
Prækondition	Blodtryksmålesystemet er monteret korrekt. Forskeren har tændt for Blodtryksmåleren. UC1 er kørt succesfuldt

Handling	Forventet observation/resultat	Faktisk observasjon/resultat	Godkendt
<i>Hovedscenarie</i>			
1. Forsker indtaster Forsøgsnavn	Systemet tilgængeliggør Start Måling-knap	Når Forsker indtaster Forsøgsnavn, gör systemet Start Måling-knap tilgængelig	✓
2. Filtreret signal er valgt per default af systemet	Radiobutton til filtrete signal er checket af	Det ses at Radiobutton til filtrete signal er checket af	✓

3.	Forsker trykker på Start Måling-knap på GUI	Signal vises i graf på GUI	Når forsker trykker på Start Måling-knap, vises signalet i graf på GUI ✓
4.	Systolisk og diastolisk blodtryk, samt puls bliver vist i bokse på GUI	GUI udskriver systoliske, diastoliske og puls værdier på GUI	GUI udskriver systoliske og diastoliske værdier på GUI. GUI udskriver ikke puls <sup>1</sup> (✓)

*Udvidelse 1: Forsker vælger filtreret/ufiltreret signal*

1.	Forsker vælger ufiltreret signal	Grafen viser ufiltreret signal	Forsker vælger ufiltreret signal og grafen viser det ufiltrerede signal ✓
2.	Forsker vælger filtreret signal	Grafen viser filtreret signal	Forsker vælger filtreret signal og grafen viser det filtrerede signal ✓

Tabel 4.7: Accepttest af Use Case 3

#### 4.2.4 Use Case 4

Test af Use Case 4	Gem data
Scenarie	Hovedscenarie
Prækondition	Blodtryksmålesystemet er monteret korrekt. Forskeren har tændt for Blodtryksmåleren. Use Case 1 er kørt succesfuldt, Use Case 3 kører

Handling	Forventet observation/resultat	Faktisk observation/resultat	Observation/Resultat	Godkendt
<i>Hovedscenarie</i>				

<sup>1</sup>Se problemrapport

1.	Forsker trykker på Start Gem-knap	Start Gem-knap bliver highlightet med blå kant	Når Forsker trykker på Start Gem-knap bliver den highlightet med blå kant	✓
2.	Forsker trykker på Stop Gem-knap for, at stoppe med at gemme	Filnavnet (Forsøgsnavn_Id) bliver vist i tekstboks GUI	Når Forsker trykker på Stop Gem-knap for at gemme bliver filnavnet (Forsøgsnavn_Id) vist i tekstboks i GUI	✓
3.	Forsker trykker på Gem-knap for at stoppe med at gemme	Det fremgår af GUI at rådata er gemt i Database	Når Forsker trykker på Gem-knap for at stoppe med at gemme, fremgår det af GUI at rådata er blevet gemt i Database	✓

*Undtagelse 1: Forsker trykker på Stop Måling-knap*

1.	Forsker trykker på Stop Måling-knap til et givent tidspunkt	Grafen på GUI fastholdes og datostempel på seneste indlagte rådata aflæses i Databasens tabel. Tiderne sammenlignes	Når Forsker trykker på Stop Måling-knap til et givet tidspunkt, fastholdes grafen på GUI og datostempel på seneste indlagte rådata aflæses i Databasens tabel, hvor efter tiderne sammenlignes	✓
----	---	---	--	---

*Tabel 4.9: Accepttest af Use Case 4*

### 4.2.5 Use Case 5

---

Test af Use Case 4	Stop måling
Scenarie	Hovedscenarie
Prækondition	Blodtryksmålesystemet er monteret korrekt. Forskeren har tændt for Blodtryksmåleren. Use Case 1 er kørt succesfuldt, Use Case 3 kører

---

Handling	Forventet observation/resultat	Faktisk observasjon/resultat	Godkendt
<i>Hovedscenarie</i>			
1. Forsker trykker på Stop Måling-knap	Målingen stoppes og blodtryksgrafen fastholdes	Forsker trykker på Stop Måling-knap og målingen stoppes og blodtryksgrafen fastholdes	✓

Tabel 4.11: Accepttest af Use Case 5

### 4.3 Accepttest af ikke-funktionelle krav

Krav nr.	Krav	Test	Forventet resultat	Resultat	Godkendt
1.	Blodtryksmåleren skal indeholde en Start Måling-knap til at igangsætte målingerne	Kør Use Case 1 og 3	Start knap er på GUI	Måling-knap er på GUI	Use Case 1 og 3 køres og der er en Start Måling-knap på GUI ✓
2.	Blodtryksmåleren skal indeholde en Stop Måling-knap, hvorfra måling kan stoppes	Kør Use Case 1 og 3	Stop knap er på GUI	Måling-knap er på GUI	Use Case 1 og 3 køres og der er en Stop Måling-knap på GUI ✓
3.	Blodtryksmåleren skal indeholde en Start Gem-knap til påbegyndelses af at gemme måling i Database	Kør Use Case 1 og 3	Start Gem-knap er på GUI	Gem-knap er på GUI	Use Case 1 og 3 køres og der er en Start Gem-knap på GUI ✓
4.	Blodtryksmåleren skal indeholde en Stop Gem-knap til påbegyndelses af at gemme måling i Database	Kør Use Case 1 og 3	Stop Gem-knap er på GUI	Gem-knap er på GUI	Use Case 1 og 3 køres og der er en Stop Gem-knap på GUI ✓

5.	Blodtryksmåleren skal indeholde en tekstboks til forsøgsnavn, hvori Forsker indtaster det pågældende Forsøgsnavn	Kør Use Case 1 og 3	Tekstboks til Forsøgsnavn er på GUI	Use Case 1 og 3 køres og der er en tekstboks til Forsøgsnavn, hvori Forsker indtaster det pågældende Forsøgsnavn	✓
6.	Blodtryksmåleren skal indeholde radiobutton til filtreret signal, denne skal være default valget	Kør Use Case 1 og 3	Radiobutton til filtreret signal er på GUI	Use Case 1 og 3 køres og der er en radiobutton til filtreret signal, der er valgt per default	✓
7.	Blodtryksmåleren skal indeholde radiobutton til ufiltreret signal	Kør Use Case 1 og 3	Radiobutton til ufiltreret signal er på GUI	Use Case 1 og 3 køres og der er en radiobutton til ufiltreret signal	✓
8.	Blodtryksmåleren skal indeholde teknikbokse til puls, systolisk og diastolisk blodtryk, som vises med op til tre cifre	Kør Use Case 1 og 3	Systolisk-boks, diastolisk-boks og puls-boks er på GUI	Use Case 1 og 3 køres og der er teknikbokse, der indeholder puls, systolisk og diastolisk blodtryk, som vises med op til tre cifre	✓
9.	Blodtryksmåleren skal indeholde en tekstuarkasse, som viser filnavn (Forsøgsnavn og Id) på målingen, efter målingen er gemt	Kør Use Case 1 og 3	Tekstboks til filnavn er på GUI	Use Case 1 og 3 køres og der er en tekstuarkasse, der viser filnavn (Forsøgsnavn og Id) på målingen, efter måling er gemt	✓

10.	GUI'en skal se ud som på figur 2.4 i KS	GUI'en ser ud som figur 2.4 i KS	GUI'en ser ud som figur 2.4 i KS	GUI'en ser ud som figur 2.4 i KS	✓
11.	Forskeren skal kunne starte en default-måling maksimalt 30 sekunder efter systemet er startet	Systemet er åbnet og samtidig startes et stopur. Efter tryk på Start Måling-knap og målingen er startet stoppes uret	Måling er startet og stopuret viser mindre end 30 sekunder	Måling viste 3,73 sekunder	✓
12.	Det skal maksimalt tage 5 timer at gendanne systemet (MTTR - Mean Time To Restore)	Kan ikke testes på prototypen	Kan ikke testes på prototypen	Kan ikke testes på prototypen	✓
13.	Systemet skal have en oppetid uden nedbrud på minimum 1 måned (720 timer) (MTBF - Mean Time Between Failure)	Kan ikke testes på prototypen	Kan ikke testes på prototypen	Kan ikke testes på prototypen	✓
14.	Systemet skal have en oppetid/køretid på: $\frac{MTBF}{MTBF + MTTR}^*$ $100 = 99,31\%$	Kan ikke testes på prototypen	Kan ikke testes på prototypen	Kan ikke testes på prototypen	✓

15.	Blodtryksmåleren skal, indenfor 3 sekunder, kunne vise systolisk og diastolisk blodtryk via grafen. Dette accepteres med en tolerance på +/- 15 %	Kør Use Case 1 og 3. Der trykkes på Start Måling-knappen og samtidig startes et stopur. Når måling vises i graf stoppes uret	Stopuret viser mellem 2.55 - 3.45 sekunder	Stopuret viste sekunder for systolisk. Den diastoliske værdi blev vist efter 6 sekunder	(✓)
16.	Blodtryksmåleren skal, indenfor 5 sekunder fra der er trykket på Stop Gem-knap, have gemt målingerne i Databasen. Dette accepteres med en tolerance på +/- 15 %	Kan ikke testes på prototypen	Kan ikke testes på prototypen	Kan ikke testes på prototypen	✓
17.	Grafen vises i ét vindue, hvor y-aksen måles i mmHg og x-aksen i tid pr. sekund	Kør Use Case 1 og 3	På GUI er y-aksen målt i mmHg og x-aksen i tid pr. sekund	Use case 1 og 3 køres og grafen vises i ét vindue, hvor y-aksen måles i mmHg. x-aksen vises ikke i sekunder, men i antal samples <sup>2</sup>	X

<sup>2</sup>Se problemrapport

18.	Hvert 3. sekund skal værdier for systolisk og diastolisk blodtryk, samt puls opdateres. Dette accepteres med en tolerance på +/- 15 %	Kør Use Case 1 og 3. Forsøgsnummer indtastes og der trykkes på Start Måling-knappen samtidig med at et stopur startes. Når værdier i bokse vises stoppes uret	Stopuret viser mellem 2.95 - 3.02 sekunder 3.15 sekunder	✓
19.	Grafen for blodtryk skal køre kontinuerligt i GUI efter princippet på figur 2.5	Kør Use Case 1 og 3	Grafen i GUI kører kontinuerligt efter principippet på figur 2.5	Use case 1 og 3 køres og grafen i GUI kører kontinuerligt efter principippet på figur 2.5
20.	Når der trykkes på Stop Gem-knap gemmes signalets rådata under det indtastede Forsøgsnavn og et autogenereret Id. " <i>Forsøgsnavn_Id</i> "	Kør Use case 1, 3 og 4	Rådata er blevet gemt i Databasen under filnavnet " <i>Forsøgsnavn_Id</i> "	Use case 1, 3 og 4 køres og når der trykkes på Stop Gem-knap gemmes signalets rådata under det indtastede Forsøgsnavn og et autogenereret Id
21.	Systemet skal kunne måle blodtryksværdier fra 0 til 250 mmHg	Kør Use Case 1 og 3	Det indhentede signals blodtryksværdier er indenfor 0 til 250 mmHg på grafens y-akse	Der er ingen begrænsning <sup>3</sup> (✓)

<sup>3</sup>Se problemrapport

22.	Forskeren skal kunne udskifte batterierne til hardwaren på 2 minutter.	Udskiftning af batterier påbegyndes samtidig med at stopur startes. Når de er udskiftet stoppes uret	Stopuret viser mindre end 2 minutter	Forsker kan hurtigt skifte dette. Stopuret visste 15 sekunder	✓
23.	Softwaren skal opbygges med lav kobling	Åbn systemets programkode	Koden er opbygget med lav kobling	Koden er opbygget med lav kobling	✓

Tabel 4.12: Accepttest af Ikke-funktionelle krav

## 4.4 Godkendelsesformular

Godkendes af Peter Johansen

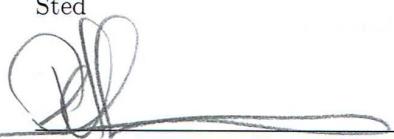
Kunde IHA

Dato for test 11/12 2015

Ved underskrivelse af dette dokument godkendes den kørte accepttest.

Aarhns

Sted



Kundens underskrift

11/12 2015

Dato

Mette Hansen H-H

Leverandørens underskrift

## 4.5 Problemrapport

Use Case 3, handling 4: Systolisk og diastolisk blodtryk, samt puls bliver vist i bokse på GUI. Alle boksene er oprettet og vises i GUI, men kun systolisk og diastolisk bokse blive udfyldt. Puls er ikke kommet til at fungere.

Ikke-funktionelle krav, handling 17: Grafen vises i ét vindue, hvor y-aksen måles i mmHg og x-aksen i tid pr. sekund.

Y-aksen vises i mmHg, men x-aksen viser tid i antal samples. Dog passer det overens med at grafen viser 300 samples, så det vides at grafen viser 6 sekunder.

Ikke-funktionelle krav, handling 21: Systemet skal kunne måle blodtryksværdier fra 0 til 250 mmHg.

Dette kan systemet også, det er blot blevet valgt ikke at sætte nogle begrænsninger på, så y-aksen med mmHg stopper ikke ved de 250 mmHg. Hvis dette ønskes kan det uden problemer tilføjes i koden, ved at låse grafområdets y-akse til at gå fra 0 til 250 mmHg.

# Litteratur

---

- [1] Gregory J. Toussaint Rolande E. Thomas, Albert J. Rose. *The Analysis and Design og Linear Circuits*. Wiley, 2012.
- [2] Richard G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, 2011.
- [3] Peter Johansen. *Instrumentationsforstærkeren*. IHA, 2014.
- [4] *INA114 Datasheet*.
- [5] [https://en.wikipedia.org/wiki/Sallenkl\\_13.16](https://en.wikipedia.org/wiki/Sallenkl_13.16).
- [6] Gregory J. Toussaint Rolande E. Thomas, Albert J. Rose. *The Analysis and Design og Linear Circuits*. Wiley, 2012.
- [7] Lene Hauser. *3-lags model*. IHA, 2015.
- [8] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.
- [9] *Testen af URL i referencer*. Online, accessed 15.12.
- [10] Kurt Nørmark. Objekt-orienteret programmering introduktion, integrationstest. <http://people.cs.aau.dk/~normark/prog1-01/html/noter/test-dokumentation-note-integrationstest.html>, 2001. Online, accessed 14.12.

# Figurer

---

2.1	Use Case-diagram	3
2.2	Aktør-kontekstdiagram	4
2.3	Use Case-diagram	5
2.4	Skitse af GUI	10
2.5	Graf for blodtryk	11
3.1	Block Definition Diagram for hardware	13
3.2	Internal Block Diagram for hardware	13
3.3	Bodeplot	17
3.4	Diagram over HW	18
3.5	Forstærknings blok	18
3.6	Måling for 10 Hz	19
3.7	Måling for 50 Hz	20
3.8	Måling for 60 Hz	20
3.9	Graf til kalibrering, fra udregninger	21
3.10	Opstilling	21
3.11	Måling ved 50 mmHg	22
3.12	Måling ved 10mmHg	22
3.13	Måling ved 100mmHg	23
3.14	Overordnet sekvensdiagram for systemet	24
3.15	Domænemodel	25
3.16	Applikationsmodel for software	25
3.17	Sekvensdiagram for Use Case 1	27
3.18	Sekvensdiagram for Use Case 2	27
3.19	Sekvensdiagram for Use Case 3	28
3.20	Sekvensdiagram for Use Case 4	29
3.21	Sekvensdiagram for Use Case 5	29
3.22	Klassediagram	31
3.23	NulpunktsjusteringGUI og HovedGUI	31
3.24	Observer mønstre	33
3.25	Konfigurations XML-fil	35
3.26	Aktivitetsdiagram af metoden Filtrering()	36
3.27	Udsnit af koden til det glidende middelværdifilter	36
3.28	Aktivitetsdiagram for metoden updateListe()	38
3.29	SQL-kode til oprettelse af tabeller i database	39
3.30	Kode for eventet StartKnapClick()	40
3.31	Kode for Logik metoden indhentDataLogik()	40
3.32	Kode for IndhentDAQData metoden indhentData()	40
3.33	Kode for InitializeDataTable()	40
3.34	Kode for IndhentDAQData metoden AnalogInCallBack()	40

3.35 Kode for IndhentDAQData metoden DataToDataTable() . . . . .	41
3.36 Kode for metoden Attach(), denne bruges på samme måde i klasserne IndhentDAQ- Data som i Logik . . . . .	41
3.37 Kode for metoden Notify() i IndhentDAQData klassen . . . . .	41
3.38 Kode for metoden Gennemsnit() i Logik-klassen . . . . .	41
3.39 Kode for Logik metoden StartTraad() . . . . .	42
3.40 Kode for tråden updateUI . . . . .	42
3.41 Kode for logik metoden updateListe() . . . . .	42
3.42 Kode for metoden FiltreringLogik() . . . . .	43
3.43 Kode for klassen Filter(). Her bliver summen af 5 værdier divideret med 5 . . . . .	43
3.44 Kode for metoden Notify() i Logik-klassen . . . . .	43
3.45 Kode for metoden i GUI Laget Gennemsnit() . . . . .	44
3.46 Kode for metoden i GUI Laget updateChart() . . . . .	44
3.47 Opstilling til Integrationstest . . . . .	45
3.48 Målinger under Integrationstest . . . . .	45

# Bilag

---

Herunder findes en liste over bilagene.

1. Tidsplan
2. Samarbejdsaftale
3. Development Process
4. NI-6009 DAQ Datasheet
5. INA114 Datasheet
6. Instrumentationsforstærkeren
7. Transducer Datasheet
8. OP27 Datasheet
9. Beregninger af overføringsfunktion
10. HW-diagram
11. HW komponentliste
12. Trelagsmodel
13. Klassediagram
14. Softwarekode pdf
15. Softwarekode Visual Studio
16. NI-DAQ kode
17. Mødeindkaldelser
18. Logbog
19. Mødreferater