



## 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 16/12-2015

Anne Bundgaard Hoelgaard (201404492) Dato

Mette Hammer Nielsen-Kudsk 16/12-2015

Mette Hammer Nielsen-Kudsk (201408391) Dato

Ditte Heebøll Callesen 16/12-2015

Ditte Heebøll Callesen (201408392) Dato

Martin Banasik 16/12-15

Martin Banasik (201408398) Dato

Albert Jakob Fredshavn 16/12-2015

Albert Jakob Fredshavn (201408425) Dato

Johan Mathias Munk 16/12-15

Johan Mathias Munk (201408450) Dato

## Vejleder

Peter Johansen 16/12/15

Peter Johansen Dato

# Ordliste

---

Ord	Forklaring
(F)URPS+	Et akronym, der repræsenterer en model til klassificering af softwarens 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
KSS	Kommunikation og Samarbejde i Sundhedsvæsnet
BDD	Block Definition Diagram
IBD	Intern Block Diagram
SD	Sekvensdiagram
UML	Unified Modeling Language
ISE	Indledende System Engineering
KVI	Kardiovaskulær Instrumentering
Hjerteinsufficiens	Hjertesvigt
DSB	Digital Signalbehandling
Hypertension	Forhøjet blodtryk
Hypotension	Lavt blodtryk
MTTR	Mean Time To Restore
MTBF	Mean Time Between Failure
Anatomi	En organismes indre og ydre opbygning
Veroboard	Printplade med isoleret kobberører
CMRR	Common Mode Rejection Ratio
In Vitro	Maskine til simulering af tryk, i dette tilfælde et blodtryk



# 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 . . . . .	15
3.1.3 Modultest . . . . .	18
3.2 Software . . . . .	23
3.2.1 Design . . . . .	23
3.2.2 Implementering . . . . .	29
3.2.3 Modultest . . . . .	39
3.2.4 Integrationstest . . . . .	46
<b>Kapitel 4 Accepttest</b>	<b>48</b>
4.1 Accepttest af Use Cases . . . . .	48
4.2 Indledning . . . . .	48
4.2.1 Use Case 1 . . . . .	49
4.2.2 Use Case 2 . . . . .	49
4.2.3 Use Case 3 . . . . .	50
4.2.4 Use Case 4 . . . . .	51
4.2.5 Use Case 5 . . . . .	53
4.3 Accepttest af ikke-funktionelle krav . . . . .	54
4.4 Godkendelsesformular . . . . .	61
4.5 Problemrapport . . . . .	62
<b>Figurer</b>	<b>63</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

## 2.2 Godkendelsesformular

Antal sider: 63

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

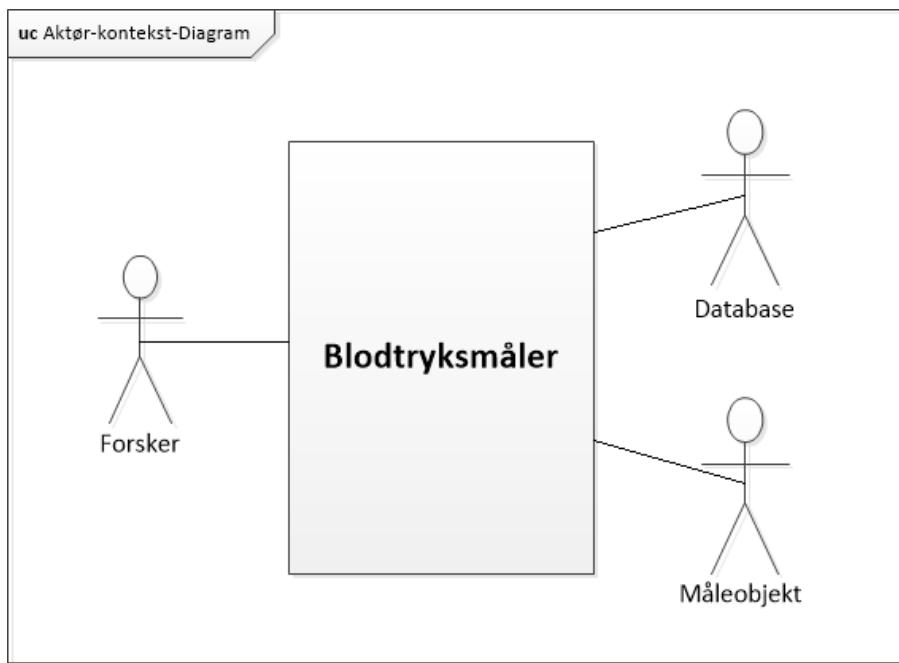
På baggrund af krav fra kunden, samt hvad leverandøren finder muligt, er denne kravspecifikation blevet udarbejdet. Kravspecifikationen har til formål at specificere kravene til produktet. Dette projekt tager udgangspunkt i en blodtryksmåler, hvortil der er en række aktører, som interagerer med systemet.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

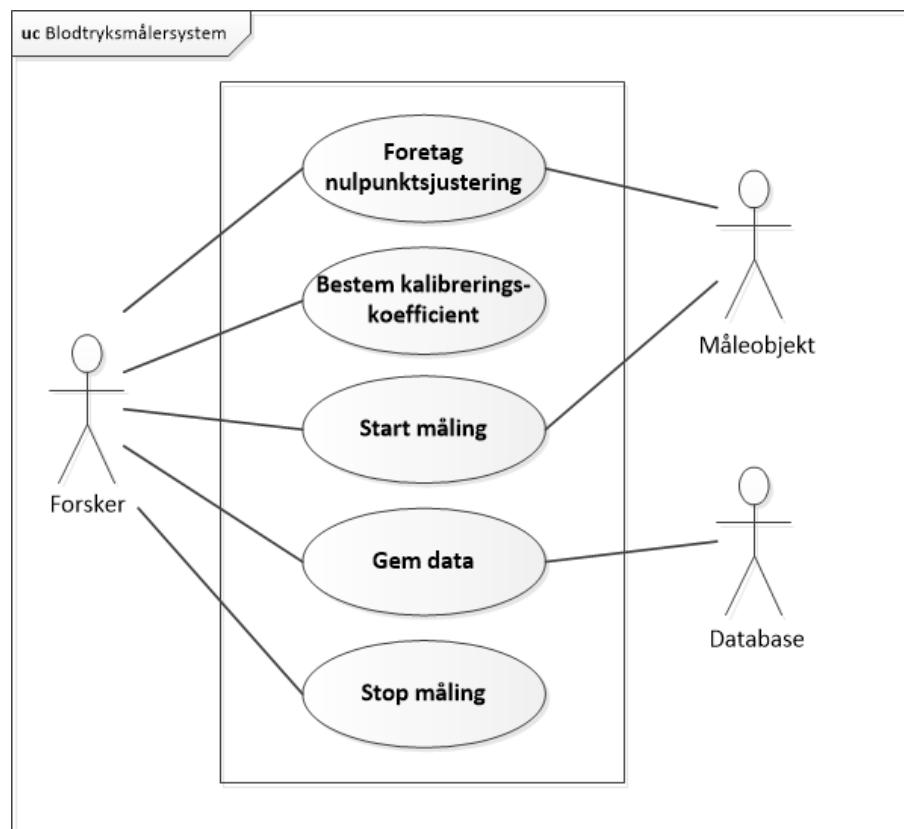
På figur 2.2 ses aktørerne til at være: Forsker, Måleobjekt og Database. Herunder er der en detaljeret beskrivelse af hver aktør.

### 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>Pop-up vindue for nulpunktsjustering er åben</li> <li>Forsker trykker på Foretag-knap:</li> <li>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 væskesøjle 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	Ét 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> <p>[Udvidelse 1:] Forsker vælger filtreret/ufiltreret signal</p>
---------------	--

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> <li>b. Det viste signal er nu ufiltreret</li> <li>c. Forsker vælger filtreret signal</li> <li>d. Det viste signal er nu filtreret</li> </ol>
---------------------------	---

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	Et 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</li> </ol>
---------------	---

	[ <i>Undtagelse 1:</i> ] Forsker trykker på Stop Måling-knap
	4. Det vises at rådata er gemt ved at filnavnet (Forsøgsnavn og Id) for målingen vises på GUI
Undtagelser	[ <i>Undtagelse 1:</i> ] Forsker trykker på Stop Måling-knap
	a. Systemet gemmer ikke målingen og blodtryksgrafen fastholdes

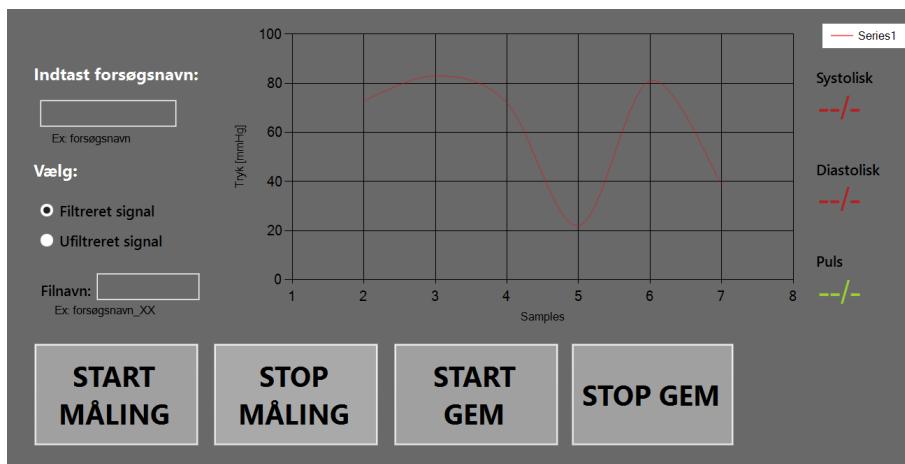
*Tabel 2.6: Fully dressed Use Case 4***Use Case 5**

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 filtreter 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

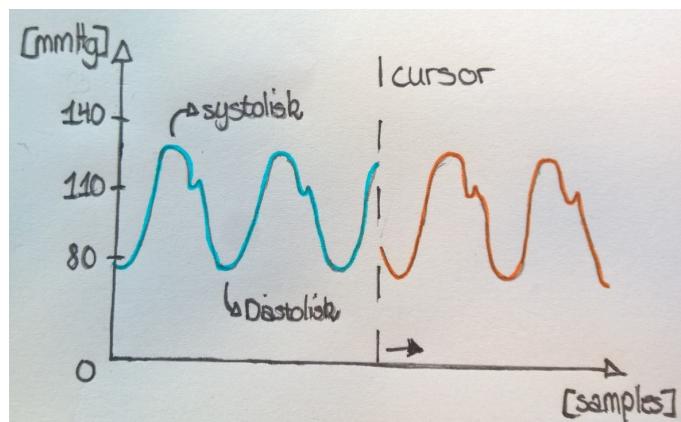
1. Det skal maksimalt tage 5 timer at gendanne systemet (MTTR - Mean Time To Restore)
2. Systemet skal have en oppeid uden nedbrud på minimum 1 måned (720 timer) (MTBF - Mean Time Between Failure)

3. Systemet skal have en oppetid/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 kviksølv) 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 måle 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 vandsøjle
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

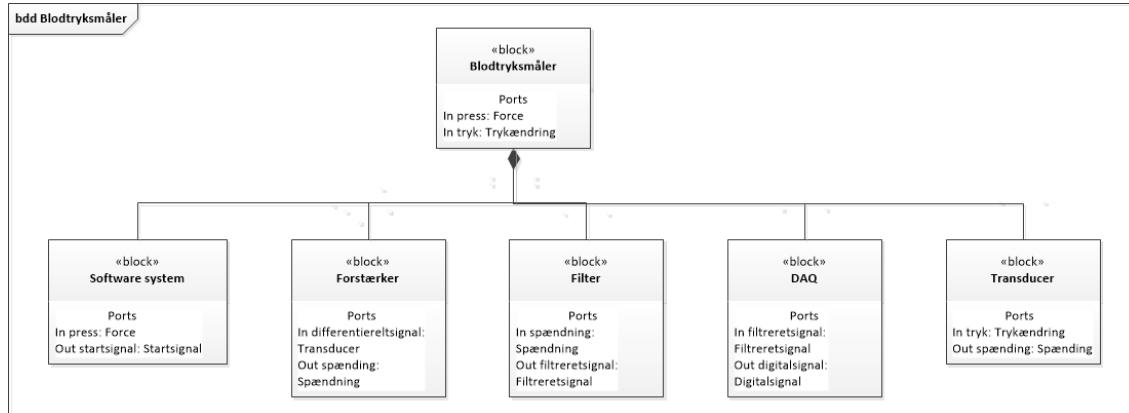
I det følgende beskrives arkitekturen for systemet. Systemarkitekturen er udviklingsramme for den videreudvikling af design og implementering af blodtrykssystemet. Designet af systemet er grebet an således at, der først kigges på det overordnede system, hvorefter systemet arbejdes ned i mindre brudstykker.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

tilhørende beskrivelser.

### 3.1 Hardware

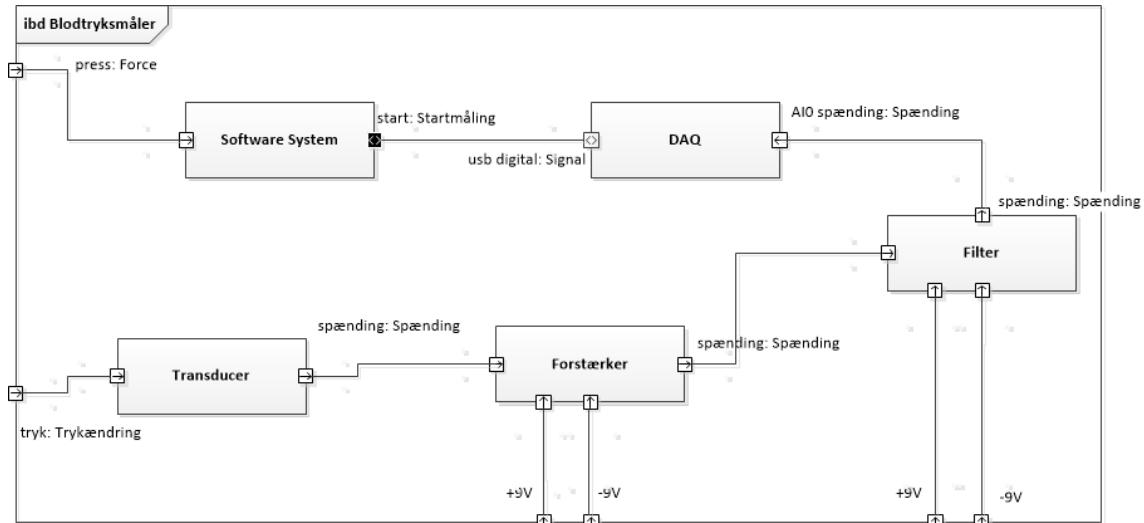
### 3.1.1 Design

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.



*Figure 3.2: Internal Block Diagram for hardware*

## Forstærkning

For at kunne måle fysiske parametre er der benyttet en strain gauge[?], 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[? ] 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 [? ] 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[?] 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 eksistationsspændingen, 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 [?] er 5V, kan forstærkningen (Gain) nu udregnes:

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

[?] 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[?].

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

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 spændingsdelen ikke.

### Lavpas

For at opnå den ønskede effekt i lavpasfilteret, blev det oplyst at  $f_c = 50$  Hz,  $f_s = 1kHz$ ,  $R_1 = R_2$  og  $C_2 = 680nF$ . 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 [?]:

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

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 "Beregninger til overføringsfunktion" under Bilag 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 at 2. ordens Butterworth filter, vides det at udsvinget  $\zeta$  skal være 0.7 [? ]. Den sidste overføringsfunktion sammenlignes med den generelle for 2. ordens systemer. Det gælder at  $C2 = 680 \cdot 10^{-9} nF$ . Det er muligt at isolere forskellige led. Først isoleres der for modstanden:

$$\begin{aligned} \frac{2}{R \cdot C2} &= 2 \cdot \zeta \cdot \omega_n \\ \frac{2}{R \cdot 680 \cdot 10^{-9}} &= 2 \cdot 0.7 \cdot (2 \cdot \pi \cdot 50) \\ &\Downarrow \\ R &= 6687\Omega \end{aligned} \quad (3.10)$$

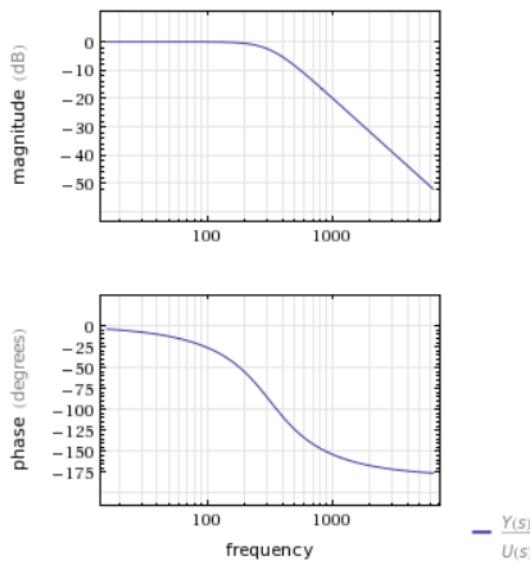
Derved er modstandene udregnet til  $R = 6687\Omega$ . Nu kan der isoleres for kondensator C1:

$$\begin{aligned} \frac{1}{C1 \cdot C2 \cdot R^2} &= \omega^2 \\ \frac{1}{C1 \cdot 680 \cdot 10^{-9} \cdot 6687^2} &= (2 \cdot \pi \cdot 50)^2 \\ &\Downarrow \\ C1 &= 333 \cdot 10^{-9} nF \end{aligned} \quad (3.11)$$

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:

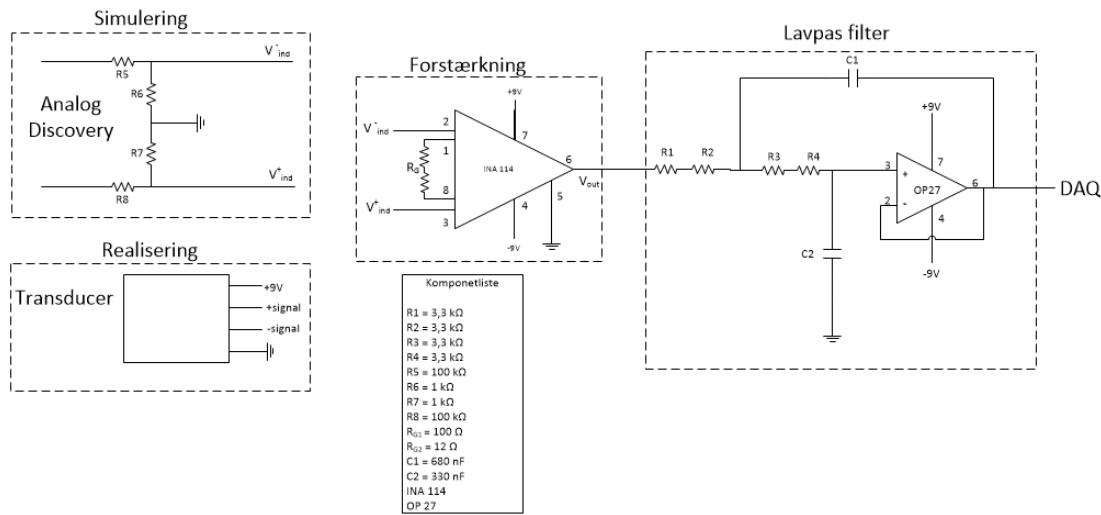
$$\frac{2}{R \cdot C1} = 2 \cdot \zeta \cdot \omega_n$$

$$\frac{2}{6600 \cdot 680 \cdot 10^{-9}} = 2 \cdot \zeta \cdot (2 \cdot \pi \cdot 50) \quad (3.13)$$

$$\downarrow$$

$$\zeta = 0.709$$

Dvs. de små ændringer i komponentværdierne har ikke haft betydnende 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.

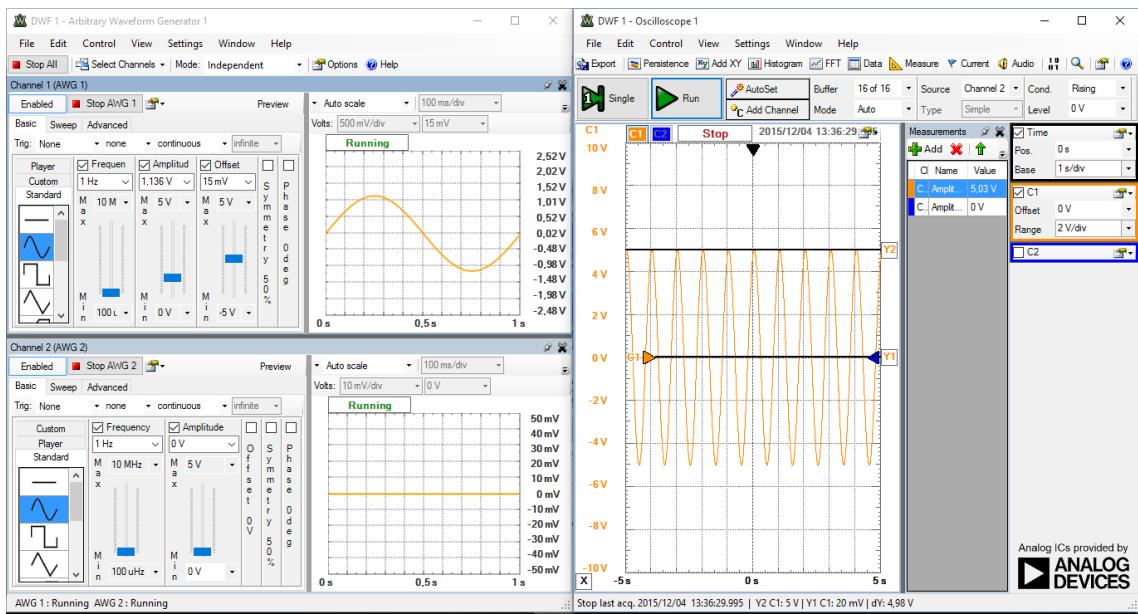
### 3.1.3 Modultest

#### 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.

### 3.1. Hardware

ASE

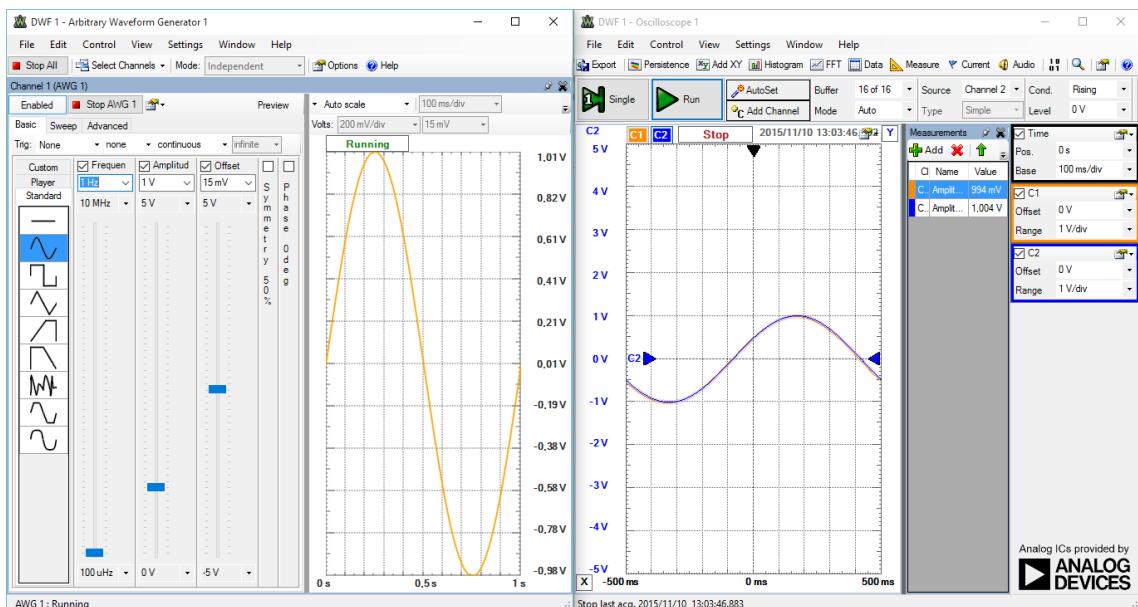


Figur 3.5: Forstærknings blok

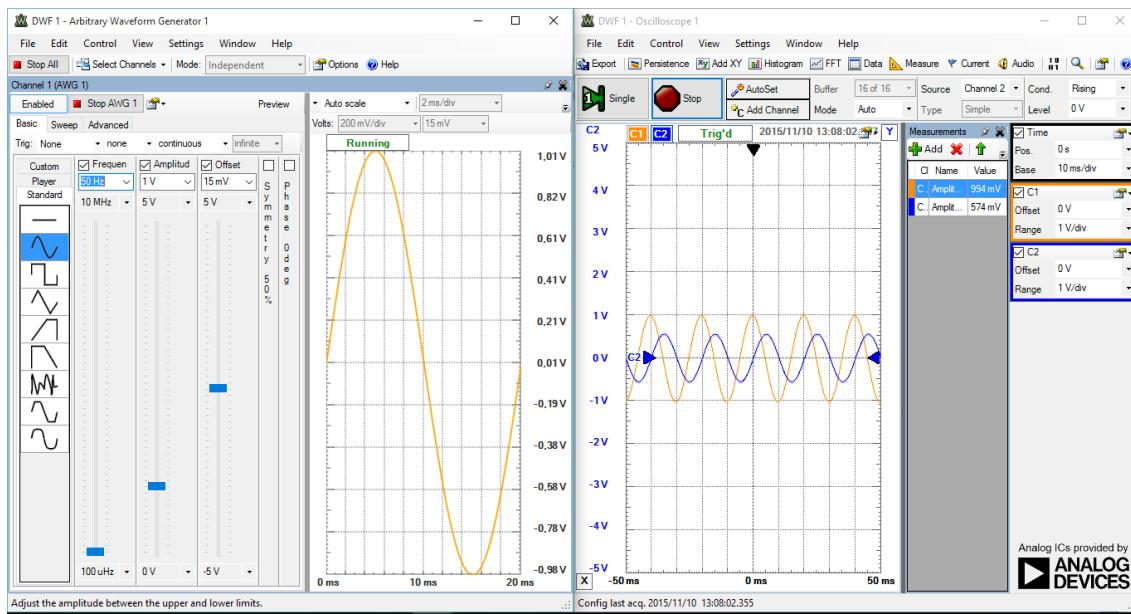
På udgangen ses det, at signalet er blevet forstærket op til 5 V DC. Herved er det maksimale output fra transduceren blevet forstærket så det passer med det maksimale input til DAQ'en. Signalet bliver ikke ændret på andre måder i Forstærker-blokken.

### Lavpas

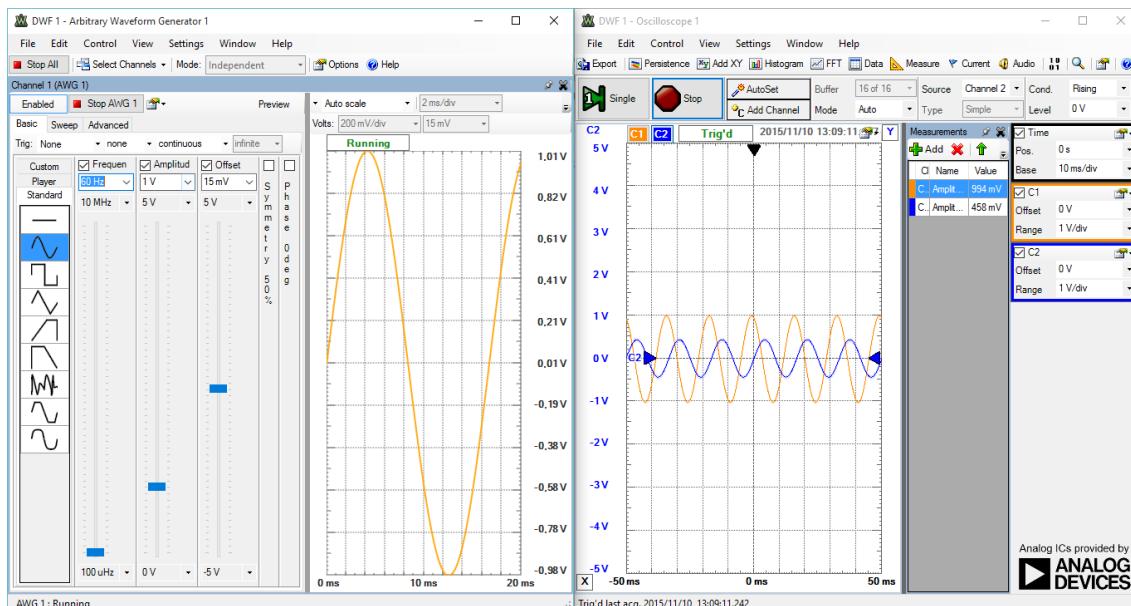
For at teste lavpasfilteret foretages målinger med en sinus, hvor frekvensen varierer for hver måling. Fasen aflæses mellem indgang- og udgangssignalet. Amplituden aflæses ligeledes for hver måling. Ved knækfrekvensen skal phasedrejningen være  $90^\circ$ . Dette kan aflæses på figur 3.7. Efter knækfrekvensen 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



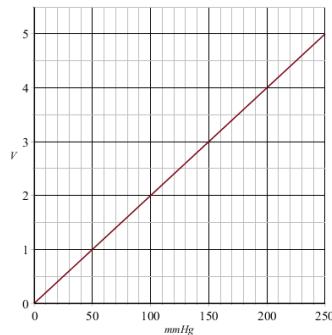
Figur 3.7: Måling for 50 Hz



Figur 3.8: Måling for 60 Hz

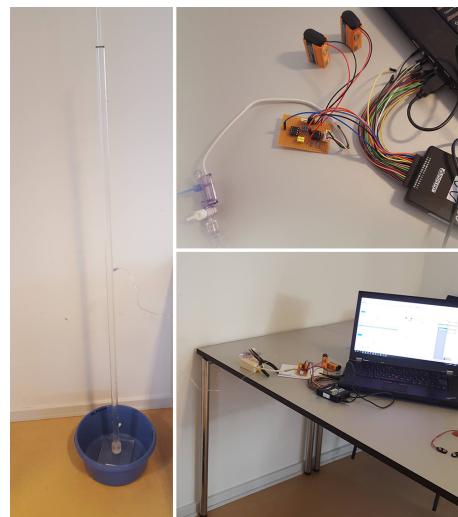
### Kalibrering med væskesøjle

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 kviksølv(mmHg). Ud fra den maksimale spænding (V) og millimeter kviksølv(mmHg) kan det udregnes, hvad hardwaren skal vise ved 100 mmHg.



Figur 3.9: Graf til kalibrering, fra udregninger

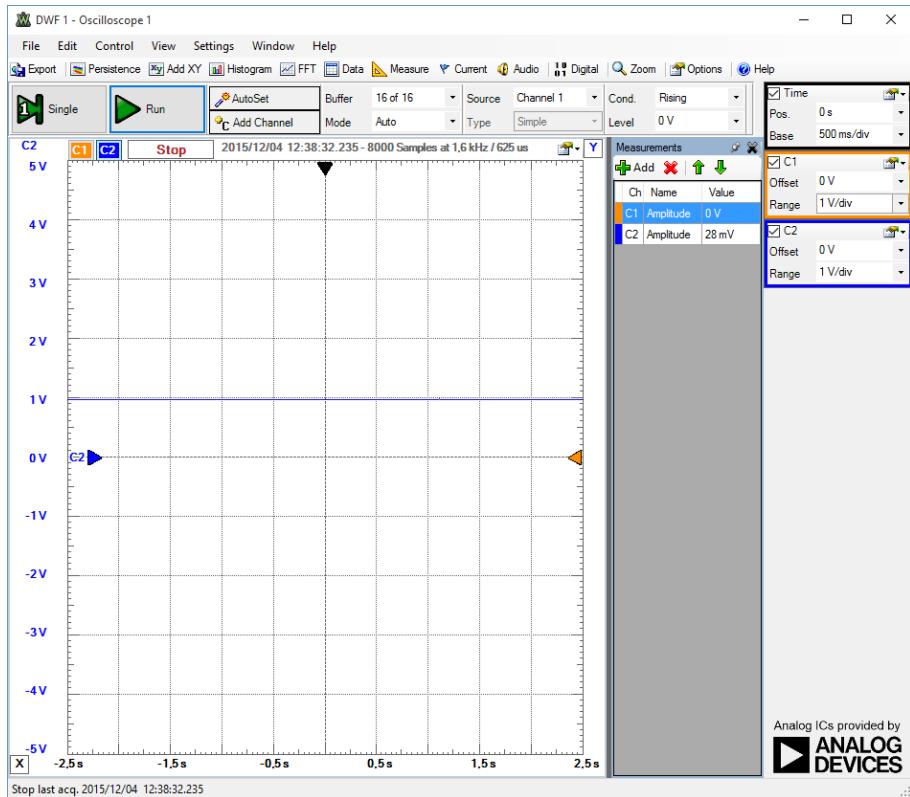
Testen udføres ved, at fyldе vand i søjlen til markeringen. Transduceren skal være tilkoblet et af de tre målepunkter, mens de andre er lukket til. Transduceren er sat til forstærkningen, hvor Analog Discovery tidligere har været sat til. Transduceren er tilkoblet 9 V ved batterierne. På samme måde som 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.



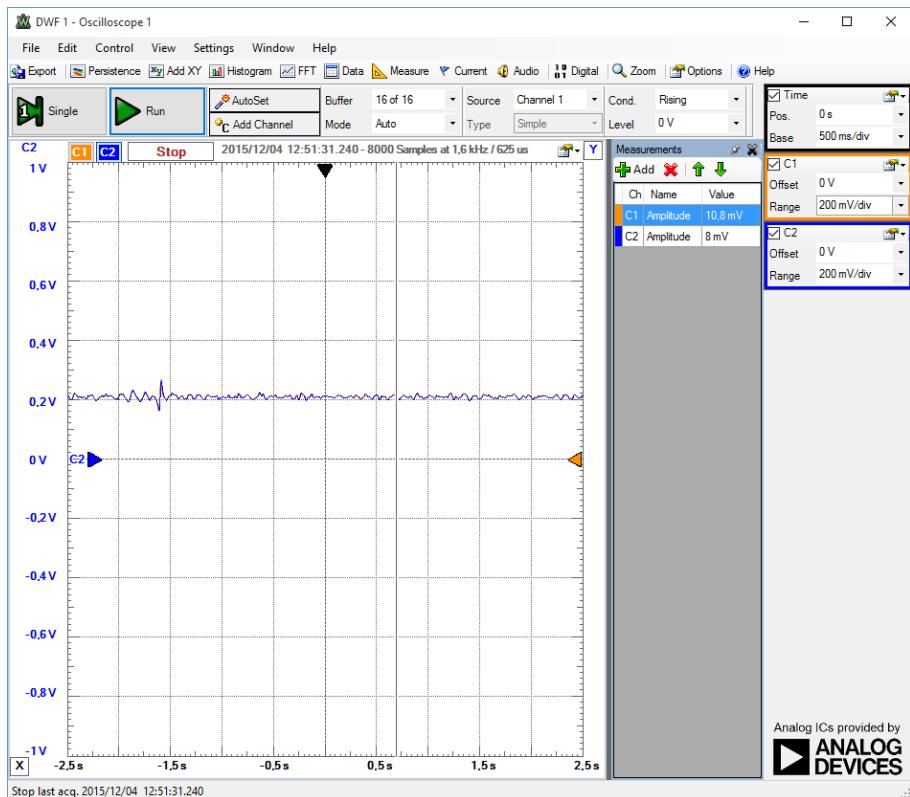
Figur 3.10: Opstilling

Opstillingen er gjort klar og der hentes ekstra vand under testen. Vandet skal bruges til at fyldе 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.

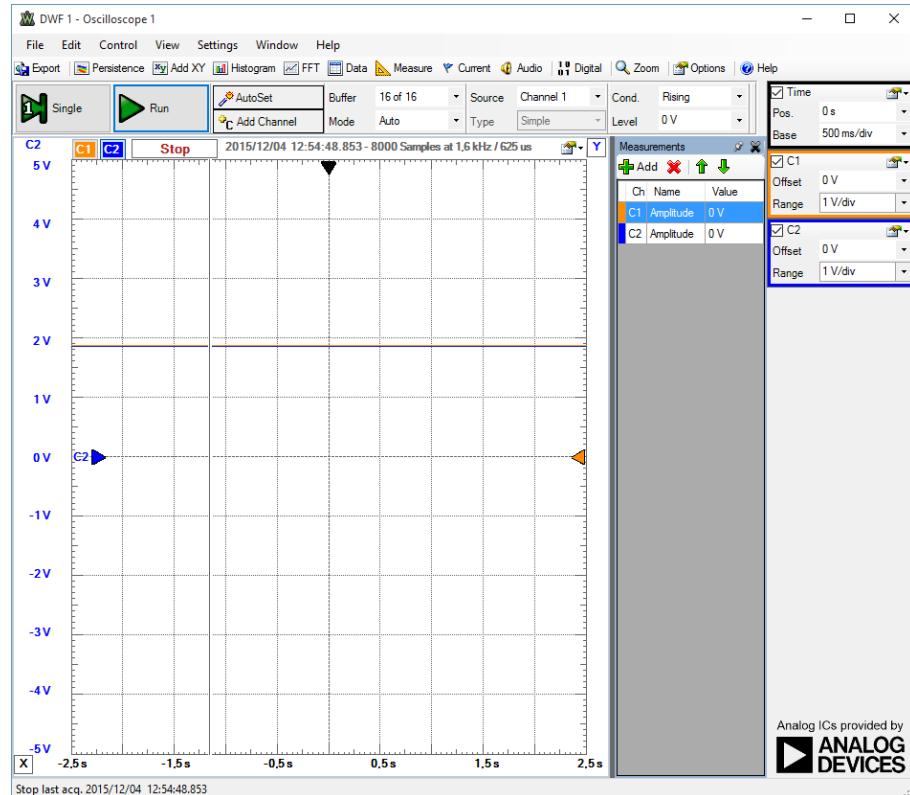


Figur 3.11: Måling ved 50 mmHg



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 præcis 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

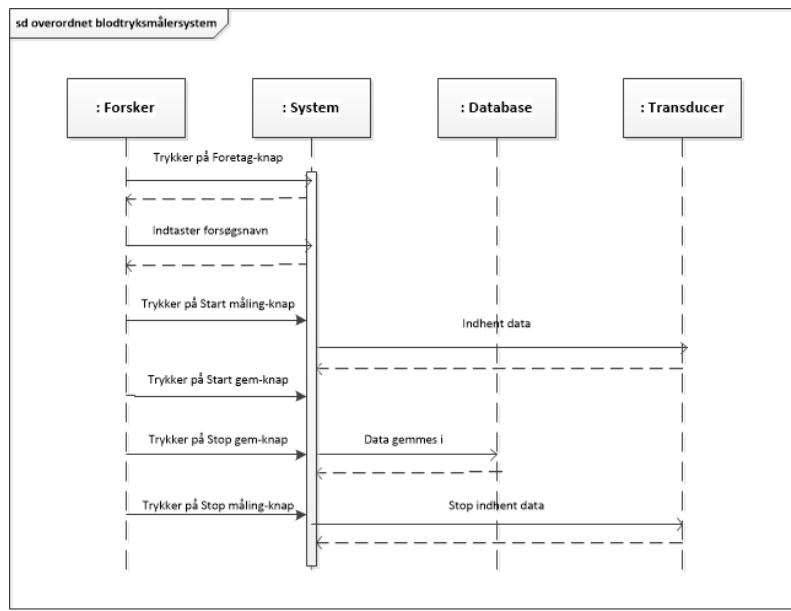
### 3.2.1 Design

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

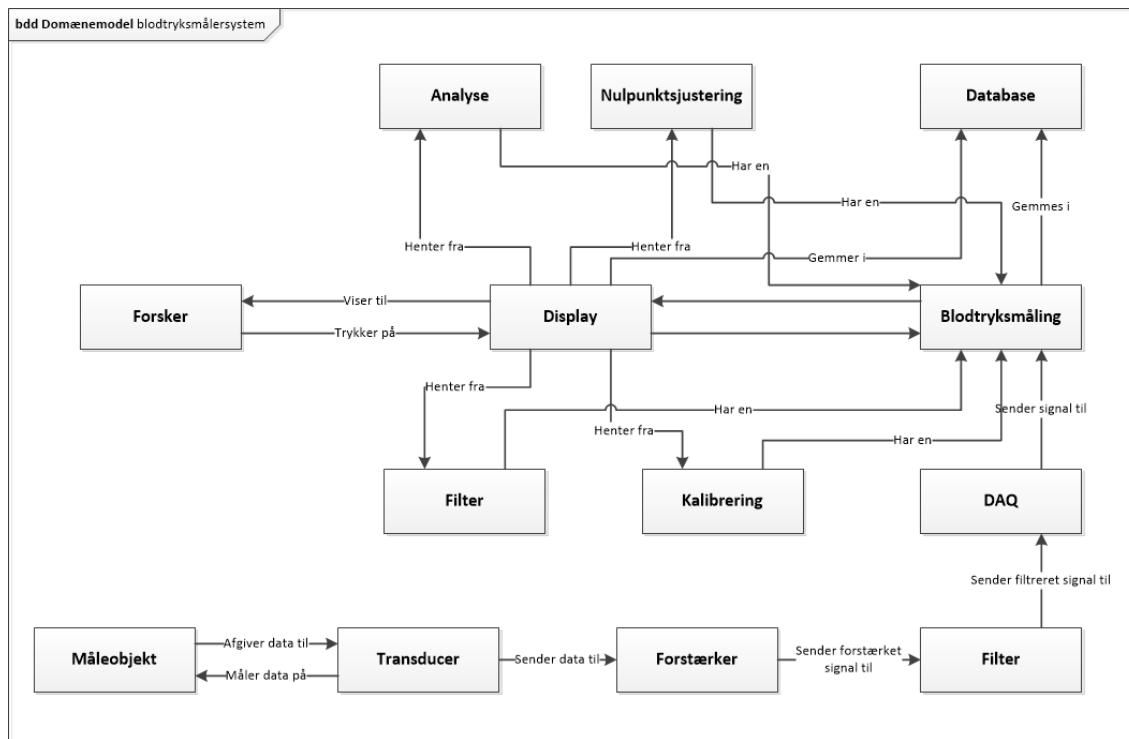
en kalibreringskoefficient, hvis det ønskes. Diagrammet er en simpel illustration, som viser systemets adfærd gennem alle fem Use Cases. Formålet med dette diagram er at skabe et overblik over det samlede system.



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.

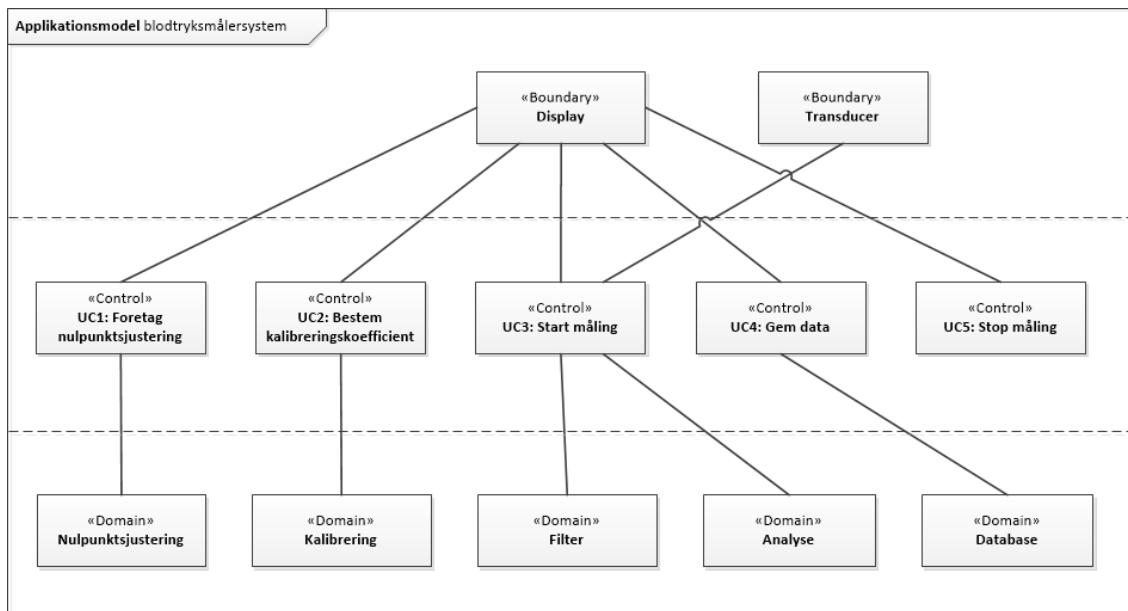


Figur 3.15: Domænemodel

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:

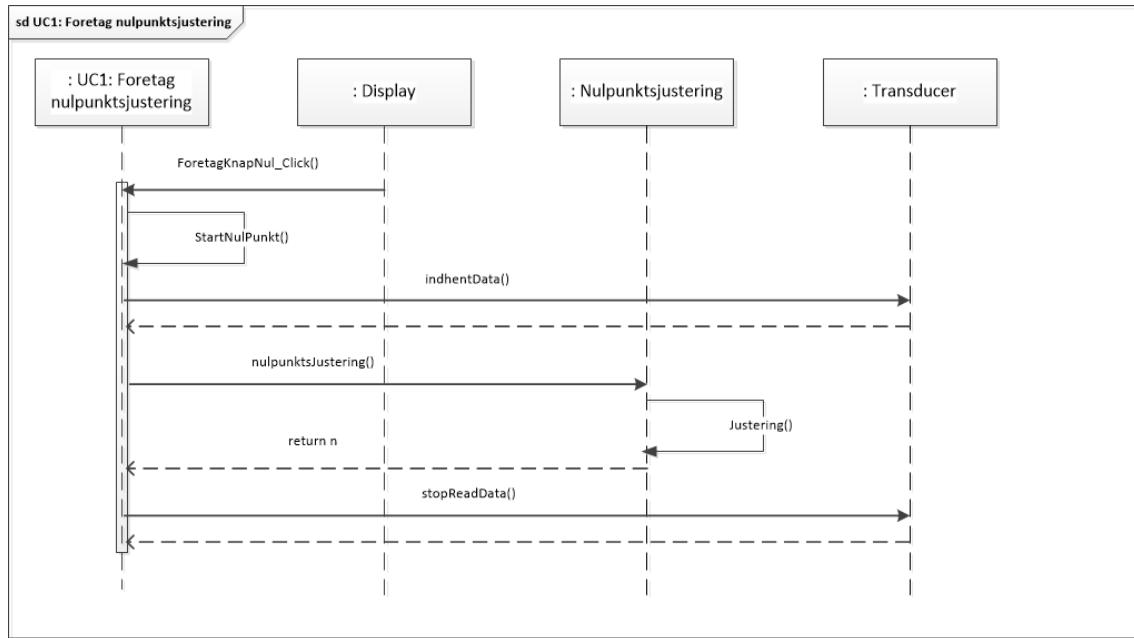
1. Grænsefladeklasse
  - a) Transducer - Indhentet data fra måleobjekt
  - b) Display - Brugergrænseflade til forsker
2. Kontrolklasse
  - a) UC1: Foretag nulpunktsjustering
  - b) UC2: Bestem kalibreringskoefficient
  - c) UC3: Start måling
  - d) UC4: Gem data
  - e) UC5: Stop måling
3. Domæneklasses
  - a) Database
  - b) Nulpunktsjustering - Bestemmer nulpunktsjusteringsværdi
  - c) Kalibrering - Bestemmer kalibreringskoefficient
  - d) Filter - Indeholder 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.

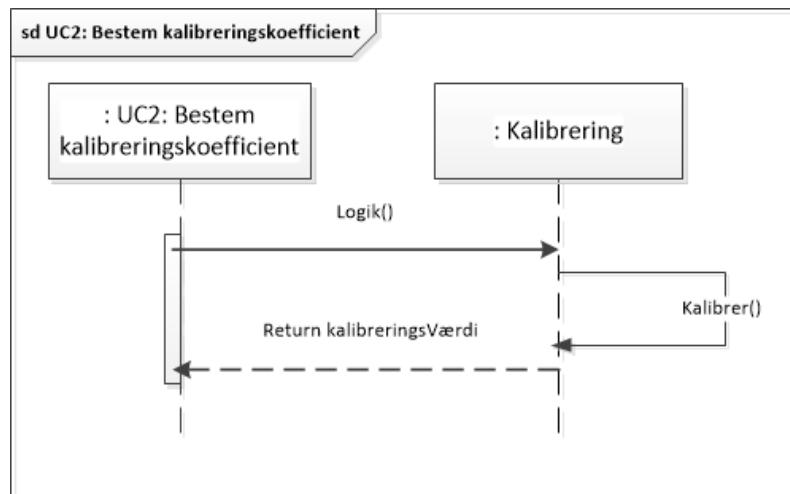
### Use Case 1



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 indhente værdi fra transduceren.

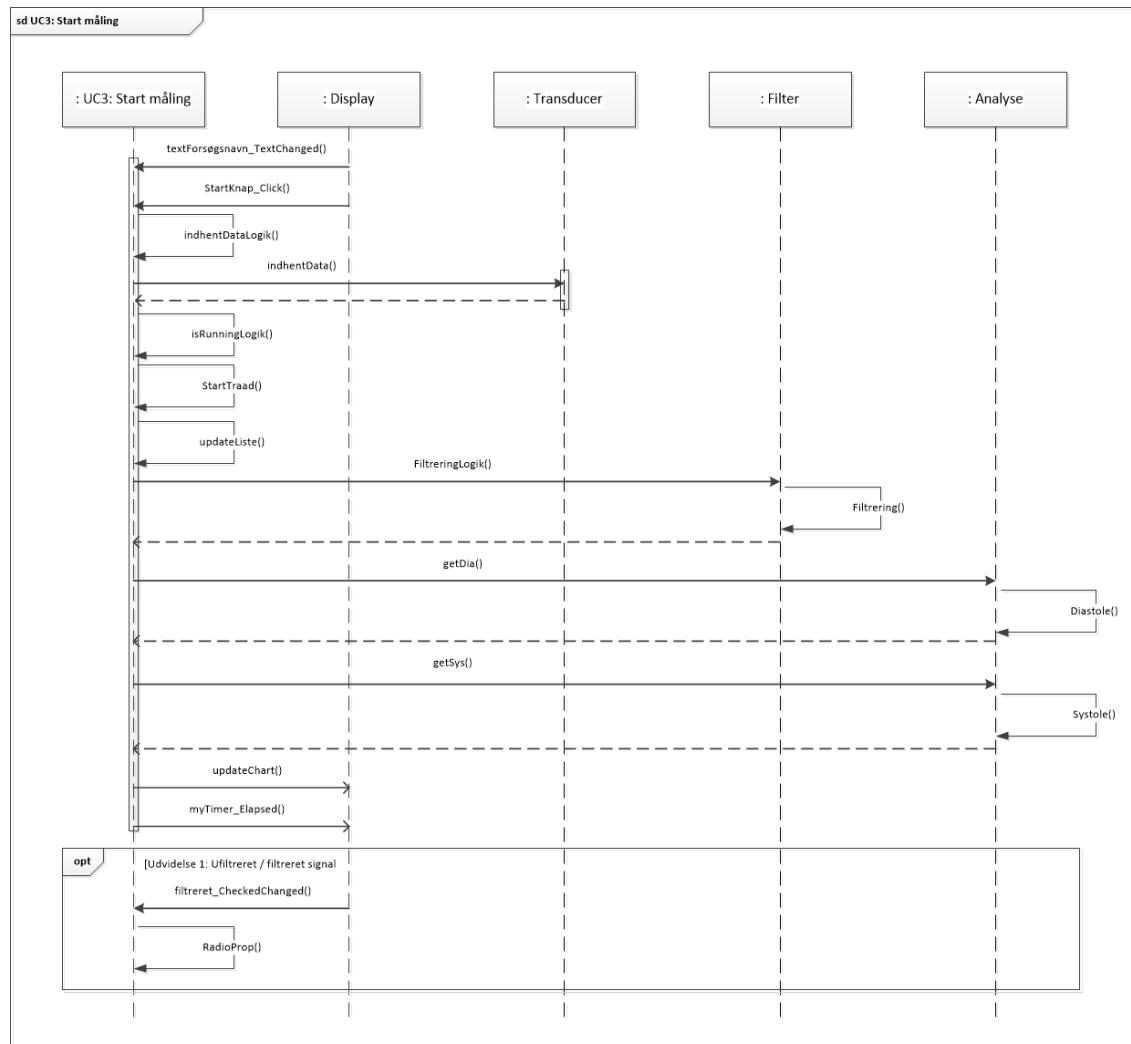
### Use Case 2



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.

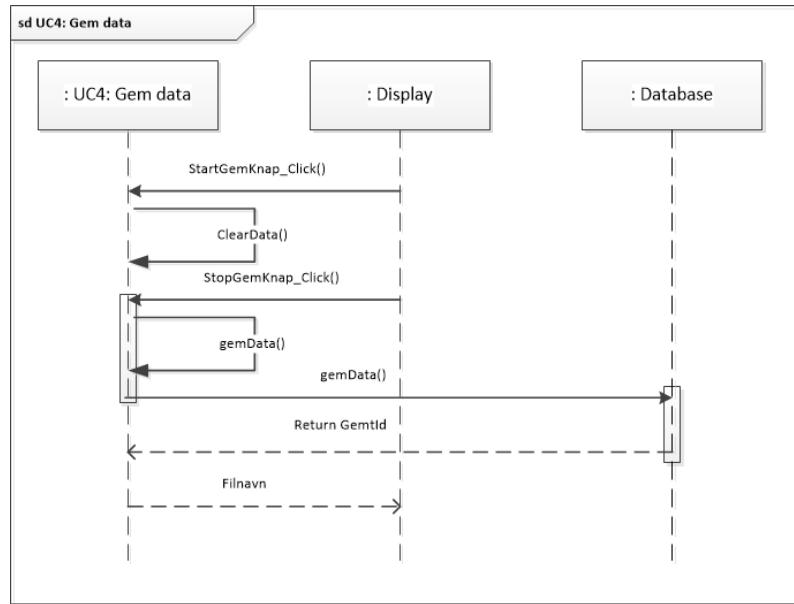
### Use Case 3



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.

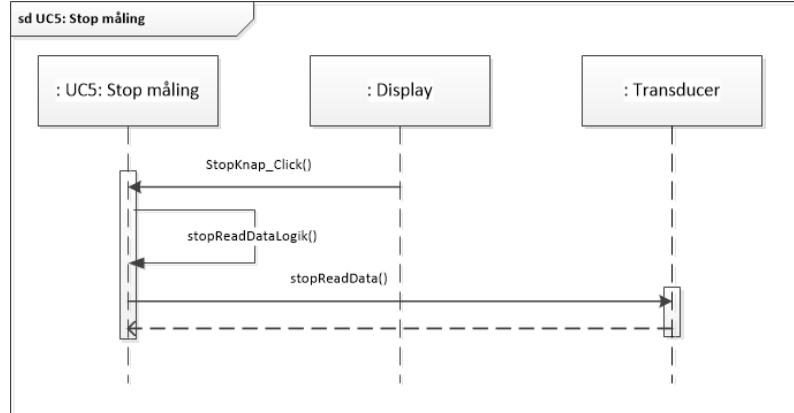
### Use Case 4



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 software-systemet.

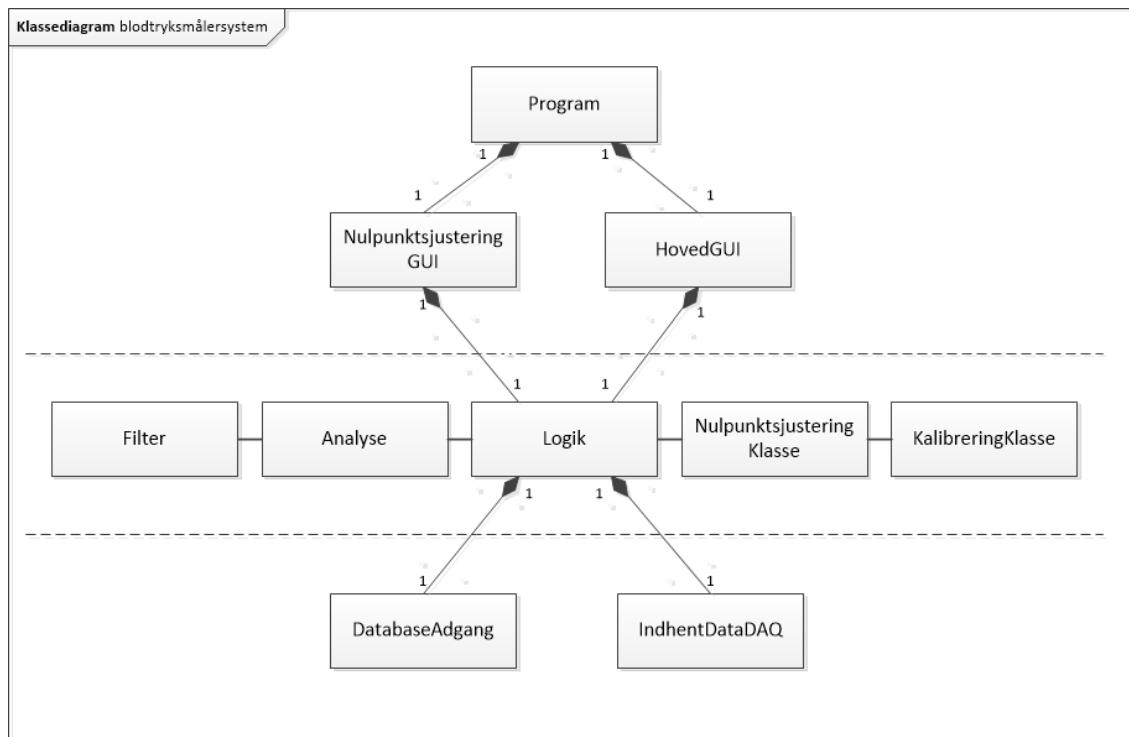
Implementeringen af softwaren sker i Visual Studio 2013 i sproget C#. Dette er valgt da programmet er et godt værktøj til at arbejde med GUI-applikationer, samt til håndtering af tråde og trådkommunikation. Tråde benyttes i softwaren da systemet, der skal implementeres, er et eventdrevet system. Det 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[? ]. 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 under "Klassediagram".

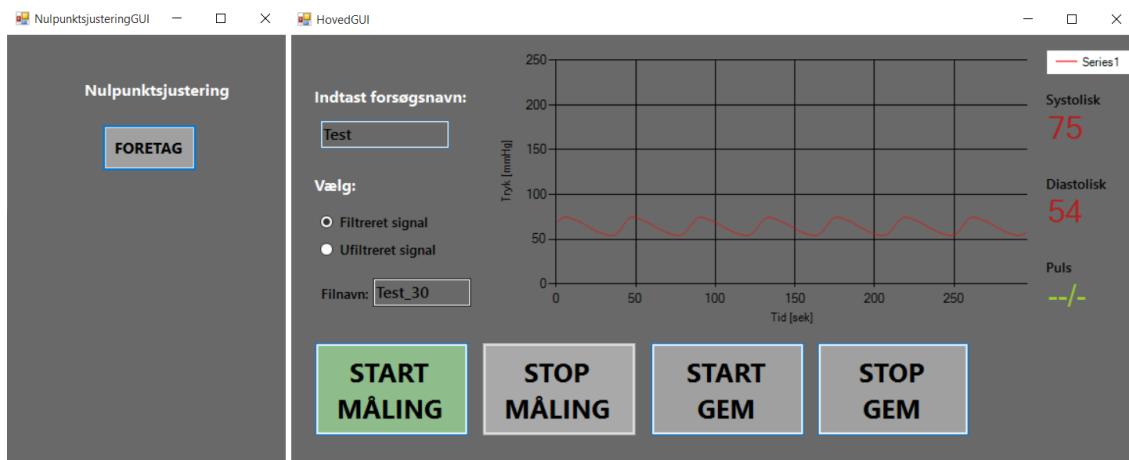


Figur 3.22: Klassediagram

### 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 udfylde 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 tekstboksen 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

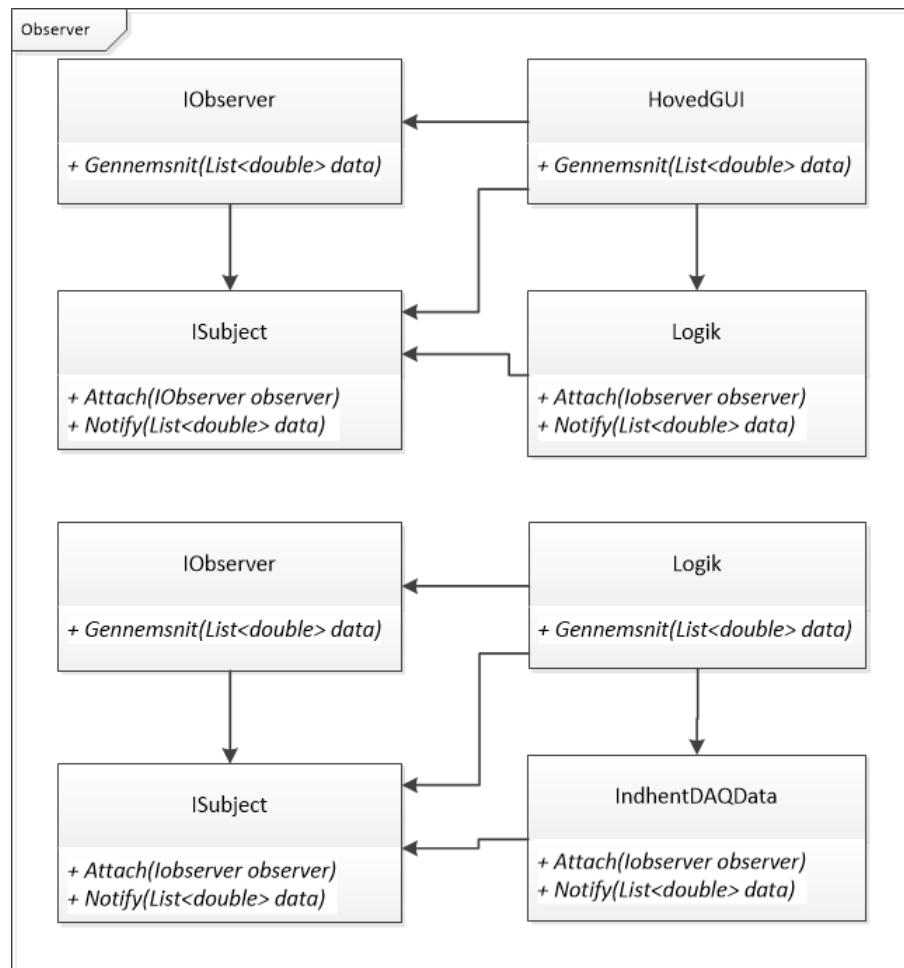
Observer og Strategy er to programmeringsmønstre. Der i samarbejde med hinanden er gode til at håndtere en overførelse af data fra et lag til et andet. Det er valgt at opbygge softwarekoden efter disse to mønstre. Observer definerer et "en til mange" relation mellem objekter således, at en ændring i 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 logik-klassen. 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 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 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 tilnærmede 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 kviksølv 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

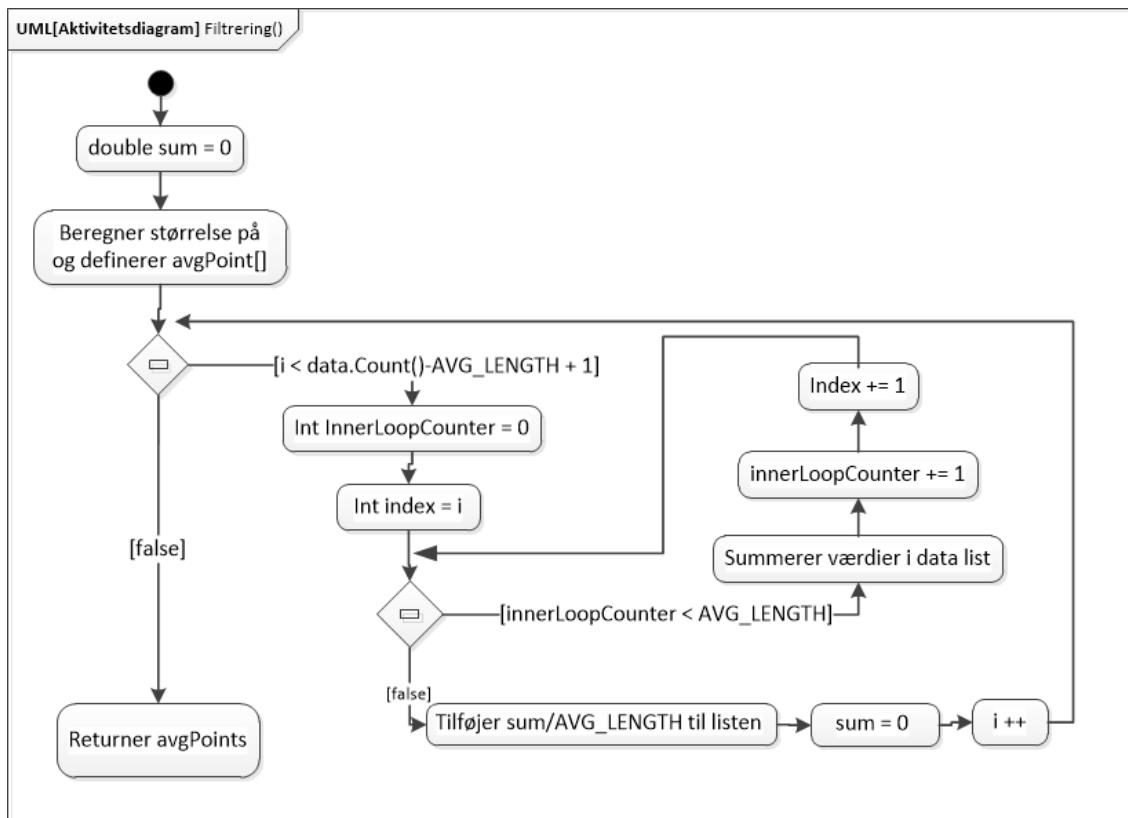
[?] Formålet med implementering af et digitalt filter er at fjerne støj fra det indhentede signal. Dette gøres ved at udglatte signalet. Til dette kan en række forskellige filtre benyttes. I projektet er det valgt at implementere et glidende middelværdifilter (moving average filter).

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 ufiltrerede 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.

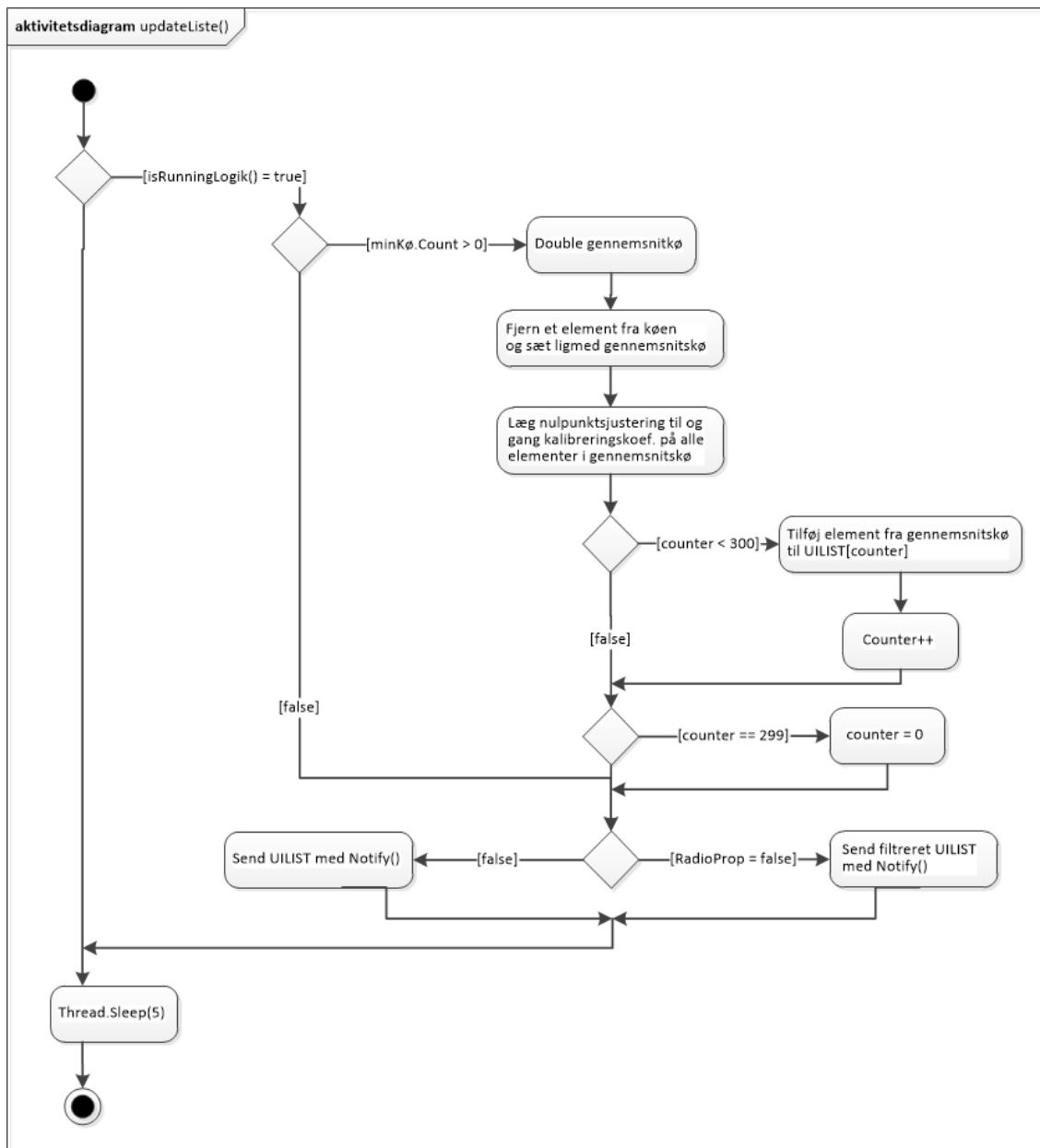
### Analyse

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 præsentationslaget med de behandlede data, som vises i grafen. I præsentations-laget er implementeret en timer, 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

Koden der bliver kigget igennem er fra når der bliver trykket Start på GUI'en og til at der bliver en graf på charten. Der er brugt debugging i Visual Studios for at sikre at koden bliver udført i korrekt rækkefølge. Stor del af koden i DAQlaget er taget fra et eksempel fra National Instruments. Gennemgangen af hvad der sker i Datalaget, nærmere DAQlaget, hvor oprettelsen til DAQ'en samt sampleliste sker, vil blive overfladisk. Koden i DAQlaget er tilpasset til vores solution. Her er der tale om samplerate og samples per channel. Samples per 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 sampleraten er henholdsvis 10 eller 20. **Fra DAQ til Logik** Programmet startes ved tryk på Start knappen på GUI'en. Det starter følgende event.

```
1 reference | Matimero, 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()
{
    DAQdata.indhentData();
}
```

Figur 3.31: Kode for Logik metoden indhentDataLogik()

Herfra starter data acquisition i DAQ laget.

IndhentData() opretter en virtual channel samt sørger for kald til InitializeDataTable() og håndterer tråde.

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

Figur 3.32: kode for IndhentDAQData metoden indhentData() kan ses i REFERENCE

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

Figur 3.33: Kode for InitializeDataTable() kan ses i REFERENCE

```
// 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() kan ses i REFERENCE

```
// 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 Logiklaget bliver der kaldt DAQData.Attach(this), derved bliver Logik objektet tilføjet som observer til subjectet IndhentDAQData.

```
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 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 Logiklaget. Her bliver listen lagt i en queue, hvorfra metoden updateListe henter alle samples fra med queue.Dequeue(). updateListe() kører på en tråd for sig selv.

Her slutter gennemgangen af hvordan sample værdierne bliver skubbet op til Logiklaget.

```

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()
{
    ≤1ms elapsed
    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 GUI laget er sat true eller false til en Radiobutton, skal der filtreres eller ikke filtreres. I tilfælde af filtrering, sendes listen først ned til Filter klassen, som parameter og bliver derefter returneret op igen 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 updateListe() bliver Notify() kaldt, med en liste som parameter, der bliver sendt op til GUI Laget.

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

Figur 3.44: Kode for metoden `Notify()` i Logik klassen. Læg mærke til at det er magen til kode, som nede i `DAQ` klassen

I GUI Laget bliver Gennemsnitsmetoden kaldt via observer pattern.

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

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

Til sidst bliver charten i GUI'en opdateret ved at `updateChart()` hele tiden 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
    {

        {
            Chart.Series["Series1"].Points.DataBindY(guiliste);
            logik.getDia();
            logik.getSys();

        }
    }
}

```

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

### 3.2.4 Integrationstest

Til sidst i projektforløbet blev en integrationstest[?] 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 gang 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 observa- tion/resultat	Faktisk observa- tion/resultat	Godkendt resultat
<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 observa- tion/resultat	Faktisk observa- tion/resultat	Godkendt resultat
<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 stemmer overens	✓

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 observa- tion/resultat	Faktisk observa- tion/resultat	Godkendt
<i>Hovedscenarie</i>			
1. Forsker indtaster Forsøgsnavn	Systemet tilgængeliggør Start Måling-knap	Når Forsker indtaster Forsøgsnavn, gør systemet Start Måling-knap tilgængelig	✓

2. Filtreret signal er valgt per default af systemet	Radiobutton til filtreret signal er checket af	Det ses at Radiobutton til filtreret 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 filtreset signal og grafen viser det filtresede 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

<sup>1</sup>Se problemrapport

Handling	Forventet observasjon/resultat	Faktisk observasjon/resultat	Godkendt
<i>Hovedscenarie</i>			
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	✓
<i>Undtagelse 1: Forsker trykker på Stop Måling-knap</i>			
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 givent 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 observa- tion/resultat	Faktisk observa- tion/resultat	Godkendt resultat
<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	Use Case 1 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	Use Case 1 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 knap er på GUI	Use Case 1 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 knap er på GUI	Use Case 1 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 tekstbokse 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 indeholder tekstbokse, 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	Kør Use Case 1 og 3	Tekstboks til filnavn er på GUI	Use Case 1 filnavn er på og 3 køres og der er en tekstboks, 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 oppe tid 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 oppe tid/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 ved accept-testen 3.03 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-aksesen målt i mmHg og x-aksesen i tid pr. sekund	Use case 1 vises i ét vindue, hvor y-aksen måles i mmHg. x-aksesen vises ikke i sekunder, men i antal samples <sup>2</sup>	✗
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 - 3.15 sekunder	Stopuret viste 3.02 sekunder	✓

<sup>2</sup>Se problemrapport

19.	Grafen for blodtryk skal kører 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 indhente signals blodtryksværdier er indenfor 0 til 250 mmHg på grafens y-akse	Der er ingen begrænsning <sup>3</sup>	(✓)
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 hurtig skifte dette. Stopuret viste 15 sekunder	✓

<sup>3</sup>Se problemrapport

23.	Softwareen 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.

Aarhus

Sted



Kundens underskrift

11/12 2015

Dato

Mette Hammer Nielsen

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.

# 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	19
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	26
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	32
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 StartKnap_Click()	40
3.31	Kode for Logik metoden indhentDataLogik()	40
3.32	kode for IndhentDAQData metoden indhentData() kan ses i REFERENCE	40
3.33	Kode for InitializeDataTable() kan ses i REFERENCE	40
3.34	Kode for IndhentDAQData metoden AnalogInCallBack() kan ses i REFERENCE	40

3.35 Kode for IndhentDAQData metoden DataToDataTable() . . . . .	41
3.36 Kode for metoden Attach(), denne bruges på samme måde i klasserne IndhentDAQData som i Logik . . . . .	41
3.37 Kode for metoden Notify() i IndhentDAQData klassen . . . . .	41
3.38 Kode for metoden Gennemsnit() i Logik klassen . . . . .	42
3.39 Kode for Logik metoden StartTraad() . . . . .	42
3.40 Kode for tråden updateUI . . . . .	42
3.41 Kode for logik metoden updateListe() . . . . .	43
3.42 Kode for metoden FiltreringLogik() . . . . .	43
3.43 Kode for klassen Filter(). Her bliver summen af 5 værdier divideret med 5 . . .	44
3.44 Kode for metoden Notify() i Logik klassen. Læg mærke til at det er magen til kode, som nede i DAQ klassen . . . . .	45
3.45 Kode for metoden i GUI Laget Gennemsnit() . . . . .	45
3.46 Kode for metoden i GUI Laget updateChart() . . . . .	46
3.47 Opstilling til Integrationstest . . . . .	47
3.48 Målinger under Integrationstest . . . . .	47