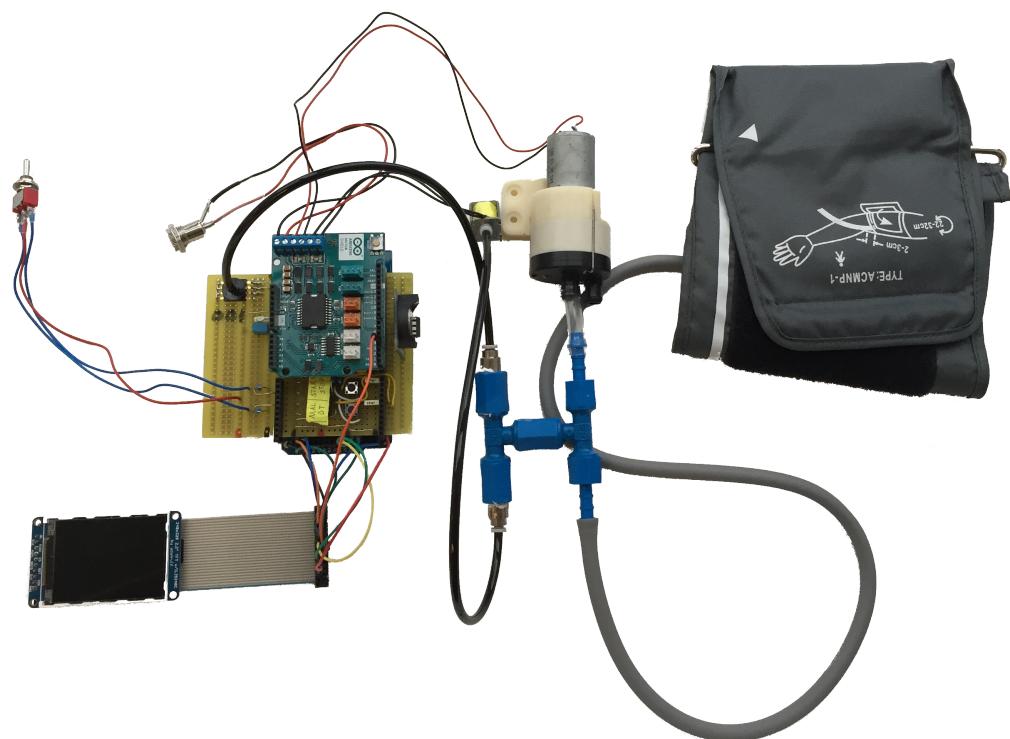


# *Remote Ischemic Conditioning*

## *Udviklingsdokumentation*



BACHELORPROJEKT  
GRUPPE 15155  
KARL-JOHAN SCHMIDT  
SIMON VAMMEN GRØNBÆK  
INGENIØRHØJSKOLEN, AARHUS UNIVERSITET

EFTERÅRET 2015



AARHUS  
UNIVERSITET

Ingeniørhøjskolen, Aarhus Universitet  
Finlandsgade 22  
8200 Aarhus N  
Tlf: 8715 0000  
<http://www.ase.au.dk/>

**Titel:**

Udviklingsdokumentation

**Projekt:**

Remote Ischemic Conditioning

**Godkendelse:**

---

Karl-Johan Schmidt

**Projektperiode:**

Juli 2015 - December 2015

**Projektgruppe:**

15155

---

Simon Vammen Grønbæk

**Deltagere:**

Simon Vammen Grønbæk

Karl-Johan Schmidt

---

Peter Johansen

**Vejledere:**

Peter Johansen

**Projektudbyder:**

Rolf Blauenfeldt

---

Rolf Blauenfeldt

**Sidetal: 126**

**Afsluttet 16-12-2015**

# Indholdsfortegnelse

I Kravspecifikation	7
II Accepttest	35
III System design	53
IV Implementeringsdokument	81



# | Indledning

Udviklingsdokumentationen er den samlede skriftlige resultat af udviklingsfasen. Dokumentet samler kravspecifikationen, accepttesten, system designet og implementeringen i et fælles dokument.

## **Formål**

Formålet med udviklingsdokumentationen er at beskrive udviklingsfasen og udarbejdelse af dokumentation som beskriver produktet i hvert udviklingstrin. Et individuelt formål findes under hvert underdokument.

## **Læsevejledning**

Dokumentet består af fire underdokumenter, hhv. kravspecifikation, accepttest, system design og implementering. En individuel læsevejledning findes under hvert underdokument. Hvert dokument har desuden sin egen indholdsfortegnelse.



# **Del I**

# **Kravspecifikation**



# Indholdsfortegnelse

<b>Kapitel 1 Indledning</b>	<b>11</b>
1.1 Formål . . . . .	11
1.2 Læsevejledning og dokumentstruktur . . . . .	11
1.3 Versionshistorik . . . . .	12
1.4 Definitioner og forkortelser . . . . .	12
1.5 Baggrund . . . . .	12
1.6 Samarbejdspartner . . . . .	13
<b>Kapitel 2 System beskrivelse</b>	<b>15</b>
2.1 Systemoversigt . . . . .	15
<b>Kapitel 3 Funktionelle krav</b>	<b>17</b>
3.1 Use case diagram . . . . .	17
3.2 Aktør beskrivelse . . . . .	18
3.3 Use cases . . . . .	19
3.3.1 Use case 1 - Konditionering . . . . .	19
3.3.2 Use case 2 - Initialiser blodtryksmåling . . . . .	20
3.3.3 Use case 3 - Mål blodtryk . . . . .	21
3.3.4 Use case 4 - Overfør data . . . . .	22
3.3.5 Use case 5 - Sikkerhedskontrol med pulsoximeter . . . . .	23
3.3.6 Use case 6 - Okklusionstræning . . . . .	24
3.3.7 Use case 7 - Afbryd . . . . .	25
3.3.8 Use case 8 - Setup . . . . .	26
<b>Kapitel 4 Ikke-funktionelle krav</b>	<b>27</b>

4.1	Microcontroller . . . . .	27
4.2	Filformat og opsætning . . . . .	27
4.3	Patient ID . . . . .	27
4.4	Hukommelse . . . . .	27
4.5	Forsyning . . . . .	28
4.6	Fysiske krav . . . . .	28
4.7	Setup . . . . .	28
<b>Kapitel 5 User interface</b>		<b>29</b>
5.1	Konditionering . . . . .	30
5.2	Okklusion . . . . .	31
5.3	Setup . . . . .	32
5.4	Bagside . . . . .	33

# 1 | Indledning

Dokumentet specificere de krav der eksisterer til produktet, Konditioneringsapparatet, som har til formål at udføre konditionerings behandling på patienter der har fået en apopleksi. Apparatet udviklings i samarbejde med læge Rolf Blauenfeldt, Neurologisk afsnit, Aarhus Universitetshospital (AUH). Med udgangspunkt i et blodtryksapparat skal produktet kunne måle blodtrykket og derefter skabe arteriel okklusion i en specifiseret tidsperiode, efterfulgt af en pause. Dette gentages i et fastsat antal cyklusser. Produktet indgår i et forskningsprojekt, hvor patienten med AIS som udgangspunkt skal behandles med RIPC så snart de møder præhospitalet, og apparatet og behandling forsættes under og efter indskrivelse på hospitalet. Kravene til Konditioneringsapparatet er fastsat over flere faser. Som udgangspunkt er kravene blevet specifiseret af projektets ophavsmand, Rolf Blauenfeldt. Dette er sket igennem flere møder og mailkorrespondancer i projektets opstartsfasen. Kravene omkring produktet også skal kunne håndtere okklusionstræning er kommet til i samarbejde med vejleder, Peter Johansen.

## 1.1 Formål

Kravspecifikation udarbejdes for at sikre enighed mellem kunden, vejleder og projektgruppen inden udviklingsfasen igangsættes. Dokumentet beskriver samtlig funktionaliteter for Konditioneringsapparatet. Udviklingen af prototypen og hele projektet er en iterativ proces og derfor kan krav opdateres undervej.

## 1.2 Læsevejledning og dokumentstruktur

Dokumentet er struktureret således at første beskrives det overordnede system som helhed. Dernæst beskrives de funktionelle krav til produktet. De funktionelle krav er struktureret som fully dressed use cases og beskriver forskellige krav som scenarier. De ikke funktionelle krav afgrænses projektet, og er beskrevet i punktform. Til sidst beskrives user interface.

### 1.3 Versionshistorik

Versions nummer	Ændring	Dato og initialer
0.1	Oprettelse af dokument	12.09.15 KJS
0.2	Splittet use cases ud i seperate .tex filer og tilføjelse af <i>samarbejdspartner</i>	25.09.15 KJS
0.3	Rettelser efter review	27.09.15 KJS
0.4	Tilføjelse af use case diagram	30.10.15 SVG
0.5	Tilføjelse af system oversigt	07.10.15 KJS, SVG
0.6	Rettelser efter samtale med Rolf	23.11.15 KJS, SVG
0.7	Rettelser efter system test	26.11.15 KJS, SVG

### 1.4 Definitioner og forkortelser

Udtryk / Forkortelse	Forklaring
RIPC	Remote ischemic pre/per/post conditioning. Længerevarende okklusion af ydre ekstremitet, efterfulgt af en deflations fase
AIS / apopleksi	Acute ischemic stroke, en pludseligt opstået neurologisk skade eller udfald på baggrund af iskæmi (nedsat blodforsyning) i hjernen
AUH	Aarhus Universitetshospital
Konditioneringsapparatet	Navnet på prototype som udvikles til at udføre RIPC
Okklusionsfase	Periode hvor på manchetten skaber arteriel okklusion
Deflationsfase	Periode der er altid er efterfulgt en okklusionsfase, hvor manchetten er deflateret i under 50mmHg
Cyklus	Forløb bestående af én <i>okklusionfase</i> og én <i>deflationsfase</i>
Gennemført afklemning	Boolean værdi der bruges til at bestemme om en cyklus er gennemfør eller ej
Tid pr cyklus	Værdi, der angiver hvor mange sekunder en cyklus skal være. For at simplificere use cases er denne værdi fastsat til 5 minutter, men det kan ændres
Antal cyklusser	Værdi, der angiver hvor mange cyklusser konditioneringen skal vare. For at simplificere use cases er denne værdi fastsat til 4 cyklusser

### 1.5 Baggrund

Beskrivelse af projektet: *Akut blodprop i hjernen (Acute Ischemic Stroke – AIS)* er en førende årsag til død og alvorlig handicap hos personer over 60 år. Intravenøs trombolysebehandling administreret indenfor 4,5 time fra symptomdebut er den nuværende bedste medicinske behandling. Grundet sikkerhedshensyn og det snævre tidsvindue er det desværre kun et fåtal af AIS patienterne, der modtager denne behandling. Målet er at op løse blodproppen og genoprette blodforsyning og dermed redde hjernevæv, der lider af

*iltmangel men endnu ikke er dødt. Om et område af hjernen dør eller står til at redde ved en blodprop afhænger ikke kun af selve blodproppe men også om hjernen er i stand til at få blod via omveje dannet af hjernens små blodkar. Et område af hjernen går til grunde med det samme (infarktkernen). Denne kerne af dødt hjernevæv kan i dagene efter en blodprop sprede sig og vokse. Der er således behov for at kunne beskytte hjernen mod iltmangel og øge andelen af hjernevæv, der overlever en blodprop. Iltmangel induceret periodevis i et fjernt organ (remote ischemic conditioning RIC) kan udføres ved at puste en blodtryksmanchet med afklemning af armen. Konditionering kan leveres som pre, per, og postconditionering, afhængig af om stimulus udøves før iltmangel, under iltmangel men før blodproppe er opløst og endelig efter blodproppe er opløst. Dyrestudier og senest kliniske studier har vist at RIC kan mindske det område af hjertet eller hjernen, der dør ved en blodprop. Det er ikke tilstrækkeligt undersøgt om RIC mindsker risikoen for handicap efter en blodprop i hjernen.*

## 1.6 Samarbejdspartner

Bachelor gruppen kunde er Rolf Blauenfeldt, Neurologisk afsnit, Aarhus Universitetshospital (AUH). Projektet er udbudt af Rolf og det er i samarbejde med ham at projektet er blevet specifiseret

Peter Johansen er vejleder for bachelor gruppen. Der afholdes faste møder med vejlederen hver anden uge, hvor der gives det status over projektet og diskuteres aktuelle problemstilling

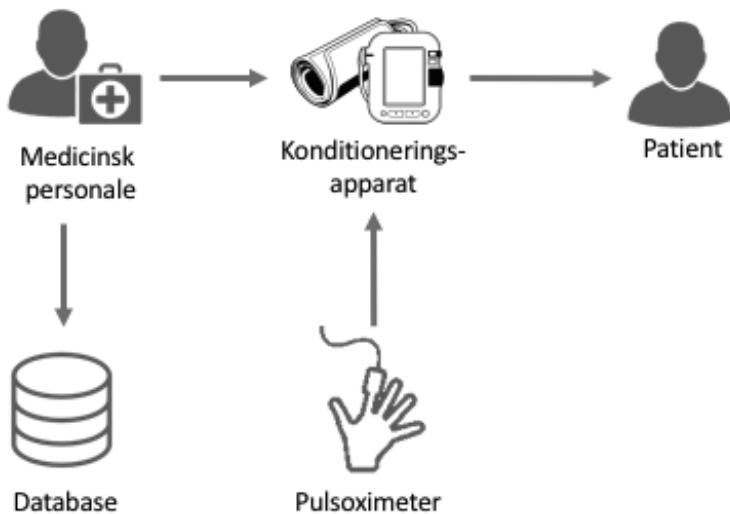
Anders Esager og Anders Toft er bachelor gruppen review partner. De har også underskrevet tavshedserklæring og har derfor mulighed for at blive sat ind i projektet og fungere som sparringspartnere.



## 2 | System beskrivelse

Systemet er beregnet til behandling af patienter med acute ischemic stroke(AIS). Formålet er pre, peri og postkonditioning af disse patienter. Systemet skal kunne lave arteriel okklusion i de øverste ydre ekstremiteter. For at sikre tilstrækkelig okklusion, skal det systoliske blodtryk først måles og derefter pumpe manchetten op til plus 25 mmHg over det målte tryk. Som minimum skal der afklemmes med et tryk på 180 mmHg. Okklusionenfasen bliver holdt konstant i 5 minutter, hvorefter trykket lukkes ud der holdes en "pause" på 5 minutter, deflationfasen. Denne process gentages indtil det antal specificeret cyklusser er kørt. Endvidere kræves der af produktet at både længden og antallet af okklusionfasen og deflationfasen skal kunne ændres løbende. For at sikre at den arterielle afklemning ikke skader patienten, skal produktet kunne indikere om der opnås tilfredsstillende perfusion af det afklemte væv efter okklusionfasen. Produktet indbefatter derfor også et pulsoximeter, der efter hver afklemning tjekker kredsløbet. Systemet skal også kunne dokumenterer behandlingsforløbene, derfor udstyres systemet med ekstern hukommelse, der gør det muligt for bruger at eksportere dataen og se oversigt over forløbet. Som sideløbende krav kan produktet også bruges til okklusionstræning. Her pumpes manchet trykket op til 100 mmHg og det tryk holdes konstant indtil okklusionsættet er færdigt.

### 2.1 Systemoversigt



*Figur 2.1.* Oversigt over systemet *Konditioneringsapparat*

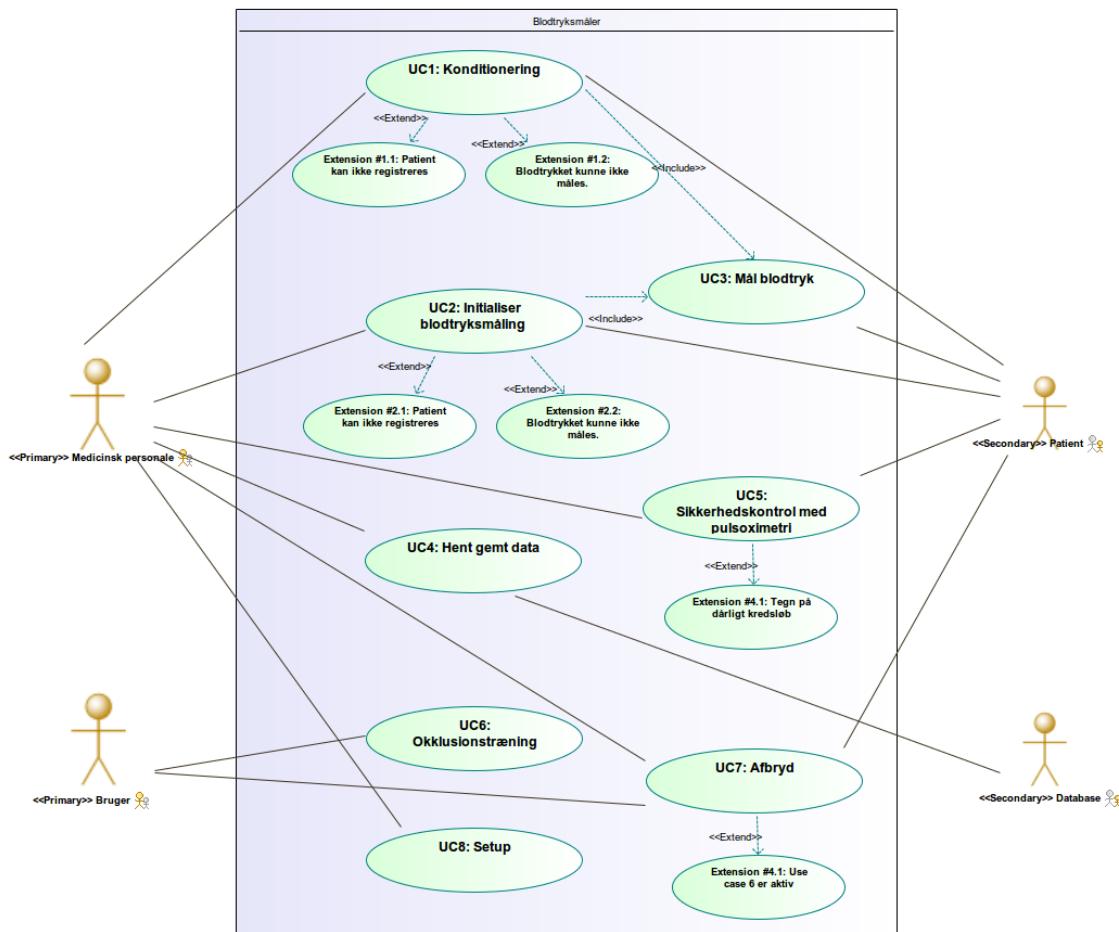


# 3 | Funktionelle krav

Dette afsnit beskriver de funktionelle krav for *Konditioneringsapparatet*. De funktionelle krav er udarbejdet som *fully dressed* use cases, der beskriver krav til produktets opførelse.

## 3.1 Use case diagram

Nedenfor illustreres alle use cases (Se figur 3.1) for systemet i et samlet diagram. Aktørerne som er beskrevet ovenfor indgår i diagrammet, og deres interaktion og roller med scenarierne er beskrevet ved hjælp af pilene



Figur 3.1. Use case diagram for *Konditioneringsapparatet*

## 3.2 Aktør beskrivelse

Systemet har tre aktør og disse er beskrevet nedenfor (Se tabel 3.1, 3.2, 3.3 og 3.4). Aktørerne skal ses som brugere eller spillere der skal interagere med scenarierne for at de lykkes.

Aktørnavn	Medicinsk personale
Aktørtype	Primær og sekundær
Beskrivelse af aktør	Aktør som påmontere manchetten og styre konditionering eller person som observere de gemte data fra behandlingsforløb

**Tabel 3.1.** Aktør beskrivelse af medicinsk personale

Aktørnavn	Patient
Aktørtype	Sekundær
Beskrivelse af aktør	En person med AIS som skal konditioneres

**Tabel 3.2.** Aktør beskrivelse af patient

Aktørnavn	Database
Aktørtype	Sekundær
Beskrivelse af aktør	Gemmer data og logfiler omkring konditioneringsforløb

**Tabel 3.3.** Aktør beskrivelse af database

Aktørnavn	Bruger
Aktørtype	Primær
Beskrivelse af aktør	Person der gør brug af konditioneringsapparatet til okklusionstræning, denne person behøver ikke besidde særlig faglig viden

**Tabel 3.4.** Aktør beskrivelse af bruger

### 3.3 Use cases

Dette afsnit beskriver alle use cases fra figur 3.1 ved hjælp af fully dressed use case.

#### 3.3.1 Use case 1 - Konditionering

Tabel 3.5 beskriver use casen for konditioneringsbehandlingen.

Mål	Gennemføre én konditioneringsbehandling
Initiering	Medicinsk personale
Aktører og interessecenter	<ul style="list-style-type: none"> <li>• Medicinsk personale(primær)</li> <li>• Patient (sekundær)</li> </ul>
Referencer	Use case 3
Antal samtlige forekomster	En til mange
Startbetingelser	<ul style="list-style-type: none"> <li>• Mode switch er sat til “<i>Konditionering</i>”</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• <i>Antal cyklusser</i> er gennemført og gemt på hukommelsen</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. <i>Medicinsk personale</i> placerer manchetten på patienten</li> <li>2. <i>Medicinsk personale</i> trykker på knappen [Start/Stop] [Undtagelse #1]</li> <li>3. Et nyt patient ID genereres [Undtagelse #2]</li> <li>4. Patient ID'et vises på skærmen</li> <li>5. Blodtrykket måles via <i>use case 3</i></li> <li>6. Blodtrykket vises på displayet og værdien gemmes i hukommelsen</li> <li>7. Manchetten fyldes med luft til et tryk på 25 mmHG over systolisk tryk (minimum 200 mmHg)</li> <li>8. Tidsstempel gemmes når systoliske tryk er opnået</li> <li>9. Trykket opretholdes i 5 minutter(Okklusion) og resterende tid vises på displayet</li> </ol> <p>(Normal forløbet fortsætter på næste side)</p>

Normal forløb	10. Deflaterer manchetten helt og forbliver i dette stadiet i 5 min(Reperfusion) Ved deflation start gemmes tidsstempel. Tid til næste okklusion vises på displayet 11. Gentag punkt 7-10 (en <i>cyklus</i> ) fire gange. Det nuværende cyklus nummer vises i displayet
Undtagelser	[Undtagelse #1] SD kortet er ikke monteret korrekt [Undtagelse #2] Et patient ID eksisterer allerede på apparatet. Der genereres ikke noget nyt patient ID.

**Tabel 3.5.** Fully dressed use case diagram over use case 1

### 3.3.2 Use case 2 - Initialiser blodtryksmåling

Denne use case beskriver hvordan en blodtryksmåling initieres. Use casen er gældende når en blodtryksmåling skal foretages uden af apparatet udfører konditioneringsbehandling, (Se tabel 3.6).

Mål	Mål et blodtryk
Initiering	Medicinsk personale
Aktører og interessecenter	<ul style="list-style-type: none"> <li>• Medicinsk personale(primaær)</li> <li>• Patient (sekundær)</li> </ul>
Referencer	Mål blodtryk(UC3)
Antal samtlige forekomster	En til mange
Startbetingelser	<ul style="list-style-type: none"> <li>• Manchetten er placeret på armen</li> <li>• Mode switch er sat til “Konditionering”</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Patientens blodtryk er målt</li> </ul>
Normal forløb	1. Brugeren trykker på [Mål blodtryk] [Undtagelse #1] 2. Et nyt patient ID genereres [Undtagelse #2] 3. Patient ID'et vises på skærmen 4. Blodtrykket måles via <i>use case 3</i>
Undtagelser	[Undtagelse #1] SD kortet er ikke monteret korrekt [Undtagelse #2] Et patient ID eksisterer allerede på apparatet. Der genereres ikke noget nyt patient ID.

**Tabel 3.6.** Fully dressed use case diagram over use case 2

### 3.3.3 Use case 3 - Mål blodtryk

Uce case 3 kan ses i fully dressed form på tabel 3.7. Denne use case beskriver forløbet under en blodtryksmåling.

Mål	Mål et systolisk, diastolisk og middel(MAP) tryk
Initiering	Konditionering (UC1) eller Initialiser blodtryksmåling (UC2)
Aktører og interesser	-
Referencer	-
Antal samtlige forekomster	En til mange
Startbetingelser	<ul style="list-style-type: none"> <li>• Patient ID er oprettet</li> <li>• Manchetten er placeret på armen</li> <li>• Mode switch er sat til “<i>Konditionering</i>”</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Blodtrykket er mål</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. Manchetten fyldes til tryk over systolisk niveau</li> <li>2. Luften lukkes gradvist ud og det systoliske tryk registreres</li> <li>3. Middel trykket (MAP) måles</li> <li>4. Det diastoliske tryk udregnes ud fra systole og MAP</li> <li>5. Blodtrykket vises på skærmen og værdien gemmes i hukommelsen med et tidsstempel</li> </ol>
Undtagelser	-

**Tabel 3.7.** Fully dressed use case diagram over use case 3

### 3.3.4 Use case 4 - Overfør data

I tabel 3.8 beskrives forløbet når data skal eksporteres fra *Konditioneringsapparatet*. Denne use case sikre at SD kortet håndteres ens hver gang data eksportes og dette mindsker fejl.

Mål	Eksportér data fra blodtryksapparat til databasen
Initiering	Medicinsk personale
Aktører og interesser	<ul style="list-style-type: none"> <li>• Medicinsk personale(primær)</li> <li>• Patient (sekundær)</li> </ul>
Referencer	-
Antal samtlige forekomster	Én pr behandlingsforløb
Startbetingelser	<ul style="list-style-type: none"> <li>• Der eksisterer en logfil på hukommelsen</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Logfilen er overført til database</li> <li>• Blodtryksapparat udstyres med formateret hukommelse og klar til næste patient</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. Tag SD kortet ud af blodtryksapparatet</li> <li>2. Sæt SD kortet i computeren og overfør filen</li> <li>3. Formatér SD kortet</li> <li>4. Sæt SD kortet tilbage i konditioneringsapparatet</li> </ol>
Undtagelser	-

**Tabel 3.8.** Fully dressed use case diagram over use case 4

### 3.3.5 Use case 5 - Sikkerhedskontrol med pulsoximeter

Denne use case viser hvordan *Konditioneringsapparatet* skal håndtere sikkerhedskontrol under et konditioneringsforløb, (Se tabel 3.9).

Mål	Sikre at patientens kredsløb tåler konditionering
Initiering	Konditionering (UC1)
Aktører og interessecenter	<ul style="list-style-type: none"> <li>• Patient (sekundær)</li> </ul>
Referencer	-
Antal samtlige forekomster	En til mange
Startbetingelser	<ul style="list-style-type: none"> <li>• Konditionering (UC1) igangværende</li> <li>• Pulsoximeteret er monteret på patients finger</li> <li>• Patient har gennemført én afklemnings cyklus</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Patients tilstand er bestemt</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. Saturation detekteres</li> <li>2. Saturation gemmes på SD-kort</li> <li>3. Saturation er tilfredsstillende [Undtagelse #1.1][Undtagelse #1.2]</li> <li>4. Behandlingen kan fortsætte</li> </ol>
Undtagelser	[Undtagelse #1.1] Tegn på dårlig kredsløb: Blodtryksapparatet stopper konditionerings forløbet [Undtagelse #1.2] Kør use case 7

**Tabel 3.9.** Fully dressed use case diagram over use case 5

### 3.3.6 Use case 6 - Okklusionstræning

Her beskrives hvordan prototypen skal udfører okklusionstræning og hvordan scenariet skal forløbe, se use case 6 i tabel 3.10. Her er den primære aktør skiftet fra *medicinsk personale* til *bruger*.

Mål	Gennemføre okklusion af venøs kredsløb under træning
Initiering	Bruger
Aktører og interessecenter	<ul style="list-style-type: none"> <li>• Bruger (primær)</li> </ul>
Referencer	-
Antal samtlige forekomster	En pr træningspas
Startbetingelser	<ul style="list-style-type: none"> <li>• Mode switch er sat til “<i>okklusionstræning</i>”</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Okklusions træningssæt gennemført</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. Montere manchetten på arm/ben</li> <li>2. Tryk på knap [Start/Stop]</li> <li>3. Manchetten pumpes op til 100mmHg</li> <li>4. Træningssættet begyndes og trykket holdes konstant på 100mmHg (+/-10mmHg)</li> <li>5. Efter træningssættet trykker <i>bruger</i> stop</li> <li>6. Manchetten deflateres</li> </ol>
Undtagelser	-

**Tabel 3.10.** Fully dressed use case diagram over use case 6

### 3.3.7 Use case 7 - Afbryd

Når et givet scenarie skal afbrydes udføres use case 7. Denne use case (Se tabel 3.11) sikre at *Konditioneringsapparatet* kan afbrydes når som helst og sørge for at manchetten tømmes for luft.

Mål	Tømme manchetten for luft og afbryde nuværende procedure
Initiering	Medicinsk personale, Patient
Aktører og interesser	<ul style="list-style-type: none"> <li>• Medicinsk personale</li> <li>• Bruger</li> </ul>
Referencer	Konditionering (UC1), Mål blodtryk (UC3) og Okklusionstræning (UC6)
Antal samtlige forekomster	En til mange
Startbetingelser	<ul style="list-style-type: none"> <li>• Konditionering (UC1) eller Okklusionstræning (UC6) er igangværende</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Behandlingen er afbrudt og manchetten er tom for luft</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. Brugeren trykker på knappen [Start/Stop konditionering]</li> <li>2. Den igangværende use case afbrydes</li> <li>3. Manchetten tømmes for luft og tidsstempel med “<i>Afklemning afbrudt = false(0)</i>” gemmes i hukommelsen [Undtagelse #1]</li> </ol>
Undtagelser	[Undtagelse #1] Use case 6 er aktiv: ingen data gemmes i hukommelsen.

**Tabel 3.11.** Fully dressed use case diagram over use case 7

### 3.3.8 Use case 8 - Setup

Use case 8 (Se tabel 3.12) beskriver forløbet når der skal ændrings i konditioneringsprotokollen. Her kan ændres i tiden pr cyklus og antallet af cyklusser.

Mål	Ændre konditioneringsforholdene
Initiation	Medicinsk personale
Aktører og interessecenter	<ul style="list-style-type: none"> <li>• Medicinsk personale</li> </ul>
Referencer	Illustration over setup
Antal samtlige forekomster	En til mange
Startbetingelser	<ul style="list-style-type: none"> <li>• Mode switch er sat til “setup”</li> <li>• Displayet er ændret til setup</li> </ul>
Slutbetingelser	<ul style="list-style-type: none"> <li>• Cyklus længden og/eller antallet af cyklusser er ændret</li> </ul>
Normal forløb	<ol style="list-style-type: none"> <li>1. Brugeren trykker på knappen [Mål blodtryk] for at vælge <i>Tid pr cyklus</i></li> <li>2. Bruger trykker på knappen [Start/Stop] for at ændre <i>Tid pr cyklus</i></li> <li>3. Bruger trykker på knappen [Mål blodtryk] for at gemme ændringen</li> <li>4. Bruger trykker på knappen [Start/Stop] for at navigere til <i>Antal cyklusser</i></li> <li>5. Ved knap tryk på [Mål blodtryk] vælges <i>Antal cyklusser</i></li> <li>6. Ved knap tryk på [Start/Stop] ændre <i>Antal cyklusser</i></li> <li>7. Brugeren trykker på knappen [Mål blodtryk] for at gemme ændringen</li> </ol>
Undtagelser	-

**Tabel 3.12.** Fully dressed use case diagram over use case 8

## 4 | Ikke-funktionelle krav

Afsnittet beskriver de ikke-funktionelle krav til produktet. De ikke-funktionelle krav adskiller sig fra de funktionelle krav, ved at beskrive kriterier der er afgørende for hvordan systemet operere. De ikke-funktionelle krav kaldes i nogen sammenhænge for *kvalitets attributter*.

### 4.1 Microcontroller

1. Atmega2560

### 4.2 Filformat og opsætning

1. Data logged i formatet .csv og hver kolonne indeholder følgende værdier og enheder:
  - a) Tidsstempel [hh:mm:ss dd-mm-yyyy]
  - b) Gennemført afklemning [Boolean]
  - c) Afklemningstryk [mmHg]
  - d) Systoliske blodtryk [mmHg]
  - e) Middelblodtryk (MAP) [mmHg]
  - f) Diastolisk blodtryk [mmHg]
  - g) Afklemning afbrudt [Boolean]
2. Der oprettes én fil pr patient, med filnavn tilsvarende det unikke patient ID og apparatets ID i følgende format: "PatientIDApparatID"

### 4.3 Patient ID

1. Består af karaktererne A-F og tallene 0-9
  - a) ID'et er fem karakterer lang: \*\*\*\*\* svarende til 1 millioner kombinationer
  - b) ID'et er ikke case sensitiv

### 4.4 Hukommelse

1. Information lagres på micro SDHC af typen:
  - a) Class 4
  - b) Fil system [fat32] og minimum 128mb

## 4.5 Forsyning

1. Konditionerings apparatet skal forsynes med 12V, min 1A
  - a) DC-connector, ydre Ø=5,5mm, indre Ø = 2,1mm
  - b) 8 stk AAA batterier (1,5V)

## 4.6 Fysiske krav

1. Knapper
  - a) [Start/Stop]
  - b) [Mål blodtryk]
  - c) Modeswitch [Okklusionstræning/Konditionering/Setup]
2. Hvert apparat udstyres med et unik serie nummer, kaldet apparat ID

## 4.7 Setup

1. Der kan ændres i tiden pr cyklus og antallet af cyklusser
  - a) *Tid pr cykles* kan sættes mellem 3 til 8 minutter og skifter med intervaller af 30 sekunder
  - b) *Antal cyklusser* kan sættes mellem 1-9 og skifter med intervaller af 1

## 5 | User interface

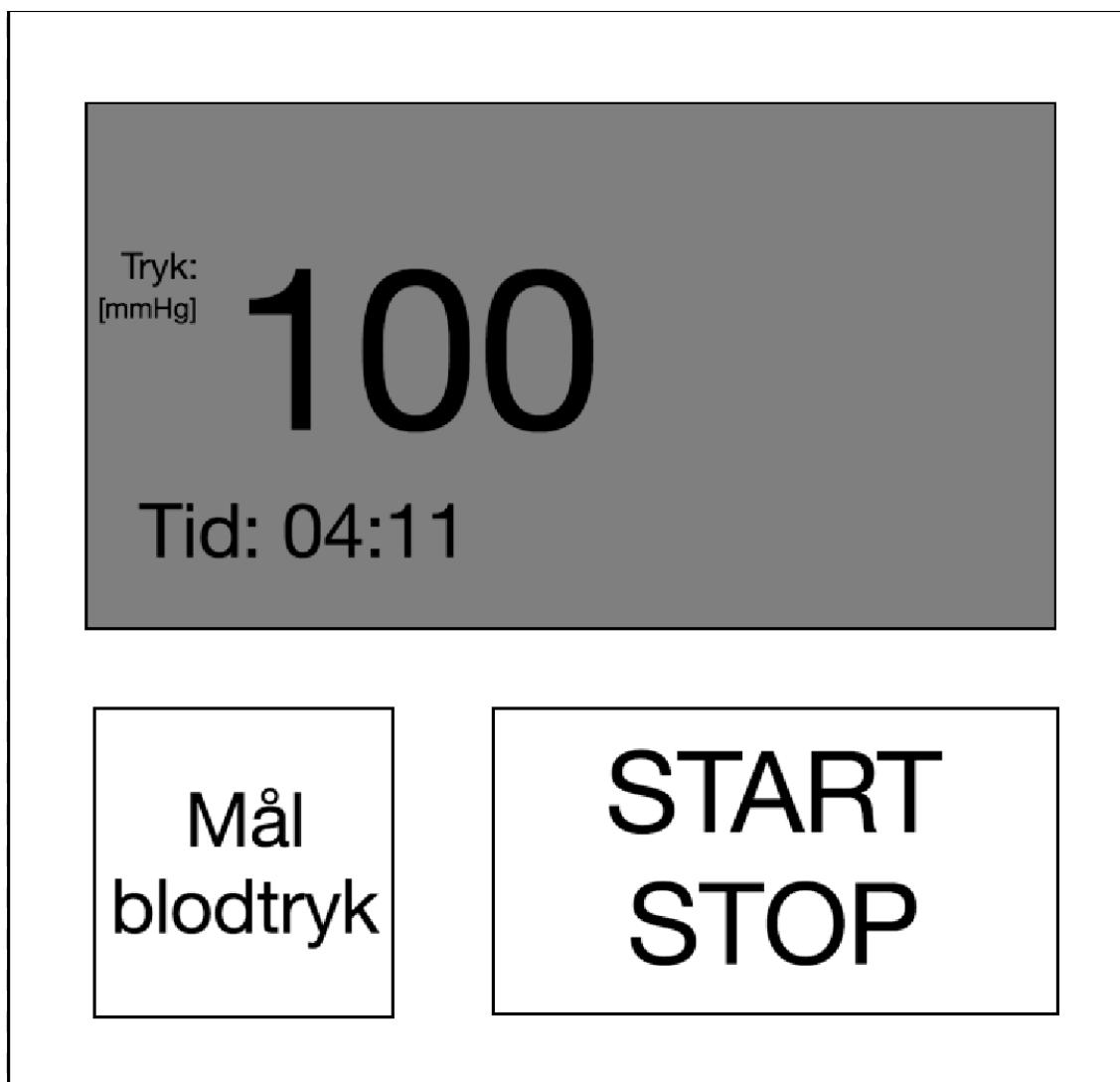
User interface består overordnede af en skærm og 2 knapper. En knap til at start og stoppe funktioner og en knap til at måle blodtryk. Til brugerfeedback er der en display hvorpå trykket, patient ID, resterende tid og antallet af gennemførte cyklusser vises. Apparatet betjenes i tre forskellige stadier “Konditionering, Okklusion og setup”. Disse tre stadier er beskrevet med hver siden illustration nedenfor. For at skifte mellem disse stadier er der en mode switch på bagsiden af apparatet, den er også beskrevet nedenfor:

## 5.1 Konditionering



*Figur 5.1.* Illustration over brugergrænseflade ved konditionering

## 5.2 Okklusion



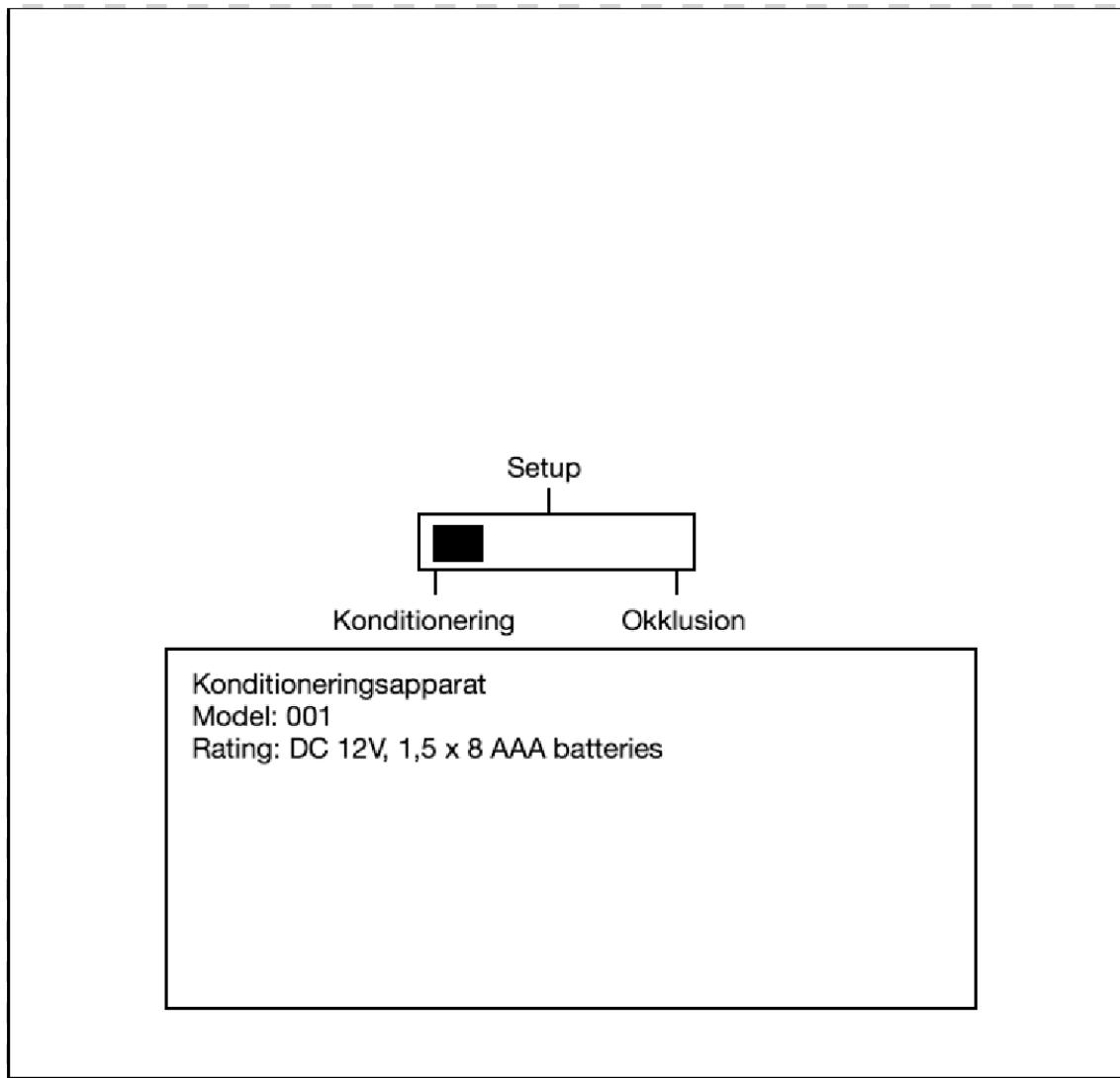
*Figur 5.2.* Illustration over brugergrænseflade ved okklusion

### 5.3 Setup



*Figur 5.3.* Illustration over brugergrænseflade ved setup

## 5.4 Bagside



*Figur 5.4.* Illustration over brugergrænseflade på bagsiden af *Konditioneringsapparatet*



**Del II**

**Accepttest**



# Indholdsfortegnelse

<b>Kapitel 6 Indledning</b>	<b>39</b>
6.1 Formål . . . . .	39
6.2 Læsevejledning og dokumentstruktur . . . . .	39
6.3 Versionshistorik . . . . .	39
6.4 Definitioner og forkortelser . . . . .	40
<b>Kapitel 7 Funktionelle krav</b>	<b>41</b>
7.1 Use case 1 - Konditionering . . . . .	41
7.2 Use case 2 - Initialiser blodtryksmåling . . . . .	43
7.3 Use case 3 - Mål blodtryk . . . . .	44
7.4 Use case 4 - Overfør data . . . . .	45
7.5 Use case 5 - Sikkerhedskontrol med pulsoximeter . . . . .	45
7.6 Use case 6 - Okklusionstræning . . . . .	46
7.7 Use case 7 - Afbryd . . . . .	47
7.8 Use case 8 - Setup . . . . .	48
<b>Kapitel 8 Ikke funktionelle krav</b>	<b>49</b>
8.1 Microcontroller . . . . .	49
8.2 Filformat og opsætning . . . . .	49
8.3 Patient ID . . . . .	50
8.4 Hukommelse . . . . .	50
8.5 Forsyning . . . . .	51
8.6 Fysiske krav . . . . .	51
8.7 Setup . . . . .	52



# 6 | Indledning

Dokumentet beskriver hvordan de specificerede krav skal testes. Accepttesten beskriver alle krav, både funktionelle og ikke funktionelle. For hvert krav er beskrevet indholdet, hvordan man tester det aktuelle krav, og hvad resultatet forventes til at være. Hver test skal godkendes af kunden. I nogle tilfælde er flere krav slæt sammen til én test. Dette gøres da nogle krav forløb er afhængig af hinanden og de ikke kan testes individuelt.

## 6.1 Formål

Formålet med accepttest er, i fællesskab med kunden, at gennemgå kravene og udføre det beskrevne test. På den måde sikre bachelorgruppen af produktet kan de krav som der er stillet til produktet.

## 6.2 Læsevejledning og dokumentstruktur

Dokumentet er struktureret lige som kravspecifikationen, de funktionelle krav er i scenarier svarende til use cases. De ikke funktionelle krav er opdelt under samme overskift som kravspecifikation, for at lette forståelsen. Accepttesten skal ses som en slags tjekliste, hvor man løber kravene igennem punkt for punkt, og så ledes sørger for at produktet lever op til kravene.

## 6.3 Versionshistorik

Versions nummer	Ændring	Dato og initialer
0.1	Oprettelse af dokumentet, opbygning af tabelstruktur	15.09.15 KJS
0.2	Overføring af main scenarios fra kravspecifikation til accepttest	23.09.15 KJS
0.3	Rettelser efter review	28.09.15 KJS
0.4	Layout rettelser	15.10.15 KJS
0.5	Tilføjet rettelser af minimumstryk og antal cyklusser efter møde med Rolf	20.11.15 KJS, SVG
0.6	Tilføjet rettelser system test	26.11.15 KJS, SVG

## 6.4 Definitioner og forkortelser

<b>Udtryk / Forkortelse</b>	<b>Forklaring</b>
RIPC	Remote ischemic pre/per/post conditioning. Længere-varende okklusion af ydre ekstremitet, efterfulgt af en deflations fase
AIS / apopleksi	Acute ischemic stroke, en pludseligt opstået neurologisk skade eller udfald på baggrund af iskæmi (nedsat blodforsyning) i hjernen
AUH	Aarhus Universitetshospital
<i>Konditioneringsapparatet</i>	Navnet på prototype som udvikles til at udføre RIPC
<i>Okklusionsfase</i>	Periode hvor på manchetten skaber arteriel okklusion
<i>Deflationsfase</i>	Periode der er altid efterfulgt en okklusionsfase, hvor manchetten er deflateret i under 50mmHg
S-105B	Blodtryksapparat der bruges som reference
<i>Tid pr cyklus</i>	Værdi, der angiver hvor mange sekunder en cyklus skal være. For at simplificere use cases er denne værdi fastsat til 5 minutter, men det kan ændres
<i>Antal cyklusser</i>	Værdi, der angiver hvor mange cyklusser konditioneringen skal vare. For at simplificere use cases er denne værdi fastsat til 4 cyklusser

# 7 | Funktionelle krav

Dette afsnit beskriver tests af de funktionelle krav.

## 7.1 Use case 1 - Konditionering

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.1.1	Medicinsk personale placerer manchetten på patienten	Manchetten sidder tæt om armen, så trykket fordeles ligeligt over hele området. Hele velcro hæfte siden skal fæstnes i filtsiden.	Manchetten trækkes løst over armen og fastspændes så den er placeret tæt-siddende omkring overarmen med 2-3 cms afstand fra albuehulen	
2.1.2	Medicinsk personale trykker på knappen [Start/Stop]	Der vises et patient ID på skærmen	Knappen [Start/Stop] trykkes og displayet observeres	
2.1.3	Et nyt patient ID genereres			
2.1.4	Patient ID'et vises på skærmen			
2.1.5	Blodtrykket måles via use case 3	Se krav nr. 2.3.1 til 2.3.5		
2.1.6	Blodtrykket vises på displayet og værdien gemmes i hukommelsen			
2.1.7	Manchetten fyldes med luft til et tryk på 25 mm-HG over systolisk tryk (minimum 200 mmHg)	Manchet-trykket er 25 mmHg over det systoliske tryk	Aflæs trykket på analogt barometer. Manchettryk - systolisk tryk = 25 mmHg	

2.1.8	Tidsstempel gemmes når det systoliske tryk er opnået	Tidsstemplaget er gemt i loggen	Kontroller tidsstempeling på SD kortet	
2.1.9	Trykket opretholdes i 5 minutter (Okklusion) og resterende tid vises på displayet	Manchet trykket skal minimum svare til systolisk tryk + 10 mmHg i 5 min	Observere analogt barometer i 5 min	
2.1.10	Deflaterer mancheten helt og forbliver i dette stadie i 5 min (Reperfusions). Ved deflation start gemmes tidsstempel. Tid til næste okklusion vises på displayet	Manchet trykket er <50 mmHg i 5 min. Kontinuerlig tids nedtælling vises på display. Tidsstempel for deflation start kan aflæses på fil.	Observér analogt barometer 5 min. Kontroller tidsstempeling på SD kortet	
2.1.11	Gentag punkt 7-01 (en <i>cyklus</i> ) fire gange. Det nuværende cyklus nummer vises i displayet	Det specificerede antal cyklusser gen-nemføres	Observér at det totale antal cyklusser er tilsvarende antallet vist på displayet (tallet til højre for cyklus nr.)	

**Tabel 7.1.** Accepttest forløb for use case 1

## Undtagelser

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.1.ex1	SD kortet er ikke monteret korrekt	Display viser besked'en: <i>Fejl: Forbind SD kortet igen, og genstart.</i>	Tag SD kortet ud og genstart apparatet	
2.1.ex2	Et patient ID eksisterer allerede på apparatet. Der genereres ikke noget nyt patient ID	Allerede eksisterende logfil vedføjes data. Ingen ny logfil genereres og det gamle ID vises på skærmen	Kør use case 1 to gange og observér antallet af logfiler, samt ID på display er det samme hver gang	

**Tabel 7.2.** Undtagelser for use case 1

## 7.2 Use case 2 - Initialiser blodtryksmåling

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.2.1	<i>Medicinsk personale</i> trykker på [Mål blodtryk]	Der vises et patient ID på skærmen	Knappen [Start/Stop] trykkes og displayet observeres	
2.2.2	Et nyt patient ID genereres			
2.2.3	Patient ID'et vises på skærmen			
2.2.4	Blodtrykket måles via <i>use case 3</i>	Se krav nr. 1.3.1 til 1.3.5		

**Tabel 7.3.** Accepttest forløb for use case 2

## Undtagelser

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.2.ex1	SD kortet er ikke monteret korrekt	Se krav nr 2.1.ex1		
2.2.ex2	Et patient ID eksisterer allerede på apparatet. Der genereres ikke noget nyt patient ID	Se krav nr 2.1.ex2		

**Tabel 7.4.** Undtagelser for use case 2

### 7.3 Use case 3 - Mål blodtryk

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.3.1	Manchetten fyldes til tryk over systoliske niveau	Trykket stemmer overens med reference apparatet	Det målte tryk sammenlignes med trykket målt fra <i>S-105B</i>	
2.3.2	Luften lukkes gradvist ud og det systoliske tryk måles	med en tolerance på: Mean error +/- 20mmHg. Se EN 1060-3, punkt 7.9		
2.3.3	Middel blodtrykket måles			
2.3.4	Det diastoliske tryk udregnes ud fra MAP og systoliske tryk			
2.3.5	Blodtrykket vises på displayet og værdien gemmes i hukommelsen med et tidsstempel	Systolisk, diastolisk og MAP vises på displayet	Gennemfør testmetode fra krav nr. 2.1.1 til 2.1.5	

**Tabel 7.5.** Accepttest forløb for use case 2

## 7.4 Use case 4 - Overfør data

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.4.1	Tag SD kortet ud af blodtryksapparatet	Der eksisterer en logfil på SD-kortet og den kopieres til det lokale drev	Indsæt SD-kort i computeren. Kontroller om filen eksisterer på kortet. Overfør filen til computeren.	
2.4.2	Sæt SD kortet i computeren og overfør filen			
2.4.3	Formatér SD kortet	SD-kortet er formateret og tømt for filer	Formatér SD til FAT32. Indsæt SD-kort i apparatet og foretag blodtryksmåling. Kontroller om logfilen oprettes på SD-kort.	
2.4.4	Sæt SD kortet tilbage i konditioneringsapparatet			

**Tabel 7.6.** Accepttest forløb for use case 4

## 7.5 Use case 5 - Sikkerhedskontrol med pulsoximeter

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.5.1	Saturation detekteres	Der kan aflæses en saturation på displayet	Testes med reference pulsoximeter <sup>1</sup>	
2.5.2	Saturation gemmes på SD-kort	Saturation et gemt på SD-kortet	Kontroller tidsstemppling og saturation på SD kortet	
2.5.3	Saturation er >90%			
2.5.4	Behandlingen kan fortsætte	Saturation er >90%	Afklem arm i 5 min og test med reference	

**Tabel 7.7.** Accepttest forløb for use pulsoximeter

<sup>1</sup>fixme Note: Tilføj reference

## Undtagelser

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.5.ex1.1	Tegn på dårlig kredsløb: Blodtryksapparatet stopper konditionerings forløbet	Saturation er <90% og der gemmes et tidsstempel for hvor der afbrydes	Afklem arm til saturationen er under niveau og observe displayet. Kontroller tidsstempeling på SD kortet	
2.5.ex1.2	Kør use case 7	Behandlingen afbrydes	Kør testmetode for use case 7	

Tabel 7.8. Undtagelser for use case 5

## 7.6 Use case 6 - Okklusionstræning

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.6.1	Montere manchetten på arm/ben	Manchetten sidder tæt om arm/ben. Hele velcro hæfte siden skal fæstnes i filtsiden	Manchetten trækkes løst over arm/ben og fastspændes så det ønskede område afklemmes. Kontrollér om manchetten er passer i størrelsen	
2.6.2	Tryk på knap [Start/Stop]	Luftpumpen startet	Knappen [Start/- Stop] trykkes og observer at manchetten udvides	
2.6.3	Manchetten pumpes op til 100mmHg	Trykket i manchetten er 100 mmHg med en tolerance på +20/-10 mmHg	Observere analogt barometer i 3 min	
2.6.4	Trænings-sættet begyndes og trykket holdes konstant på 100mmHg (+/- 5mmHg)			

2.6.5	Tryk på knap [Start/Stop]	Manchet trykket er <10 mmHg efter 1 min.	Knappen [Start/Stop] trykkes og observer at trykket på det analog barometer	
2.6.6	Manchetten deflates			

**Tabel 7.9.** Accepttest forløb for use case 6

## 7.7 Use case 7 - Afbryd

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.7.1	Brugeren trykker på knappen [Start/Stop]	Use casen afbrydes og manchetten tømmes for luft	Start use case 1 eller 6 og på et vilkårligt tidspunkt tryk på knappen [Start/Stop]	
2.7.2	Den igang-værende use case afbrydes			
2.7.3	Manchetten tømmes for luft og tidsstempel med “ <i>Afklemning afbrudt = false(0)</i> ” gemmes i hukommelsen	Manchet trykket er <50 mmHg efter 1 min. Tidsstempel for “ <i>Afklemning afbrudt = false(0)</i> ” gemmes i fil	Observere analogt barometer 1 min. Kontroller tidsstempeling på SD kortet	

**Tabel 7.10.** Accepttest forløb for use case 7

## Undtagelser

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.7.ex1	Use case 6 er aktiv: ingen data gemmes i hukommelsen	Der gemmes ingen data	Kør use case 6 efterfulgt af use case 7 og observér logfilen	

**Tabel 7.11.** Undtagelser for use case 7

## 7.8 Use case 8 - Setup

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
2.8.1	Brugeren trykker på knappen [Mål blodtryk] for at vælge Tid pr cyklus	Ved knaptryk på [Mål blodtryk] vælges “Tid pr cyklus” og det valgte område begynder at blinke	Tryk på knappen [Mål blodtryk] og observér displayet	
2.8.2	Bruger trykker på knappen [Start/-Stop] for at ændre Tid pr cyklus	Værdien i det valgte område ændres med 30s per tryk.	Tryk på knappen [Start/Stop] og observér ændringen	
2.8.3	Bruger trykker på knappen [Mål blodtryk] for at gemme ændringen	Værdien gemmes og det valgte område stopper med at blinke	Tryk på knappen [Mål blodtryk] og observér displayet. Start use case 1 og kontroller okklusionstid	
2.8.4	Bruger trykker på knappen [Start/-Stop] for at navigere til Antal cyklusser	Ved knaptryk på [Start/Stop] flyttes den firkantede markør på displayet til “Antal cyklusser”	Tryk på knappen [Start/Stop] og observér ændringen	
2.8.5	Ved knap tryk på [Mål blodtryk] vælges Antal cyklusser	Ved knaptryk på [Mål blodtryk] vælges “Antal cyklusser” og det valgte område begynder at blinke	Tryk på knappen [Mål blodtryk] og observér displayet	
2.8.6	Ved knap tryk på [Start/Stop] ændre Antal af cyklusser	Værdien i det valgte område ændres med 1 per tryk.	Tryk på knappen [Start/Stop] og observér ændringen	
2.8.7	Bruger trykker på knappen [Mål blodtryk] for at gemme ændringen	Værdien gemmes og det valgte område stopper med at blinke	Tryk på knappen [Mål blodtryk] og observér ændringen. start use case 1 og tjek total antal cyklusser	

Tabel 7.12. Accepttest forløb for use case 8

# 8 | Ikke funktionelle krav

I dette afsnit beskrives hvordan der udføres accepttest på de ikke-funktionelle krav.

## 8.1 Microcontroller

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.1.1	Type: Atmega2560	Atmega2560	Kontroller type nr. på microcontroller	

**Tabel 8.1.** Testprotokol for microcontroller

## 8.2 Filformat og opsætning

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.2.1	Data logged i formatet .csv og hver kolonne indeholder følgende værdier og enheder:	Logfil er kommasepareret og at enhederne stemmer overens. of filen er af type .csv	Inspicer logfil i texteditor (Gedit, notepad, textedit osv.)	
3.2.1a	Tidsstempel [hh:mm:ss dd-mm-yyyy]			
3.2.1b	Gennemført afklemning [Boolean]			
3.2.1c	Afklemnings-tryk [mmHg]			
3.2.1d	Systoliske blodtryk [mmHg]			
3.2.1e	Middel-blodtryk (MAP) [mmHg]			
3.2.1f	Diastolisk blodtryk [mmHg]			
3.2.1g	Afklemning afbrudt[Boolean]			

3.2.2	Der oprettes én fil pr patient, med filnavn tilsvarende det unikke patient ID og apparatets ID i følgende format: "PatientIDApparatID"	En enkel fil eksisterer på SD-kortet. filnavnet består af "PatientIDApparatID"	Kør use case 2 flere gange og observer antallet og navngivningen af logfil(er)	
-------	--	--	--	--

**Tabel 8.2.** Testprotokol for filopsætning

### 8.3 Patient ID

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.3.1	Består af karaktererne A-F og tallene 0-9	PatientID består af karaktererne	Kør use case 2 og observer navngivningen af logfil	
3.3.1a	ID'et er fem karakterer langt: ***** svarende til 1 millioner kombinationer	A-Z og tallene 0-9	Visuel inspektion af logfilen	
3.3.1b	ID'et er ikke case sensitiv			

**Tabel 8.3.** Testprotokol for patient ID

### 8.4 Hukommelse

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.4.1	Information lagres på micro SDHC af typen:	SD kortet er af typen micro SDHC, class 4, fat32 og minimum 128mb	Tag SD kortet ud og se specifikationer	
3.4.1a	Class 4			
3.4.1b	Fil system [fat32] og minimum 128mb			

**Tabel 8.4.** Testprotokol for hukommelse

## 8.5 Forsyning

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.5.1	Konditioneringsapparatet skal forsynes med 12V, min 21	Systemet forsynes med 12 volt og minimum 1A	Aflæs værdierne på adapteren	
3.5.1a	DC-connector, ydre Ø=5,5mm, indre Ø = 2,1	Connectoren har målene: ydre Ø=5,5mm, indre Ø = 2,1 mm	Mål med skydelære	
3.5.1b	8 stk AAA batterier (1,5V)	8 stk AAA batterier	Visuel inspektion	

**Tabel 8.5.** Testprotokol for forsyning

## 8.6 Fysiske krav

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.6.1	Knapper			
3.6.1a	[Start/Stop]	Knapperne er tilstede på apparatet	Visuel inspektion	
3.6.1b	[Mål blodtryk]			
3.6.2	Hvert apparat udstyres med et unik serie nummer, kaldet apparat ID			

**Tabel 8.6.** Testprotokol for knapper og serie nummer

## 8.7 Setup

Krav nr.:	Handling	Forventet resultat	Testmetode	Resultat
3.7.1	Der kan ændres i tid pr cyklus og antallet af cyklusser	<i>Tid pr cyklus og antal cyklusser</i> kan ændres i det specificerede intervaller	Kør use case 8 og observer hvor meget værdierne ændres og hvilke værdier der kan vælges	
3.7.1a	Tid pr cykles kan sættes mellem 3 til 8 minutter og skifter med intervaller af 30 sekunder			
3.7.1b	Antal cyklusser kan sættes mellem 1-9 og skifter med intervaler af 1			

**Tabel 8.7.** Testprotokol for setup

# Del III

## System design



# Indholdsfortegnelse

<b>Kapitel 9 Indledning</b>	<b>57</b>
9.1 Formål . . . . .	57
9.2 Projektreferencer . . . . .	57
9.3 Læsevejledning og dokumentstruktur . . . . .	57
9.4 Versionshistorik . . . . .	57
9.5 Definitioner og forkortelser . . . . .	58
<b>Kapitel 10 Systemets dele</b>	<b>59</b>
10.1 Microcontroller . . . . .	59
10.2 Manchetten . . . . .	59
10.3 User interface, knapper og displays . . . . .	59
10.4 Power system . . . . .	59
10.5 Pumpe . . . . .	60
10.6 Ventil . . . . .	60
10.7 Tryksensor . . . . .	60
10.8 SD kort . . . . .	60
10.9 Pulsoximeter . . . . .	60
<b>Kapitel 11 Arkitektur</b>	<b>61</b>
11.1 4+1 view architecture . . . . .	61
11.2 Logic . . . . .	63
11.2.1 Block definition diagram . . . . .	63
11.2.2 Domænemodel . . . . .	64
11.2.3 State machine diagram . . . . .	65

11.3 Process . . . . .	69
11.3.1 Sekvensdiagrammer . . . . .	69
11.3.2 Konditionering - UC1 . . . . .	70
11.3.3 Initialiser blodtryksmåling - UC2 . . . . .	71
11.3.4 Mål blodtryk - UC3 . . . . .	72
11.3.5 Overfør data - UC4 . . . . .	73
11.3.6 Sikkerhedskontrol med pulsoximeter - UC5 . . . . .	74
11.3.7 Okklusionstræning - UC6 . . . . .	74
11.3.8 Afbryd - UC7 . . . . .	75
11.3.9 Setup - UC8 . . . . .	76
11.4 Implementation . . . . .	77
11.4.1 Hardware . . . . .	77
11.4.2 Software . . . . .	78
11.5 Deployment . . . . .	79

# 9 | Indledning

Designet beskrivelsen giver en formel præsentation og forklaring af systemet. Her beskrives hvordan systemet er organiseret, hvilke strukturelle elementer der indgår og hvordan elementerne interagerer med hinanden. Der lægges både vægt på software og hardware, samt deres grænseflade. System designet beskriver hvordan *Konditioneringsapparatet* er opbygget både hardware og software mæssigt.

## 9.1 Formål

System designdokumentet har til formål at beskrive og give forståelse for systemet. Dokumentet fastlægger overordnede softwarekomponentet og hardwarekomponentet, samt strukturen og grænsefladerne mellem disse. Dokumentet udgør en plan for, hvordan systemet skal udvikles og hvilke undersystemer det skal bestå af.

## 9.2 Projektreferencer

- Reference til kravspecifikation
- Reference til accepttest

## 9.3 Læsevejledning og dokumentstruktur

Dokumentet ligger sig tæt op af kravspecifikation, da disse krav ligger til grunde for hvad systemet skal kunne. For at give en struktureret gennemgang af system arkitekturen gøres der brug af modellen “*4+1 view architecture*”, der beskriver systemet fra flere forskellige vinkler. Forklaring af modellen kan læses i kapitel 11.1. Der gøres som udgangspunkt brug af SysML til at beskrive systemet. Alt SysML udvikles og skrives på engelsk.

## 9.4 Versionshistorik

Versions nummer	Ændring	Dato og initialer
0.1	Oprettelse af system arkitekturen og skabelon	01.10.15 KJS
0.2	Tilføjet beskrivelse af <i>Logical view</i>	08.10.15 KJS
0.3	Rettelse efter fælles gennemgang	19.10.15 SVG
0.4	Rettelser efter review	23.10.15 KJS
0.5	Opdatering af sysML diagrammer	26.11.15 SVG

## 9.5 Definitioner og forkortelser

<b>Udtryk / Forkortelse</b>	<b>Forklaring</b>
UML	Unified Modeling Language, sprog til forklaring af software arkitektur
SysML	System Modeling Language, sprog til forklaring af system arkitektur
PWM	Pulse-width modulation
Modeswitch	Knap til at styre hvilket program Konditionersapparatet skal køre

# 10 | Systemets dele

Dette afsnit beskriver systemet, *Konditioneringsapparats*, fysiske dele og deres funktionelit.

## 10.1 Microcontroller

Styring af alle systemets dele. Her processeres brugerens interagering med *Konditioneringsapparat* og handlingerne eksekveres. Microcontrolleren er en AtMega32 og styringen af chippen skrives i C++.

## 10.2 Manchetten

Trykmanchet til at skabe okklusion af armen. Manchetten skal kunne holde trykket, som skabes af pumpen. Manchetten kobles til apparatet via en lufttæt slange.

## 10.3 User interface, knapper og displays

Brugefladen består af et display hvor blodtryk, antal okklusioner, resterende tid og mm. vises. Displayet skal bruges til at give brugeren feedback og fx. informere det medicinske personale hvor lang tid der er indtil konditioneringen er færdig.

På *Konditioneringsapparatet* er der to knapper [Start/Stop] og [Mål blodtryk]. Disse knapper bruges til at initierer konditioneringsbehandling, blodtryksmålinger og okklusionstræning. På bagsiden af apparatet sidder desuden en Modeswitch, hvor brugeren kan skifte mellem *Okklusionstræning*, *Konditionering*, eller *Setup*.

## 10.4 Power system

Forsyning af systemet foregår med 8 batterier af typen AAA for at opnå en spænding på 12V. Systemet forsynes med en batteriløsning, for at gøre det mere mobilt.

Foruden at forsyne apparatet, er power system også bestående af et motorshield. Når microcontroller fx ønsker at starte pumpen, sørger motorshieldet for at leve det korrekte spænding.

## 10.5 Pumpe

Består af en motor og et luftindtag. Pumpe kan både bruges til at skabe tryk og vakuum. Pumpen skal bruges til at inflatere manchetten til måling af blodtryk og til okklusion af armen, både under konditionering og under træning. Pumpen skal forsynes med 12 V og hastigheden kan styres med PWM.

## 10.6 Ventil

Ventilen indgår i systemet til at nedregulere trykket i manchetten. Ventilen er “Normally closed”. Funktionen af ventilen under en blodtryksmåling er gradvis at lukke trykket ud, så det er muligt at registrere oscillationerne og det aktuelle tryk. Under okklusion har ventilen en anden funktion, her indgår ventilen i reguleringen.

## 10.7 Tryksensor

En 5 V tryksensor, der bruges til registrering af trykket i manchetten og til efter regulering. Tryksensoren skal også registrere oscillationerne, der skabes i manchetten når trykket er omkring systolisk niveau og ved middeltrykket. Ved okklusionstræning skal tryksensoren bruges til at holde trykket konstant omkring 100 mmHg.

## 10.8 SD kort

Apparatet udstyres med ekstern hukommelse, for at det er muligt for *Konditioneringsapparatet* at gemme information omkring behandlingsforløbet. Der er valgt et SD kort, fordi når behandlingen er færdig, er det muligt at skifte SD kortet ud, og på den måde have backup af information og det er nemmere at overføre informationen.

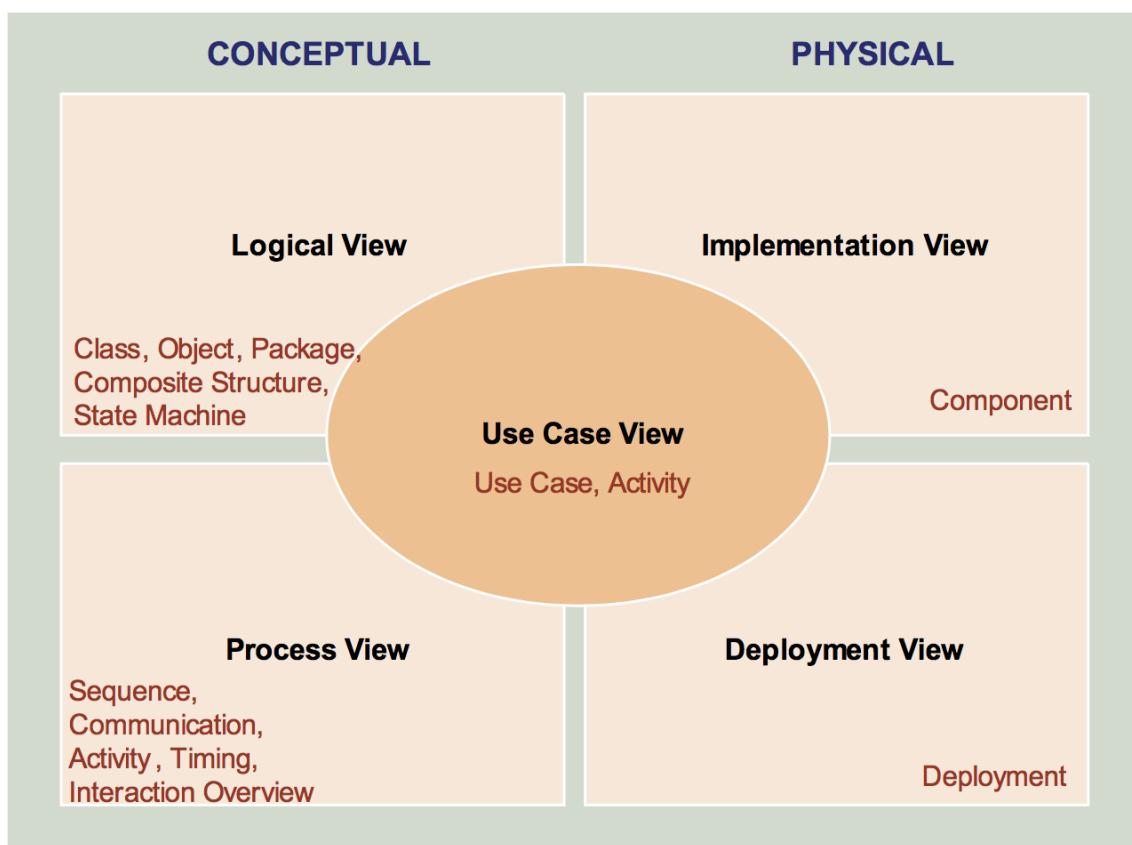
## 10.9 Pulsoximeter

Som undersystemet i *Konditioneringsapparatet* indgår et pulsoximeter, der skal bruges til overvågning af patientens tilstand under konditioningsbehandling. Pulsoximeteret leverer en saturation efter hver endt okklusion og den saturation er med til at bestemme om patientens kredsløb kan tåle behandlingen.

# 11 | Arkitektur

## 11.1 4+1 view architecture

Denne model beskriver arkitekturen af software baserede systemer. For at skabe en fyldestgørende gennemgang af systemet gøres brug af fire forskellige synsvinkler. Disse synsvinkler har til formål at tilfredsstille alle interesserter og sørge for at alle parter forstår systemet. Eksempler på parter kunne være kunden, projektleder eller udviklere. Med udgangspunkt i use cases består modellen af følgende punkter:



*Figur 11.1.* 4+1 view architecture model

*Logical view:* Denne synsvinkel beskriver systemets funktionalitet via centrale elementer, mekanismer og stadier.

*Process view:* Beskæftiger sig med de dynamiske aspekter af systemet, forklarer system processer, hvordan de kommunikerer, og fokuserer på systemets opførsel i drift.

*Implementation view:* Denne vinkel involverer udviklerens perspektiv og beskæftiger sig med hvordan software implementeres.

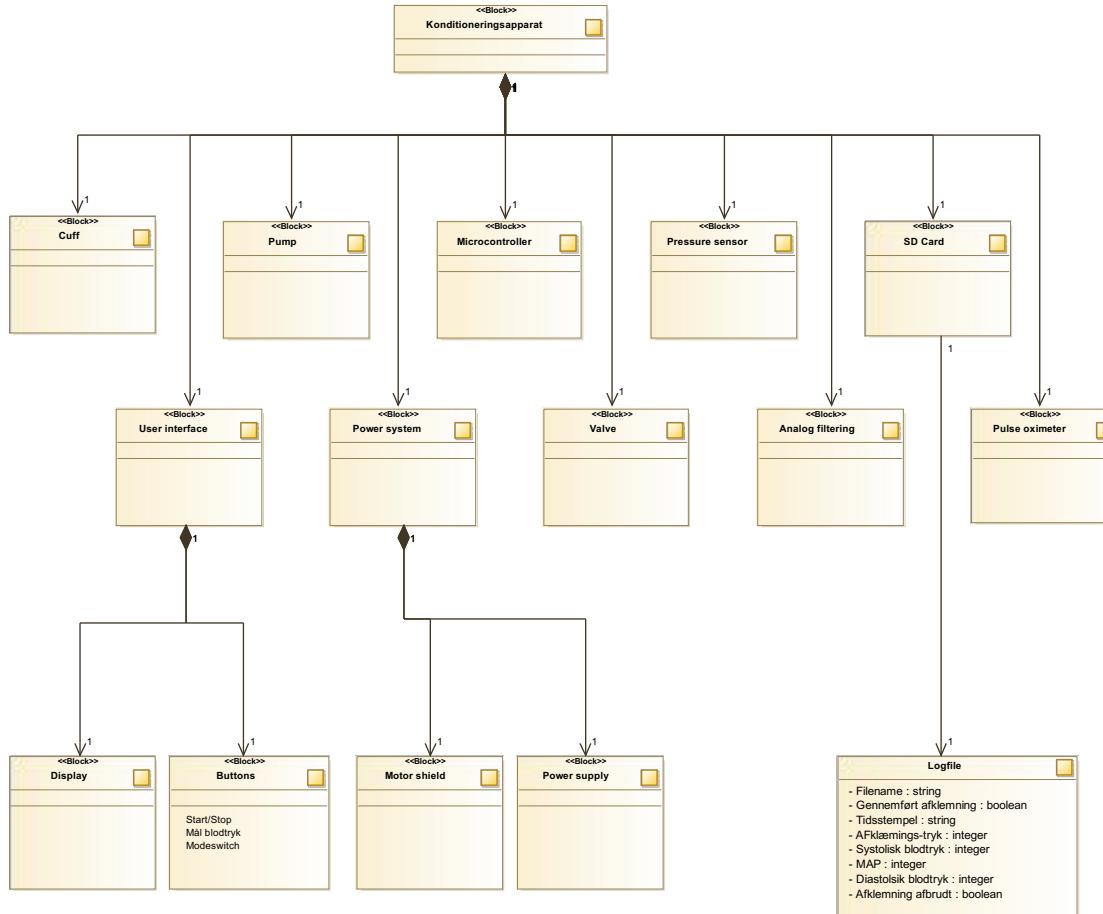
*Deployment view:* Beskriver systemet fra en fysisk synsvinkel, blandt andet hvordan eksekveringen af softwares skal foregå på apparatet og hvordan systemets fysisk setup skal se ud.

Modellen “*4+1 view architecture*” er beregnet primært til software baserede udviklingsprojekter og derfor bruges den som en retningslinje og inspiration til systemet arkitekturen. Da Konditioneringsapparatet er en prototype som involvere både hardware og software er modellen blevet tilpasset dertil. Endvidere vil systemet blive præsenteret og gennemgået ved hjælp af SysML standarden, selvom modellen er lavet til UML.

## 11.2 Logic

### 11.2.1 Block definition diagram

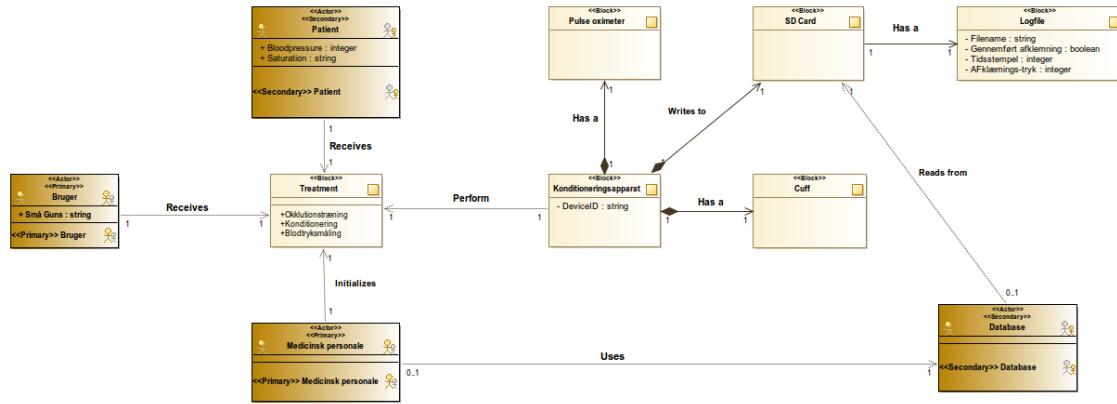
Blokdiagrammer giver et indblik på den overordnede strukturen af *Konditioneringsapparatet*. Hver kasse skal ses som en del der indgår i systemet



**Figur 11.2.** Block definition diagram over *Konditioneringsapparatet*

### 11.2.2 Domænemodel

Diagrammer beskriver det systemet som helhed. Ved gennemgang af alle use cases findes væsentlig navneord og disse oprettet som konceptuelle klasser. Det konceptuelle klasser er derefter oversat til engelsk.

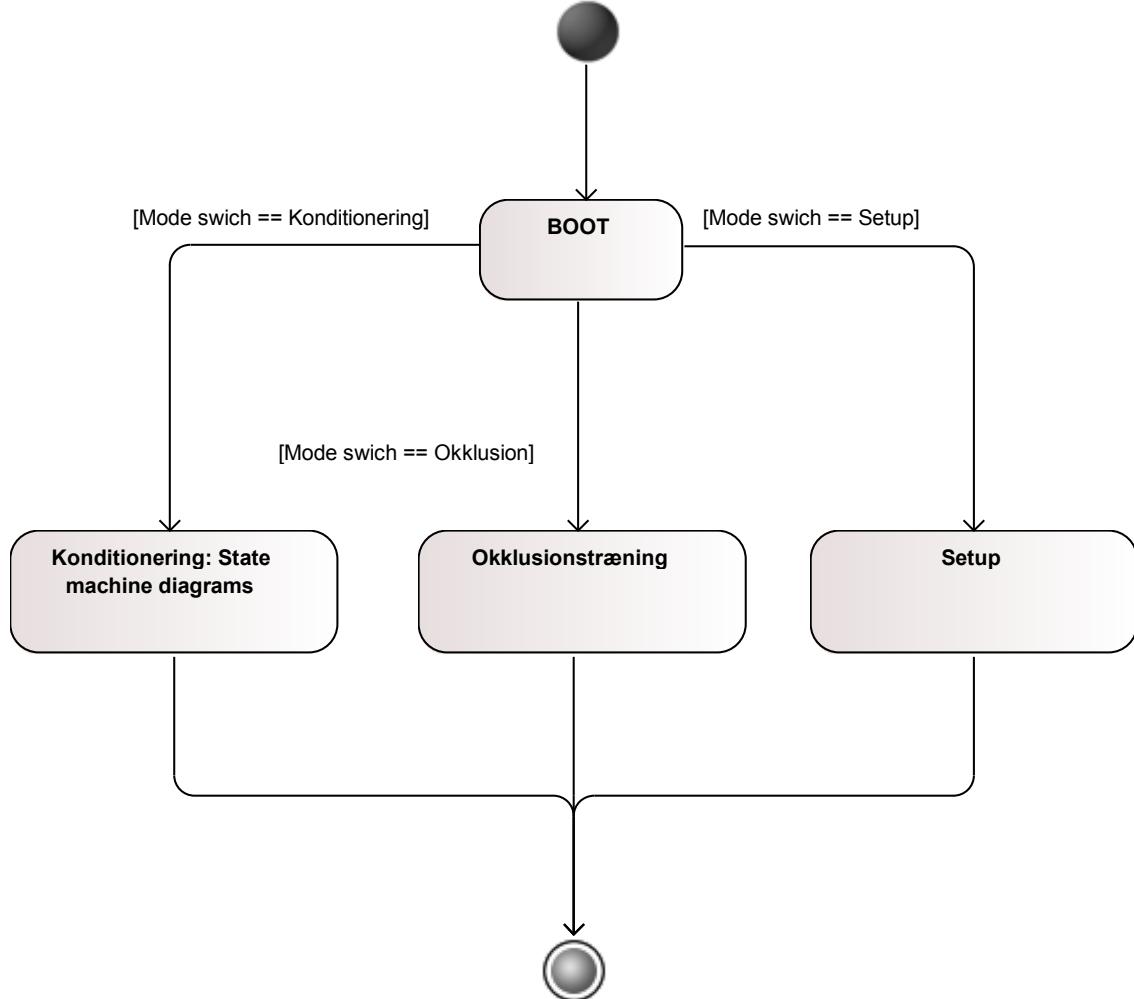


Figur 11.3. Domæne model over *Konditioneringsapparatet*

### 11.2.3 State machine diagram

For at hjælpe med forståelsen af *Konditioneringsapparats* opbygning er der udarbejdet sekvensdiagrammer over systemets tre programmer hhv. konditionering, okklusionstræning og setup. Der er desuden lavet et overordnede sekvensdiagram over hele systemet. Sekvensdiagrammerne beskæftiger sig med hvilke stadier programmet operer i, og hvilke parametre der skal være opfyldt før der skiftes til et andet stadie.

#### Boot



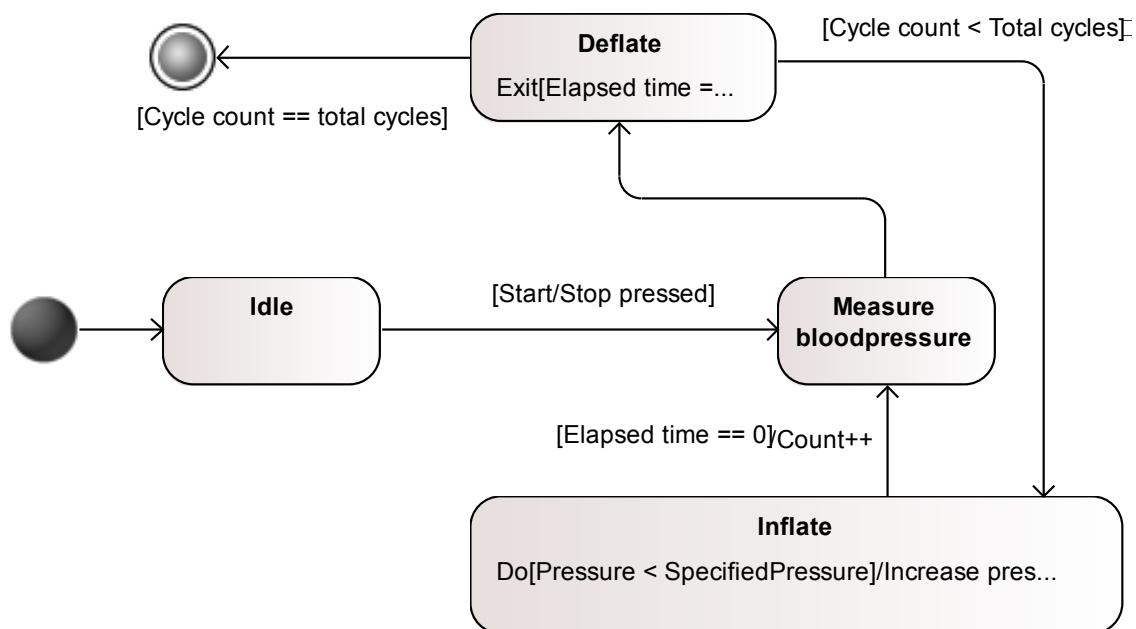
*Figur 11.4.* State machine diagram over programvalg på *Konditioneringsapparatet*

## Konditionering

I programmet *Konditionering* har brugeren mulighed for både at starte en konditioneringsbehandling eller måle et blodtryk, derfor er der udarbejdet et state machine diagram for hvert senarie. Beskrivelse af stadierne:

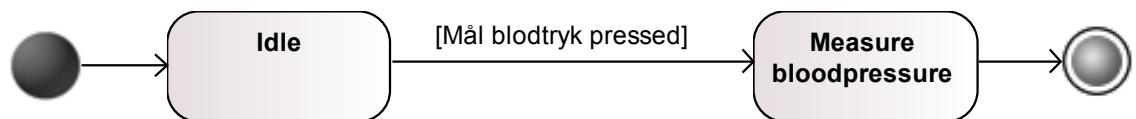
- **Idle:** Stadie som programmet befinder sig i, før *Medicinsk personale* har foretaget noget.
- **Deflate:** Reperfusionsfasen, efter en endt afklemning befinder systemets sig i denne fase.
- **Measure blood pressure:** Måling af blodtryk, ved knaptryk på [Start/Stop], efter en endt afklemningsfase eller hvis *medicinsk personale* trykker på [Mål blodtryk].
- **Inflate:** Okklusionsstadiet, her befinder systemet sig hver gang en afklemning skal foretages.

Ved knaptryk på [Start/Stop]



*Figur 11.5.* State machine diagram over Konditioneringsforløb

Ved knaptryk på [Mål blodtryk]

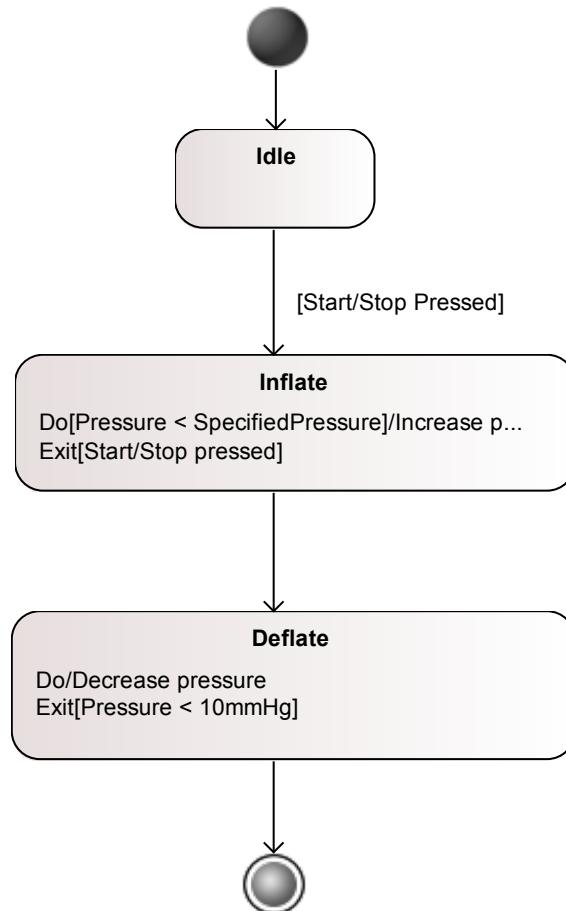


*Figur 11.6.* State machine diagram over blodtryksmåling

## Okklusion

Beskrivelse af staderne:

- **Idle:** Stadie som programmet befinner sig i, før brugeren har foretaget noget.
- **Inflate:** Ved knaptryk på [Start/Stop] skifter systemet til dette stadie, her holdes trykket konstant på 100mmHg.
- **Deflate:** Dette stadie er manchetten tømt for luft.

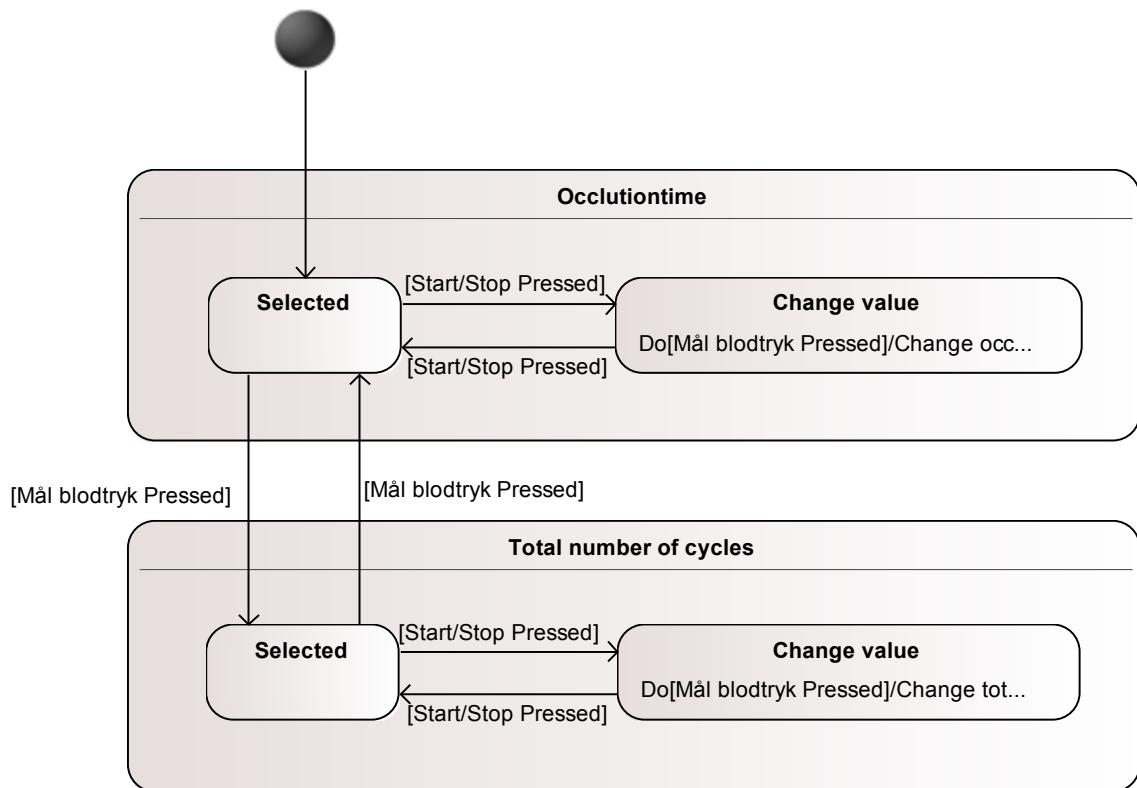


*Figur 11.7.* State machine diagram over okklusionsforløb

## Setup

Dette state machine diagram beskriver stadierne for programmet *Setup*. Dette program har ingen slut stadie, cirklen med en cirklen inden i, da dette program ikke kan forlades, uden at apparatet slukkes. Beskrivels af stadier

- **Occlusion time:** Stadie hvor *Medicinsk personale* kan ændre tidsintervallet for hvor længe en afklemnings skal være.
  - **Total number of cycles:** Opsætning af hvor mange afklemnings/reperfusions cyklusser systemet skal udføre.
  - **Selected og Change value:** er begge stadier hvor *Medicinsk personale* enten skifter mellem *Tid pr cyklus* og *Antal cyklusser* eller ændringen af disse værdier.



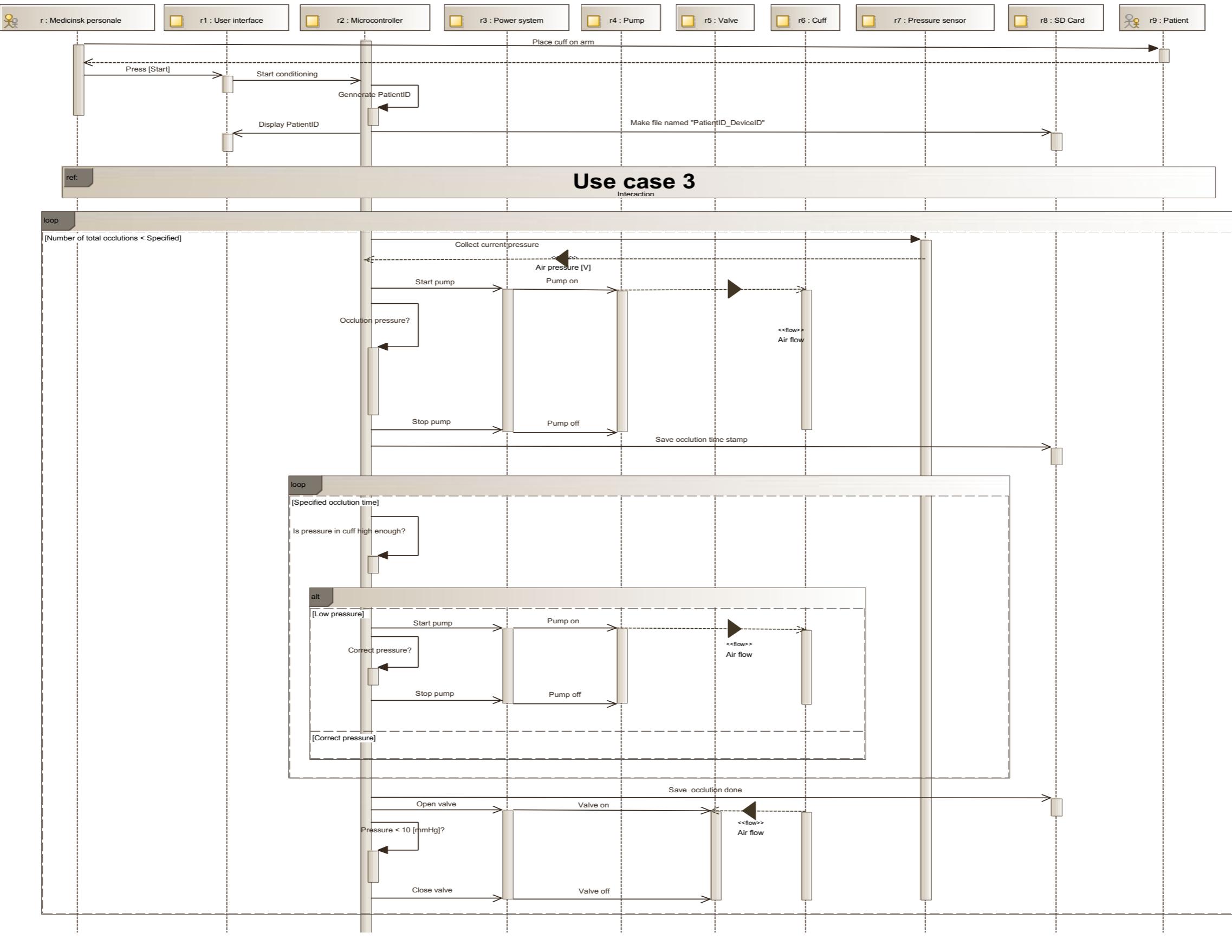
**Figur 11.8.** State machine diagram over setup forløbet

## 11.3 Process

### 11.3.1 Sekvensdiagrammer

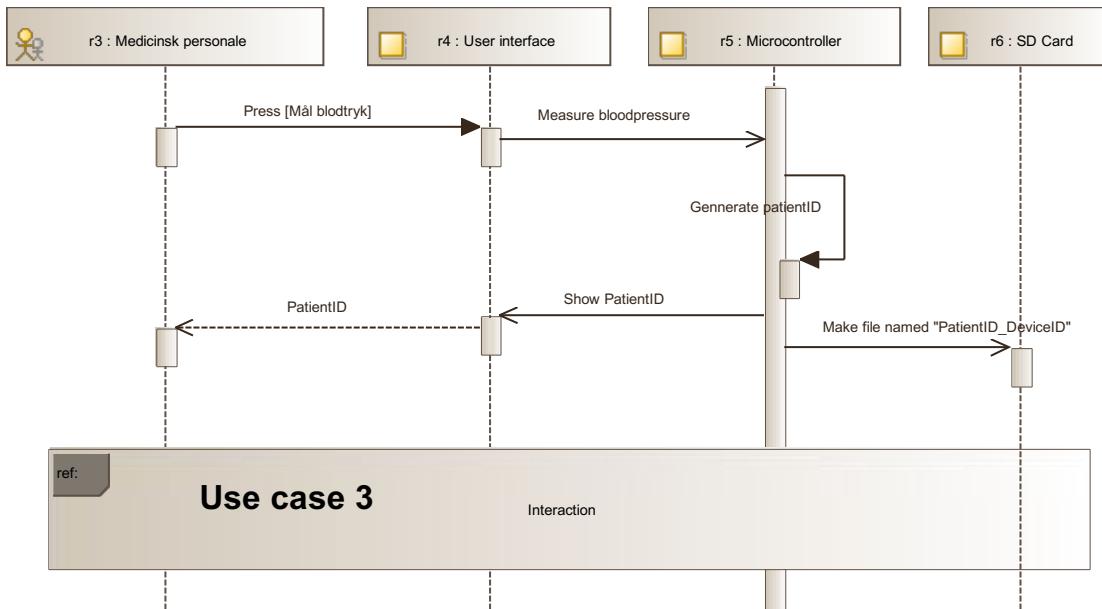
Der er udarbejdet et sekvensdiagram for hver use case. Sekvensdiagramerne beskriver hvordan systemets dele og aktører interagerer med hinanden, og hvilke processer der sker ved disse interaktioner. For at simplificere store sekvensdiagrammer gør nogle af dem brug af andre use cases, dette ses fx. i sekvensdiagrammet for use case 1.

## 11.3.2 Konditionering - UC1



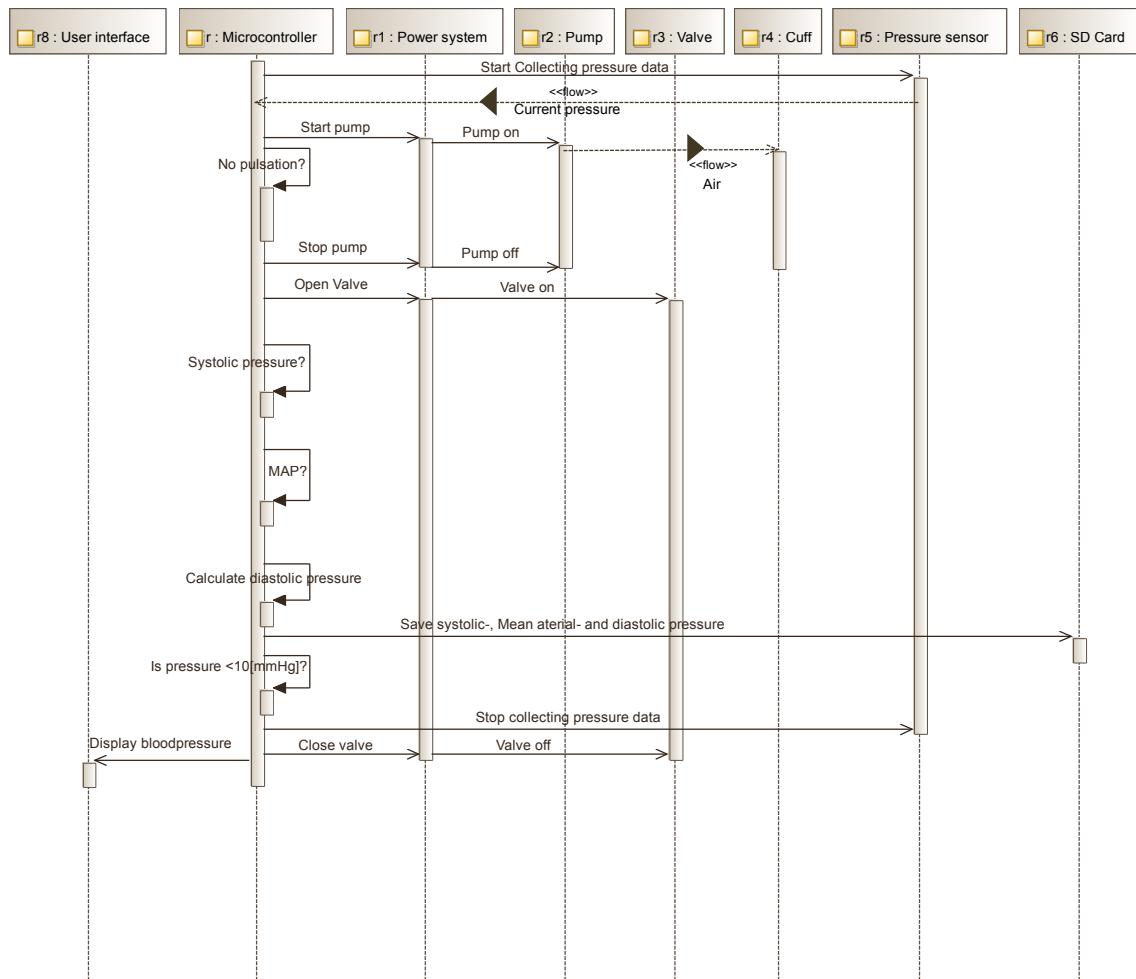
Figur 11.9. Sekvensdiagram over forløb i use case 1

### 11.3.3 Initialiser blodtryksmåling - UC2



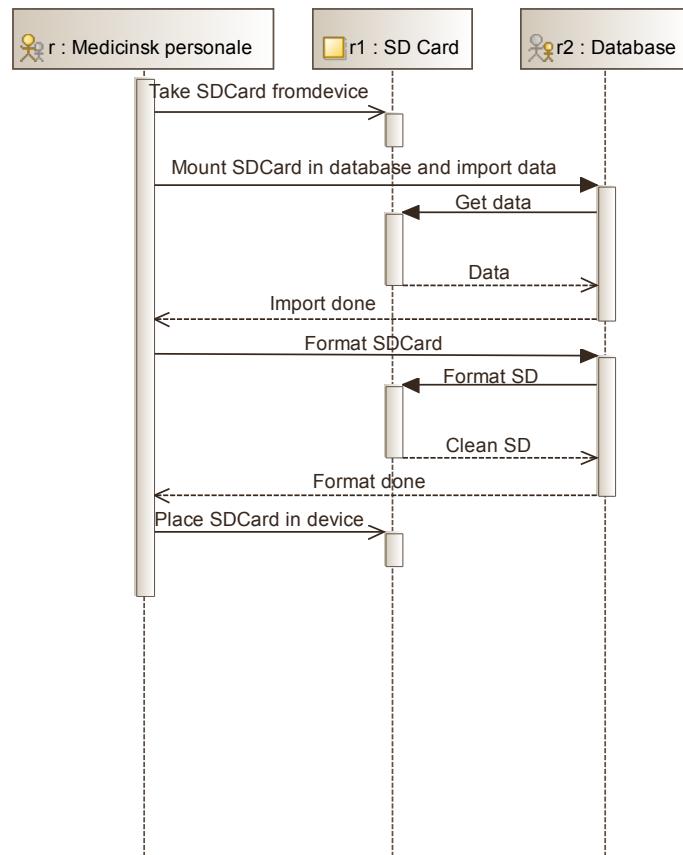
*Figur 11.10.* Sekvensdiagram over forløb i use case 2

### 11.3.4 Mål blodtryk - UC3



**Figur 11.11.** Sekvensdiagram over forløb i use case 3

### 11.3.5 Overfør data - UC4

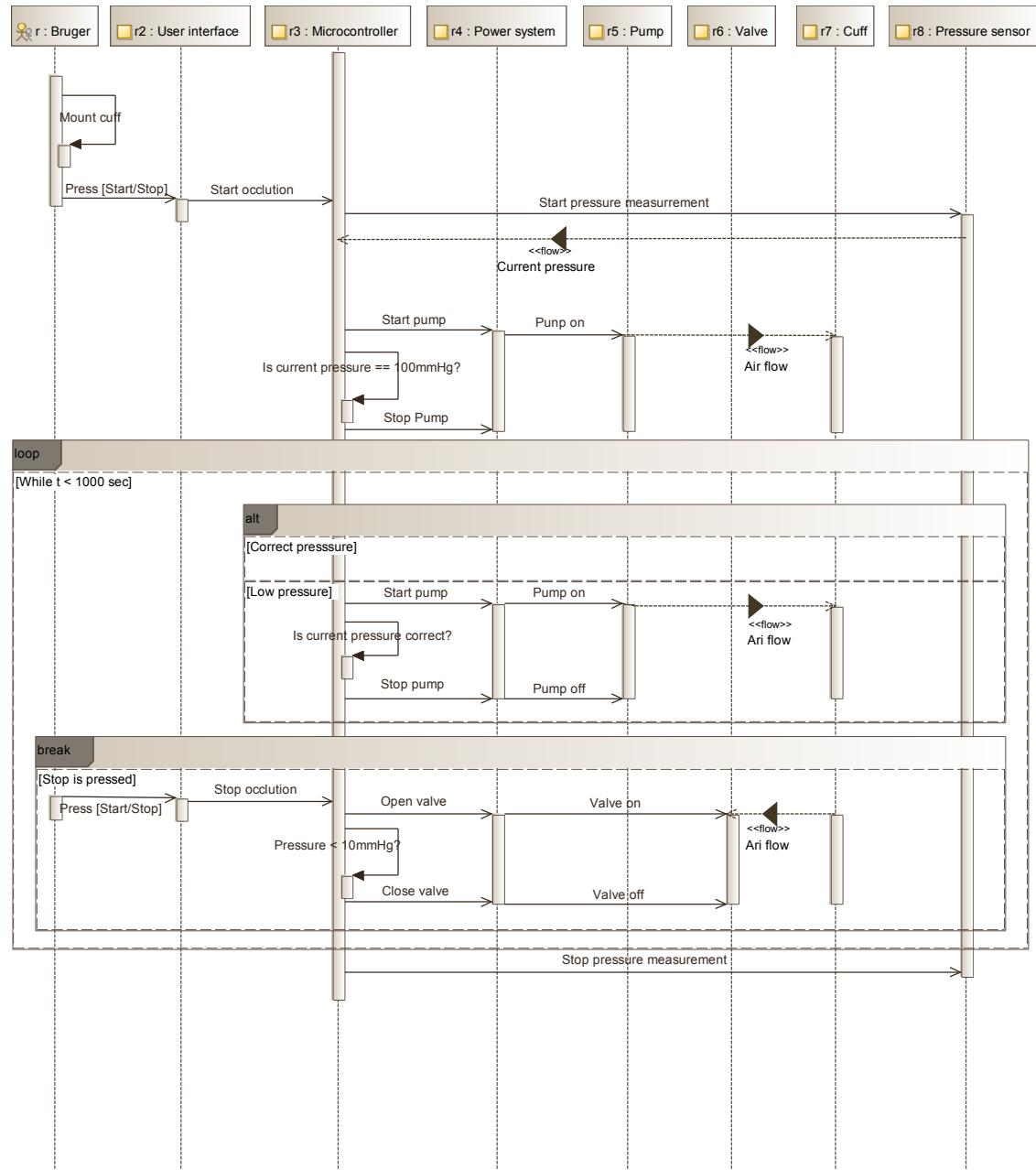


*Figur 11.12.* Sekvensdiagram over forløb i use case 4

### 11.3.6 Sikkerhedskontrol med pulsoximeter - UC5

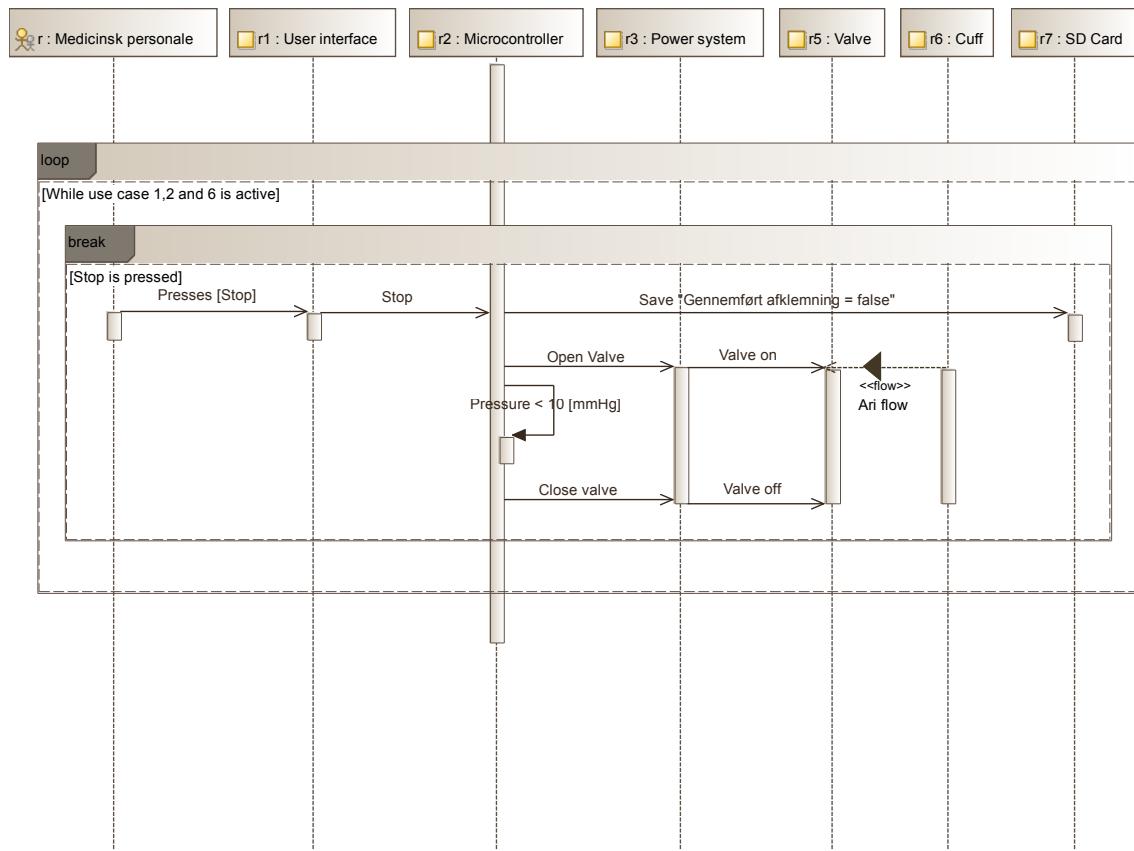
Denne use case og dets sekvens diagram er afgrænses pga. af problematikker med pulsoximetri som sikkerhedskontrol. For mere information se afsnittet omkring projektafgrænsninger i dokumentationsrapporten.

### 11.3.7 Okklusionstræning - UC6



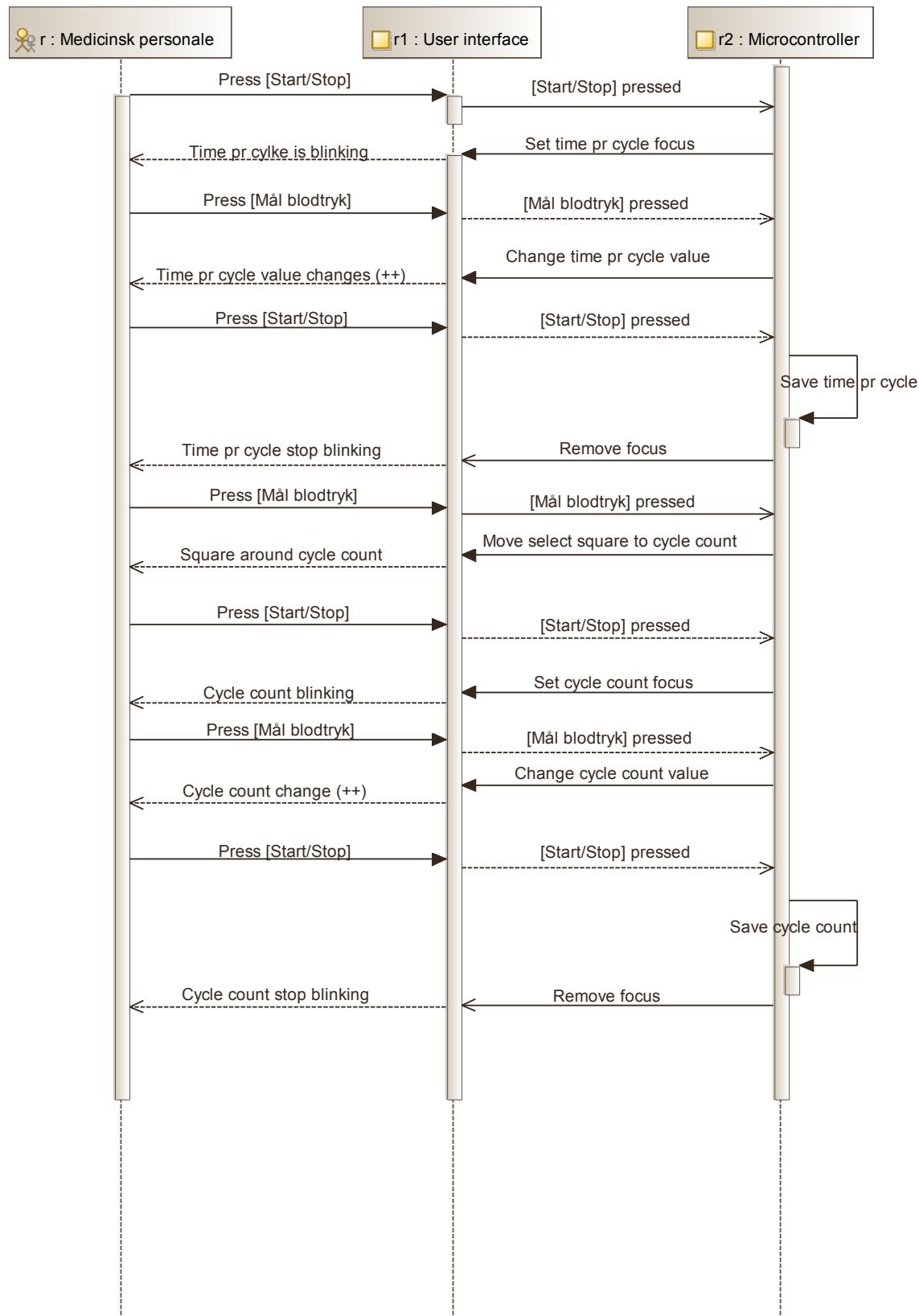
**Figur 11.13.** Sekvensdiagram over forløb i use case 6

### 11.3.8 Afbryd - UC7



**Figur 11.14.** Sekvens diagram over forløb i use case 7

### 11.3.9 Setup - UC8

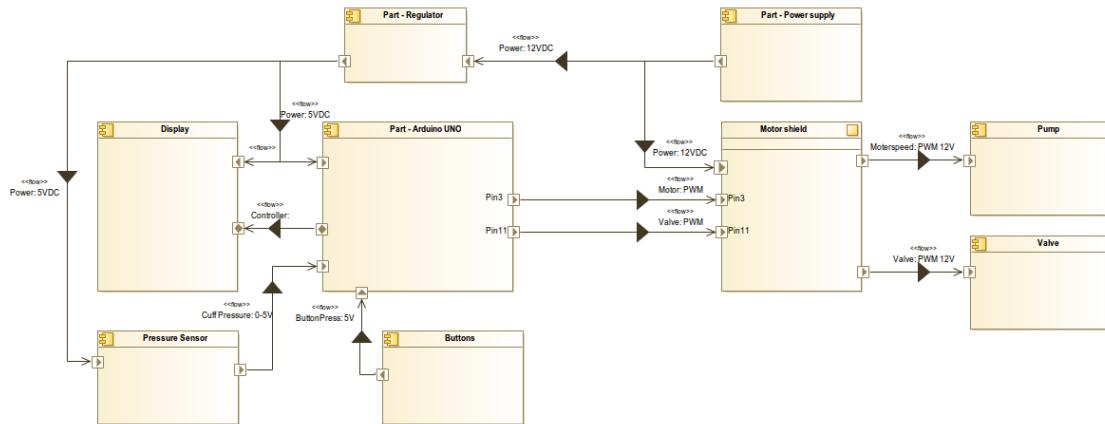


**Figur 11.15.** Sekvensdiagram over forløb i use case 8

## 11.4 Implementation

### 11.4.1 Hardware

Diagrammet viser interaktioner mellem systemets hardware dele



**Figur 11.16.** Internal block diagram over flow for hardwaren i *Konditioneringsapparatet*

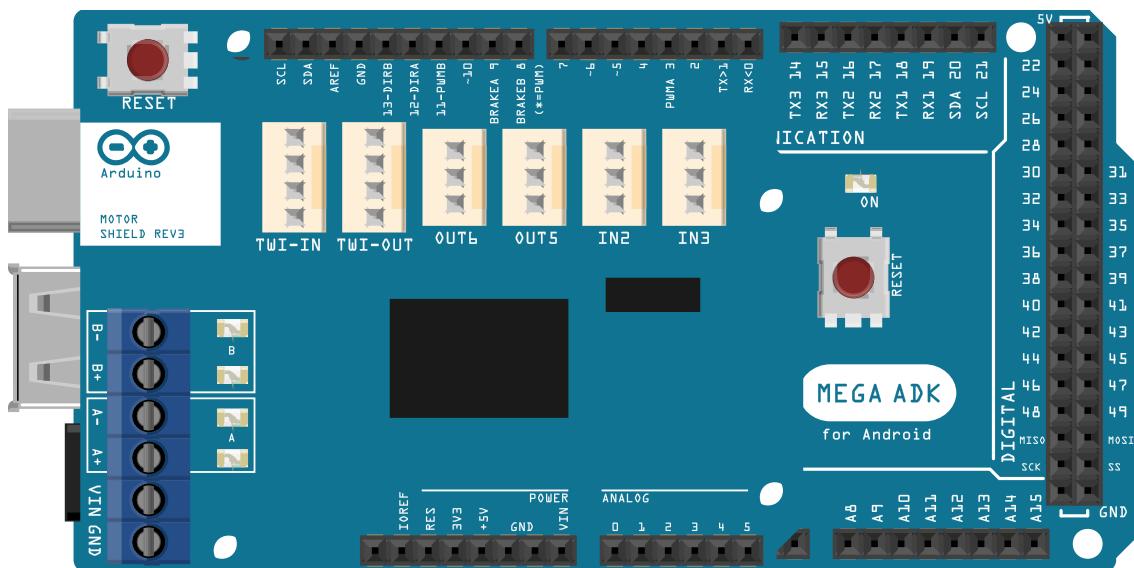
(Skal rettes til PDF)

### Beskrivelse af hardware

Se beskrivelser i kapitel 10

### Arduino Mega 2560 og Motor Shield

Til udvikling af prototypen bruges en Arduino Mega med en AtMega2560 processor og et 12V motor shield. Se oversigt tegning nedenfor



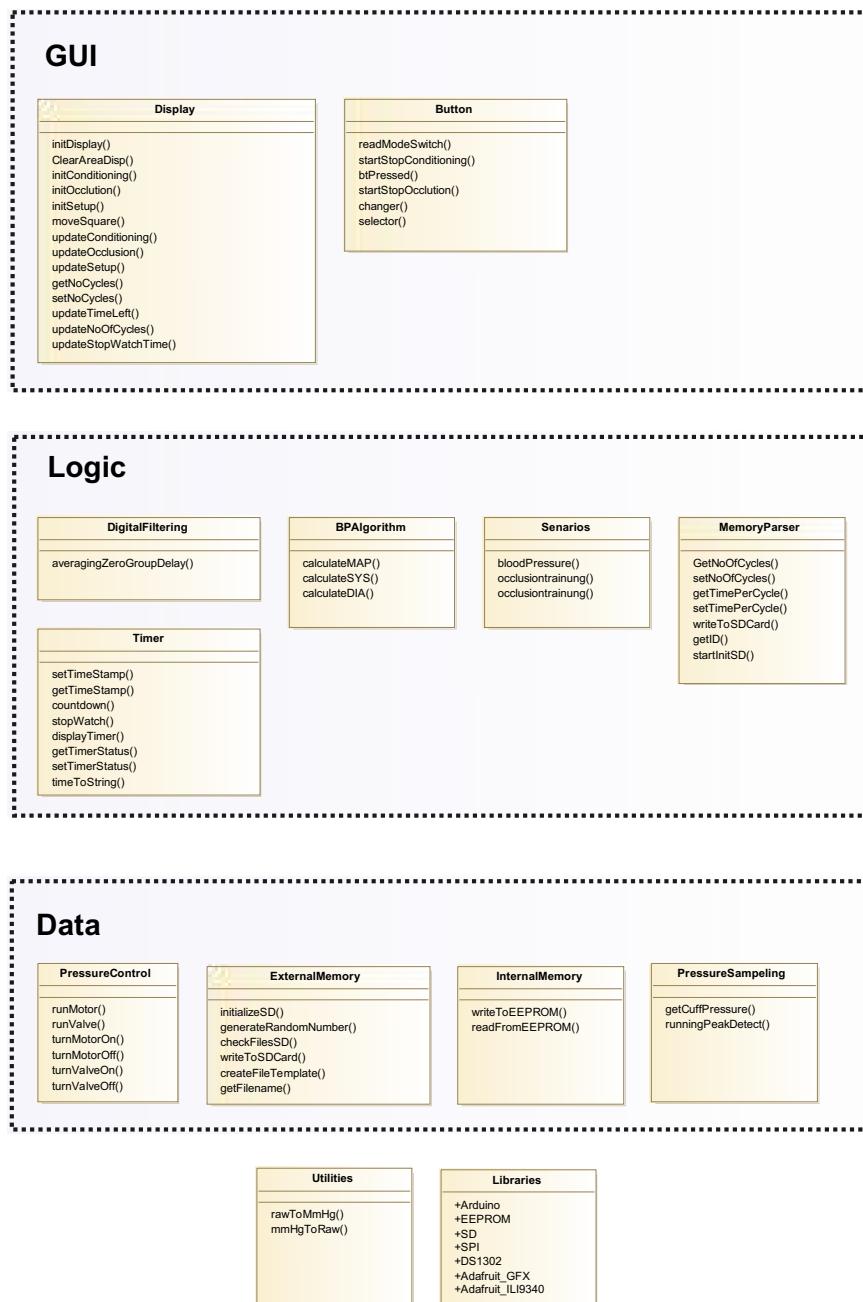
**Figur 11.17.** Arduino Mega med 12V motor shield monteret

### 11.4.2 Software

*Konditioneringsapparetet* består kun af ét software system og det kræver ikke flere software systemer for at apparatet fungere. Dette giver derfor et simpelt struktur for softvaren.

#### Klasse diagram

Diagrammet præsentere software klasser med funktioner og de er opdelt i hver sit namespace. Denne struktur er valgt for at opnå høj samhørighed og lav kobling.



**Figur 11.18.** Class diagram for softvaren i *Konditioneringsapparatet*

## Sprog

- **C++** - brugt til at skrive source code til arduino
- **SysML** - brugt til udvikling af system diagrammer

## 3-lags modellen

Softwareen er struktureret efter 3-lags princippet. Dette design er valgt for at give klare grænseflader og ansvarsfordeling. De tre lags er præsentationslaget, logik laget og datalaget. Lagdelingen giver stor fleksibilitet fordi det er nemmere at vedligeholde og genbruge kode fra et bestemt lag uden at den influere med andre lag. De tre lag er repræsenteret som hvert sit namespace i software arkitekturen

## Udviklingsværktøjer

### Eclipse

IDE til udvikling af C og C++. Dette miljø bruges til at skrive koden til arduinoen og dermed styringen af prototypen. Der er valgt versionen: "Juno Service Release". Denne version understøtter integration af Arduino' eget IDE, samtidig med at man kan gøre brug af Eclipses programmerings funktioner.

### Arduino IDE

Arduinos eget udviklingsmiljø bruges som et plugin via Eclipse. Den brugte version er 1.5.5. Grundet manglende funktion er det blevet fravalgt at bruge Arduino IDE alene.

### Git, GitHub Desktop og SmartGit

Til versionsstyring af til projekt dokumentation og source code. Bachelor gruppen har købt et privat repository grundede den igangværende patentsag. Som bruger interface for git er blevet brugt henholdsvis GitHub Desktop og SmartGit

## Eksterne biblioteker

Arduino biblioteker

- EEPROM - muliggøre operationer på arduinoen indbyggede hukommelse
- SD - muliggøre operationer med SD kort
- TFT - funktioner til at bruge tft displays til arduino

## 11.5 Deployment

Da produktet Konditioneringsapparat er en prototype beskæftiger projektet sig ikke med de redskaber der bruges i deployment view. Dette view beskriver blandt andet hvordan software mappes på hardware, kaldet et deployment diagram. Konditioneringsapparatet består kun af én software eksekverende enhed og derfor er dette overflødig at beskrive.



## **Del IV**

# **Implementeringsdokument**



# Indholdsfortegnelse

<b>Kapitel 12 Indledning</b>	<b>85</b>
12.1 Formål . . . . .	85
12.2 Projektreferencer . . . . .	85
12.3 Læsevejledning og dokumentstruktur . . . . .	85
12.4 Versionshistorik . . . . .	86
12.5 Definitioner og forkortelser . . . . .	86
<b>Kapitel 13 Software</b>	<b>87</b>
13.1 Klasse diagram . . . . .	87
13.2 Namespace: GUI Laget . . . . .	88
13.2.1 Klasse: Display . . . . .	88
13.2.2 Klasse: Buttons . . . . .	94
13.3 Namespace: Logik laget . . . . .	95
13.3.1 Klasse: BPalgorithm . . . . .	95
13.3.2 Klasse: DigitalFiltering . . . . .	96
13.3.3 Klasse: Scenarios . . . . .	97
13.3.4 Klasse: Timer . . . . .	98
13.3.5 Klasse: MemoryParser . . . . .	100
13.4 Namespace: Data laget . . . . .	101
13.4.1 Klasse: PressureControl . . . . .	102
13.4.2 Klasse: ExternalMemory . . . . .	102
13.4.3 Klasse: InternalMemory . . . . .	104
13.4.4 Klasse: PressureSampling . . . . .	104
13.5 Namespace: Global . . . . .	105

13.5.1 Klasse: Utilities . . . . .	106
13.5.2 Klasse: Konditioneringsapparat.pde (Main fil) . . . . .	106
<b>Kapitel 14 Filter design</b>	<b>109</b>
14.1 Analoge filtre . . . . .	109
14.2 Komponent udregninger . . . . .	110
14.2.1 Anden ordens butterworth filter . . . . .	110
14.2.2 Høj pas filter . . . . .	111
14.2.3 Lav pas filter . . . . .	111
14.2.4 Gain på manchet oscillationer . . . . .	113
14.2.5 Gain på manchettryk signal . . . . .	113
14.3 Praksis . . . . .	114
14.4 Digital filtrering . . . . .	118
14.4.1 Matlab . . . . .	119
<b>Kapitel 15 Hardware</b>	<b>121</b>
15.1 Schmetic . . . . .	121
15.2 Knapper . . . . .	122
15.3 Filter . . . . .	122
15.4 Timer . . . . .	123
15.5 Display . . . . .	124
15.6 Motor shield . . . . .	125

# 12 | Indledning

Implementeringsdokument giver et overblik over hvordan både hardware og software er blevet implementeret i udviklingen af prototypen. Dokumentet indeholder beskrivelse af strukturen af software klasser og deres funktionalitet. For hardware delen er der beskrevet de forskellige hardware blokke og hvordan de er blevet implementeret for at kunne leve op til kravene stillet i kravspecifikationen.

## 12.1 Formål

Dette dokument har til formål at give læseren et teknisk indblik i *Konditioneringsapparatets* funktionalitet og opbygning, samt skabe fuld forståelse for alle systemets under dele. Som en forlængelse af system arkitekturen, som beskrev for dette system skulle designes, beskriver dette dokument det færdig design og hvordan det har opnået sin funktionalitet

## 12.2 Projektreferencer

- Reference til kravspecifikation
- Reference til accepttest
- Reference til system arkitekturen
- Reference til software

## 12.3 Læsevejledning og dokumentstruktur

Da dette dokument er en del af udviklingsdokumentation, er det vigtigt at læse i sammenhæng med kravspecifikationen og systemarkitekturen. Undervejs i dokumentet vil der være referencer til kravspecifikationen og derfor er der stor samhørighed mellem disse to dokumenter. Dette dokument skal også ses som en forklaring på software implementeringen, og derfor passer navne og overskrifter i software beskrivelse overens med navne på metoder og klasser i softwaren. Desuden beskrives hver software metode ud fra tre punkter; paramter, returntype og beskrivelse. Parameter beskriver hvilke parameter og typen af disse parameter, som metode skal have. Returntypen fortæller hvilken værdi metode returnere og hvilken type det er. Beskrivelsen forklare funktionaliteten af metode, og hvordan denne funktionalitet er opnået. Nogle metoder indeholder også kode eksempler, når det anses som nødvendigt for at forstå metoden.

## 12.4 Versionshistorik

Versions nummer	Ændring	Dato og initialer
0.1	Oprettelse af implementeringsdokument og skabelon	01.11.15 KJS
0.2	Tilføjelse af filter dokumentation	11.11.15 SVG
0.3	Beskrivelse af display, timer og hukommelses klasser	12.11.15 KJS
0.4	Implementering klar til review, updatet filtrering og debouncing	13.11.15 KJS, SVG
0.5	Rettelser efter review	16.11.15 KJS, SVG
0.6	Ændret beskrivelse af metoder og ny graf over blodtryk	20.11.15 KJS, SVG

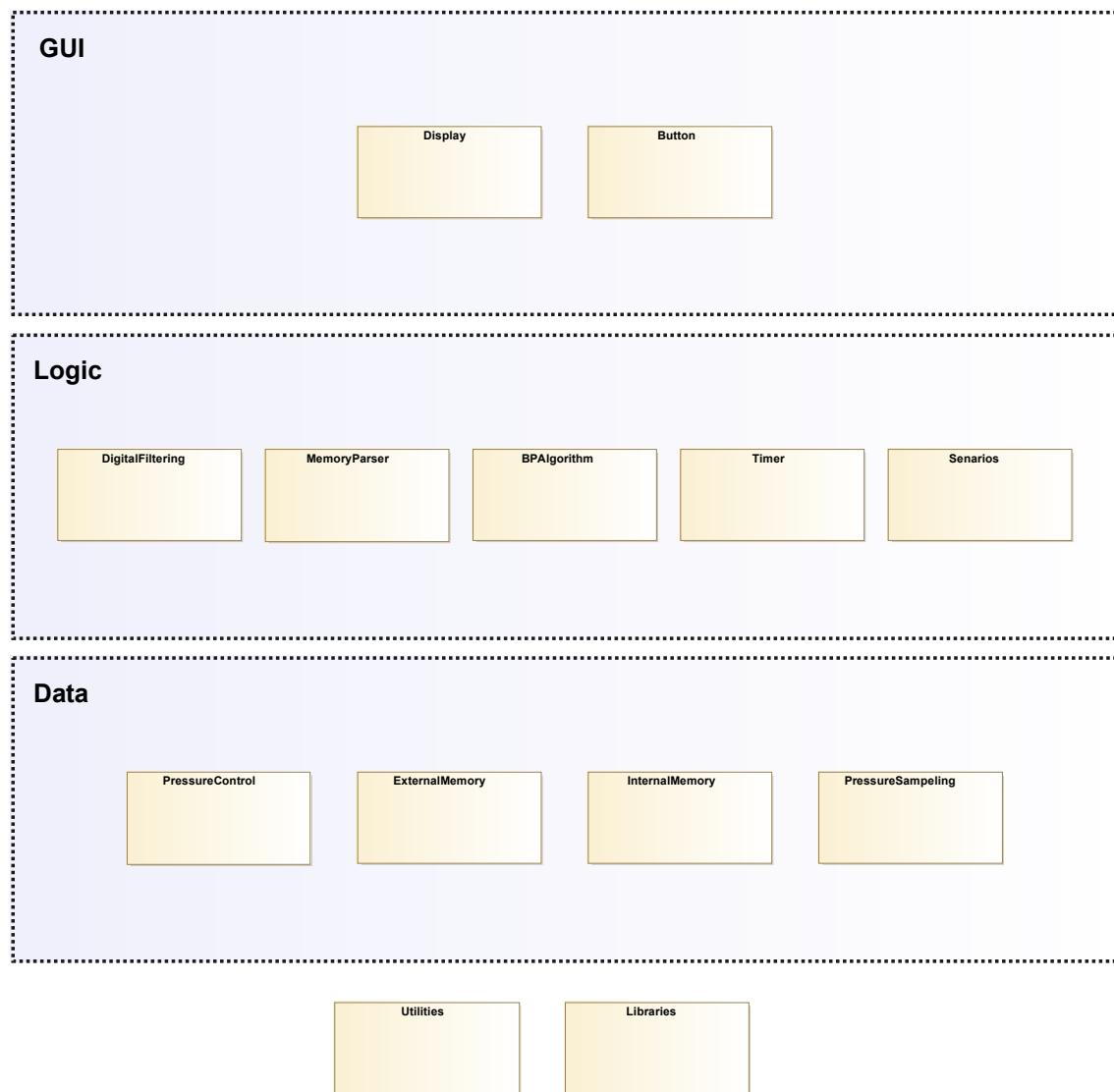
## 12.5 Definitioner og forkortelser

Udtryk / Forkortelse	Forklaring
Modeswitch	Knap til at styre hvilket program Konditioneringsapparatet skal køre
Tid pr cyklus	Variable som indeholder hvor mange sekunder et konditioneringscyklus skal vare
Antal cyklusser	Variable som indeholder hvor mange cyklusser et konditioneringsforløb skal vare
EEPROM	Intern flash hukommelse på arduino som der kan skrives og læses fra
Interrupt	Hver gang arduino kører en clock cyklus aflæses værdien af en række digital pins, hvorved man kan afbryde kode og kører en anden sekvens. Arduinoen har både interne og eksterne interrupt, men dette projekt gør kun brug af de eksterne interrupt
Sand/True, Falsk/False	Arduino genkender en sand/true når typen er bool og værdien er 1. Modsat registreres 0 som falsk/false

# 13 | Software

## 13.1 Klasse diagram

Oversigtsklassediagram, se figur 13.1, som fortæller strukturen af namespaces og klasse, men for overskueligheden er alle metoder undladt, se disse under deres respektive afsnit



*Figur 13.1.* Forsimplet klasse diagram

## 13.2 Namespace: GUI Laget



**Figur 13.2.** Klasse diagram over namespacet GUI

### 13.2.1 Klasse: Display

Denne klasse gør brug af to biblioteker for at kunne bruge TFT skærmens, henholdsvis Adafruit\_GFX og Adafruit\_ILI9340. For at kunne kommunikere med displayet gøres der brug af disse bibliotekters indbyggede funktioner. Derfor oprettes et objekt af klassen Adafruit\_ILI9340 kaldet TFTscreen.

#### Metode: initDisplay()

**Parameter:**

**Returtype:** void

**Beskrivelse:** Her initieres skærmens med funktion `.begin()`. Rotationen og baggrundsfarven af skærmens sættes også når denne metode kaldes. Skærmrotationen er sat 3, hvilket betyder at skærmens er i “landscape mode”.

#### Metode: clearAreaDisp()

**Parameter:** *unsigned short pointX, unsigned short pointY, unsigned short width, unsigned short height*

**Returtype:** void

**Beskrivelse:** Da skærmens baggrundsfarven er sat til sort, medtager denne metode 4 parameter hhv. start x-koordinat, start y-koordinat, bredde og højde. Disse parameter fortæller hvor en del af skærmens der skal farves sort, og dermed slette det område.

#### Metode: initConditioning()

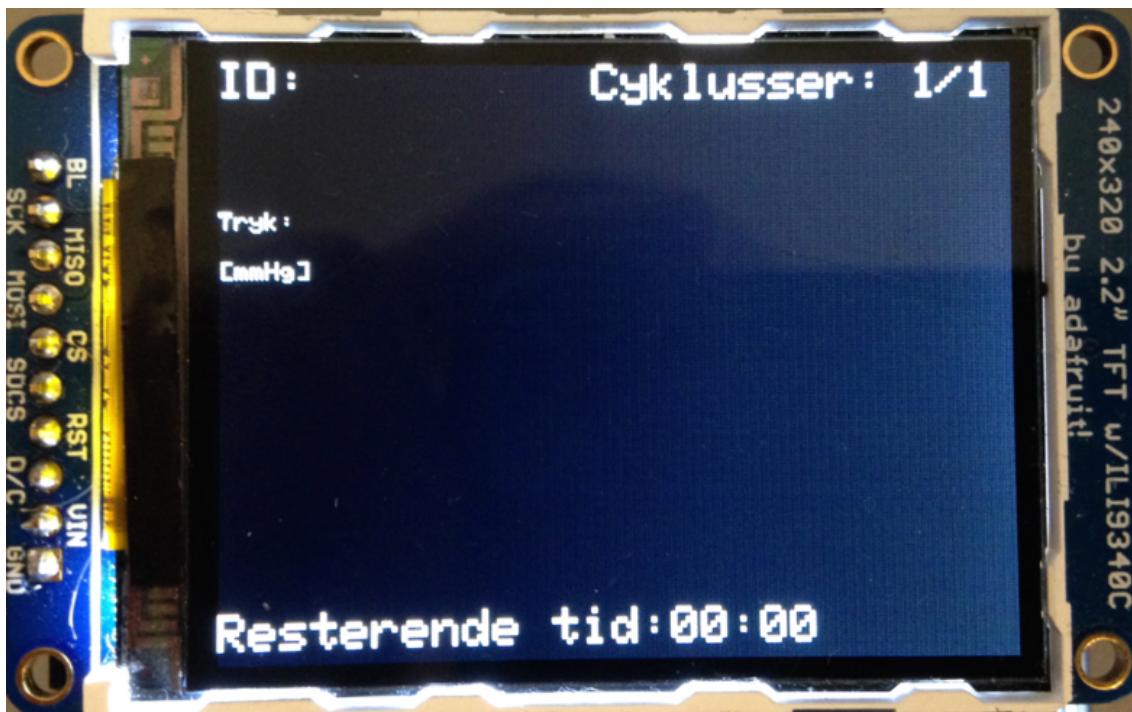
**Parameter:**

**Returtype:** void

**Beskrivelse:** Når denne metode kaldes skrives de “faste” værdier på skærmens til et konditioneringsforløb. På billedet nedenfor ses hvordan skærmens ser ud, når metoden er kørt. Et eksempel på hvordan der skrives tekst på skærmens:

```
1 TFTscreen.setTextColor(ILI9340_WHITE); TFTscreen.setTextSize(2);
2 TFTscreen.setCursor(0, 0);
3 TFTscreen.println("ID: ");
```

Først fortælles hvilken farve teksten skal have, dernæst tekststørrelse og placering. Til sidst angives hvilken tekst der skal printes på skærmen



*Figur 13.3.* Layout på displayet når initConditioning() bliver kaldt

**Metode:** initOcclusion()

**Parameter:**

**Returtype:** void

**Beskrivelse:** Denne metode bruges til at opsætte skærmen for okklusionstræningsforløb. Der skrives tid og enheden for tryk på skærmen. Se billedet nedenfor for layoutet.



Figur 13.4. Layout på displayet når initOcclusion() bliver kaldt

Metode: initSetup()

Parameter:

Returtype: void

Beskrivelse: Her opsættes skærmen for setup programmet. Teksten "Tid pr cyklus" og "Antal cyklusser" er faste værdi på skærmen. Men værdierne hentes fra logik laget, så de er opdateret.



*Figur 13.5.* Layout på displayet når initSetup() bliver kaldt

#### Metode: moveSquare()

**Parameter:** *unsigned short startX, unsigned short startY, unsigned short endX, unsigned short endY, unsigned short width, unsigned short height*

**Returtype:** *void*

**Beskrivelse:** Denne metode bruges til at flytte cursoren på skærmen under setup. For at slette noget på skærmen skal det farves samme farve som baggrunden. Derfor får metoden x- og y-koordinaterne for den firkant der skal slettes, samt x- og y-koordinaterne for hvor den nye firkant skal tegnes henne. Desuden skal metode også have bredde og højde på firkanten. For at sikre at der ikke kan laves et interrupt inde i metode, gør metode brug af den indbyggede funktion *noInterrupt()*. Et interrupt på det forkerte tidspunkt ville betyde at skærm ikke ville slette den forrige firkant eller ikke ville tegne det nye.

#### Metode: updateConditioning()

**Parameter:** *volatile bool \*buttonPressed, volatile bool \*btPressed*

**Returtype:** *void*

**Beskrivelse:** Metoden modtager to pointers, som peger på værdien af *buttonPressed* og *btPressen*. Derfor kan metoden i princippet være i 4 forskellige stadier (Se sandhedstabel 13.1 )

Parameter/Stadie	Stoppet	Konditioneringsforløb	Blodtryksmåling	Ingenting
buttenPressed	Falsk	Sand	Falsk	Sand
btPressed	Falsk	Falsk	Sand	Sand

*Tabel 13.1.* Sandhedstable over *updateConditioning()*s parametre

Nedenfor er et rammerne for strukturen i metoden, hvor *stadie* stemmer overens med stadierne fra sandhedstabellen.

```

1 //***Stadie: Blodtryksmaaling***
2 if(*btPressed && !buttonPressed)
3 {
4     //Measure blood pressure and save value to SD card
5 }
6 //***Stadie: Konditioneringsforløb***
7 if(!*btPressed && *buttonPressed && getNoCycleLeft() != 0)
8 {
9     //Measure blood pressure and save value to SD card
10    //Inflate the cuff to systolic pressure + 25mmHg
11
12    //Run conditioning in while loop
13 }
14 //***Stadie: Stoppet***
15 if(!*btPressed && !buttonPressed){
16     //Empty the cuff and clear the sensor value on the dispaly
17     //Reset the number of cycle run in conditioning
18 }
```

Metoden håndtere ikke hvis både *\*buttonPressed* og *btPressed* er trykket samtidigt  
*buttonPressed* styres ved knaptryk og bruger kan derfor starte og stoppe konditionerings  
 forløbet på denne måde. Hvis forløbet stopper af sig selv, altså hvis antallet af  
 tilbageværende cyklusser er nul, så håndterer metoden at brugeren ikke skal trykke to  
 gange på knappen for at starte et nyt forløb.

```

1 if(memory.getNoOfCycles() == 0)
2     *buttonPressed = false;
```

### Metode: updateOcclusion()

**Parameter:** volatile bool *\*buttonPressed*

**Returtype:** void

**Beskrivelse:** Denne metode køres når apparatet er sat på okklusions træningsforløb.  
 Denne metode bruger også pointeren til *buttonPressed*, hvis den er sand eksekveres et  
 while loop hvor der startes et stopur og trykket fra manchetten vises på skærmen. Hvis  
 værdien er falsk slettes sensorværdien på displayet, og slut tiden vises fra stopuret.

### Metode: updateSetup()

**Parameter:** volatile bool *\*state*

**Returtype:** void

**Beskrivelse:** Til styring af cursoren på displayet i setup. Denne metode består af en  
 switch case struktur, som har fire cases og casevalg afgøres af værdien af *\*state*. Case 0  
 og 1 gør brug af metoden *moveSquare(..)* som flytter cursoren. Case 2 og 3 sørge for at  
 vise værdien af hhv. tid pr cyklus og antal cyklusser. Da cursoren styres med interrupt er  
 interrupts slået fra så længe koden afvikles inde i case 2 og 3.

### Metode: getNoCycles()

**Parameter:**

**Returtype:** unsigned short

**Beskrivelse:** Bruges til at hente *antal cyklusser* fra logik laget. Denne værdi aflæses fra EEPROM via data laget, men for at overholde 3-lags modellen skal kommunikation gå via logik laget.

#### Metode: setNoCycles()

**Parameter:** *unsigned short value*

**Returtype:** *void*

**Beskrivelse:** Bruges til at sætte værdien af *antal cyklusser*.

#### Metode: updateTimeLeft()

**Parameter:** *unsigned short value*

**Returtype:** *void*

**Beskrivelse:** Denne metode er lavet til opdatere tiden under konditioneringsforløbet. Da det kræver fem kald af metoder fra biblioteket *Adafruit\_ILI9340* for at skrive tekst på skærmen er dette indkapslet i én metode, som blot skal have en String value. Metoden sørge også for at tiden kun opdateres når sekund tælleren på uret ændres

#### Metode: updateNumberOfCycles()

**Parameter:** *String value*

**Returtype:** *void*

**Beskrivelse:** Når metoden kaldes opdateres antallet cyklusser på skærmen. Da værdien som metoden modtager indeholder to gange så mange cyklusser som der skal foretages, håndtere metoder at der kun hvis det eksakte antal. Grunden til at der regnes med to gange så mange cyklusser skyldes at programmet intern skal håndtere en okklusion fase og en reperfusions fase for hver cyklus. Håndtering foregår på følgende måde:

```

1   if(value%2)
2     valToDisplay = (valToDisplay+1)/2;
3   else
4     valToDisplay = valToDisplay/2;
```

#### Metode: updateStopWatchTime()

**Parameter:** *unsigned short minutes, unsigned short seconds*

**Returtype:** *void*

**Beskrivelse:** Denne metode opdatere tiden fra stopuret på displayet under okklusions træningsforløbet. Displayet opdateres kun når antallet af sekunder på stopuret ændres.

#### Metode: updateBloodPressure

**Parameter:** *unsigned short sys, unsigned short dia, unsigned short map)*

**Returtype:** *void*

**Beskrivelse:** Metoden modtager tre parameter med systolisk, diastolisk og middel arterie trykket. Disse værdi konverteres til en String og skrives på skærmen i følgende format: "120/80(93)"

### 13.2.2 Klasse: Buttons

**Metode:** `readModeSwitch()`

**Parameter:**

**Returtype:** *unsigned short*

**Beskrivelse:** Denne metode læser to analoge pins hhv; A8 og A9. Her læses stadiet af modeswitch knappen. Hvis A8 er lav(0V) og A9 er høj(5V) køres konditionering, hvis både A8 og A9 er lave køres okklusionstræning og til sidst hvis A8 er høj og A9 er lav køres setup. Denne metode bliver kaldt én gang når arduinoen startes op. Skal der ændres på hvilken forløb apparatet skal køre, skal arduinoen genstartes.

**Metode:** `startStopConditioning()`

**Parameter:** *volatile bool startButtonPressed*

**Returtype:** *bool*

**Beskrivelse:** Styring af værdien for *startButtonPressed*. Hver gang metoden køres inveteres værdien af *startButtonPressed*. Desuden indeholder metoden et *if/else* statement, der hvis værdien af *startButtonPressed* er falsk sletter den gamle måling på skærmen og nulstiller antallet af kørte cyklusser. Hvis værdien er sand sættes et tidsstempel, så timeren passer når et nyt forløb startes.

**Metode:** `btPressure()`

**Parameter:** *volatile bool btPressed*

**Returtype:** *bool*

**Beskrivelse:** Styring af værdien for *btPressed*. Når denne metode kaldes inveteres værdien af *btPressed*. Hvis værdien er falsk, slettes den sidste måling på skærmen.

**Metode:** `startStopOcclusion()`

**Parameter:** *volatile bool startButtonPressed*

**Returtype:** *void*

**Beskrivelse:** Her inveteres værdien af *startButtonPressed* og returneres. Hvis denne værdi er falsk kaldes en metode fra display klassen som fjerner en værdi på skærmen og der sættes et tidsstempel, fordi at værdien af *startButtonPressed* går fra falsk til sand og derfor skal der startes et okklusionstræningsforløb

**Metode:** `changer()`

**Parameter:** *volatile unsigned short state*

**Returtype:** *unsigned short*

**Beskrivelse:** Denne metode bruges til at styre cursoren når der skal ændres i antallet af cyklusser og tiden pr. cyklus. Den modtager værdien state, som kan være et tal mellem nul og tre. Hvis state nul ændres værdien til en og omvendt, det betyder at den skifter mellem at pege på *antal cyklusser* og *tid pr cyklus*. Hvis state har værdien to ændres der på *tid pr. cyklus* og hver gang metoden kaldes forøges værdien tid pr. cyklus med 30 sekunder. Desuden håndterer metoden at denne værdi kun kan ændres i intervallet mellem 180 til 480 sekunder, dvs 3 til 5 minutter. Hver gang værdien *tid pr cyklus* ændres, skrives den nye

værdi til EEPROM via metoden *InternalMemory::setTimePerCycles()*. Når state er lige med 3, ændres værdien af antal cyklusser og denne værdi forøges med 1 cyklus hver gang knappen trykkes. Denne værdien kan ændres i intervallet mellem 1 og 5 cyklusser. Den nye værdi skrives til EEPROM.

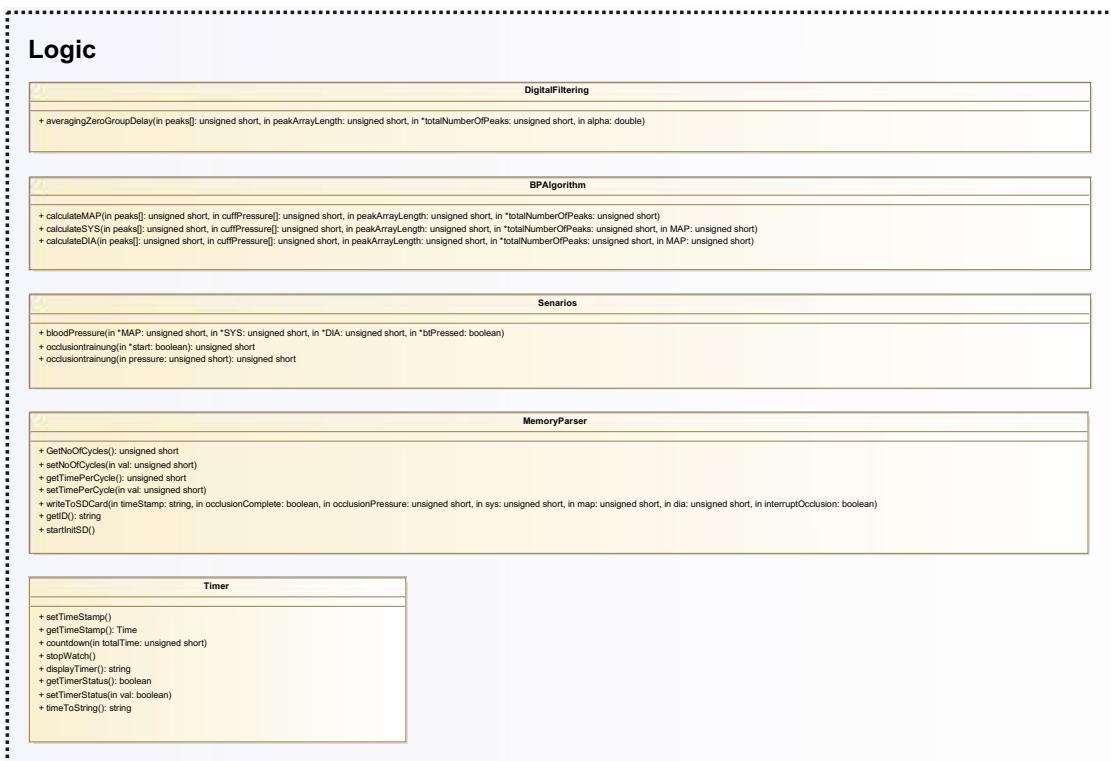
#### Metode: selector()

Parameter: *volatile unsigned short state*

Returtype: *unsigned short*

Beskrivelse: Metoden *selector()* ændres udelukkende på værdi af *state* og returnerer den nye værdi af *state*.

### 13.3 Namespace: Logik laget



*Figur 13.6.* Klasse diagram over namespacet Logic

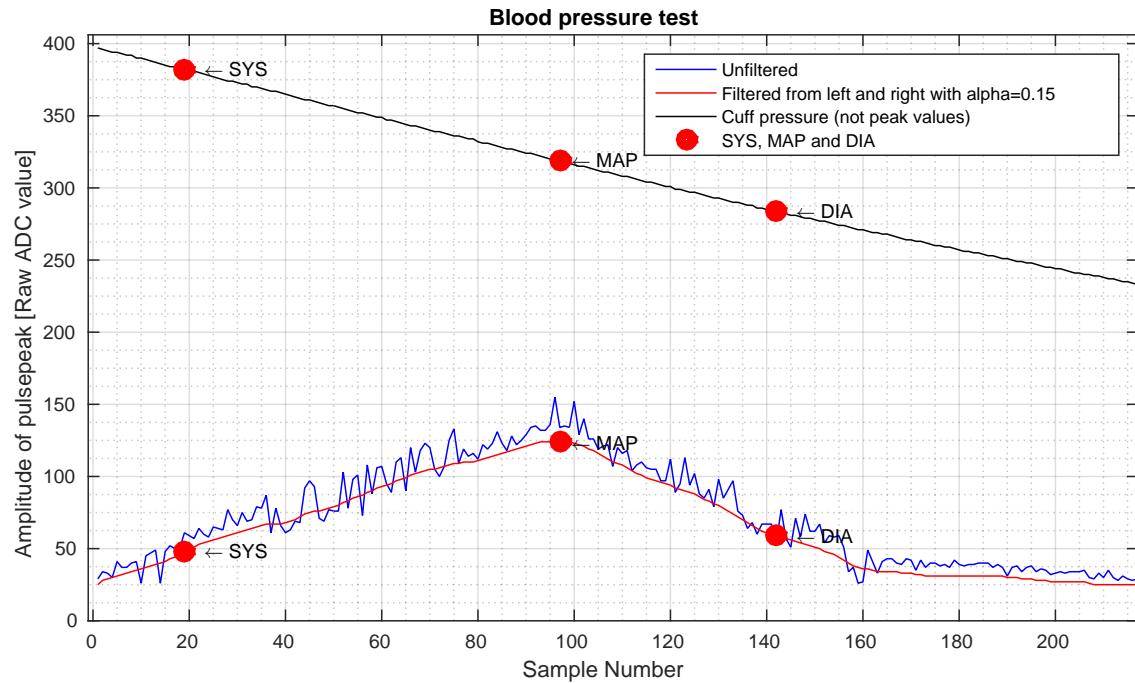
#### 13.3.1 Klasse: BPalgorithm

##### Metode: calculateMap()

Parameter: *unsigned short peaks[], unsigned short cuffPressure[], unsigned short peakArrayLength, unsigned short \*totalNumberOfPeaks*

Returtype: *unsigned short*

Beskrivelse: Denne metode beregner MAP ud fra digitalt filtreret peakdata i *peaks[]* og *cuffPressure[]*. MAP findes som trykket i manchetten ved den højeste peak amplitude (se figur 13.7)



**Figur 13.7.** Graf med data fra en blodtryksmåling på simulator(120/80)

Her ses at det rå signal er støjfyldt. Sort er manchet trykket, blå er de rå peak amplituder, rød er filtreret en gang fra venstre mod højre, sort er filtreret fra begge sider.

#### Metode: calculateSYS()

**Parameter:** *unsigned short peaks[], unsigned short cuffPressure[], unsigned short peakArrayLength, unsigned short \*totalNumberOfPeaks, unsigned short MAP*

**Returtype:** *unsigned short*

**Beskrivelse:** Denne metode beregner SYS ud fra MAP og digitalt filtreret peakdata i *peaks[]* og *cuffPressure[]*. SYS findes som trykket i manchetten ved peak amplituder på 38% af MAP. (Se figur 13.7)

#### Metode: calculateDIA()

**Parameter:** *unsigned short peaks[], unsigned short cuffPressure[], unsigned short peakArrayLength, unsigned short \*totalNumberOfPeaks, unsigned short MAP*

**Returtype:** *unsigned short*

**Beskrivelse:** Denne metode beregner DIA ud fra MAP og digitalt filtreret peakdata i *peaks[]* og *cuffPressure[]*. DIA findes som trykket i manchetten ved peak amplituder på 48% af MAP. (Se figur 13.7)

### 13.3.2 Klasse: DigitalFiltering

#### Metode: averagingZeroGroupDelay()

**Parameter:** *nsinged short peaks[], unsigned short peakArrayLength, unsigned short \*totalNumberOfPeaks, double alpha*

**Returtype:** *void*

**Beskrivelse:** Denne metode anvender eksponentiel midligsfilter teknik (Se afsnit 14.4) uden group delay til at midle over parameteren peaks.

```

1     peaks[0] = startValue;
2     peaks[totalNOPeaks] = startValue;
3     for(i = 1;i<totalNOPeaks; i++)
4     {
5         peaks[i] = alpha*peaks[i]+(1-alpha)*peaks[i-1];
6     }
7
8     for(i = totalNOPeaks-1;i>0; i--)
9     {
10        peaks[i] = alpha*peaks[i]+(1-alpha)*peaks[i+1];
11    }

```

### 13.3.3 Klasse: Scenarios

**Metode:** `bloodPressure()`

**Parameter:** `unsigned short *MAP, unsigned short *SYS, unsigned short *DIA, BPAAlgorithm bpa, Data::PressureControl pc, Data::PressureSampling ps, Logic::DigitalFiltering df, Utilities util`

**Returtype:** `void`

**Beskrivelse:** Denne metode indeholder opskriften til en blodtryksmåling. Det vil sige at kaldes metoden, udføres en blodtryksmåling og alle andre klasser og metoder, som skal bruges til dette eksekveres inde i denne metode. Pointerne til de tre variabler får værdierne MAP, SYS og DIA fra blodtryksmålingen.

**Metode:** `occlusiontraining()`

**Parameter:** `volatile bool *start` **Returtype:** `unsigned` **Beskrivelse:** Denne metode metoder en bool værdi, som afgøre hvilke to stadie metoden skal eksekveres i. Hvis `start` er true, lukkes ventil og med en if sætning pumpes manchetten op til mimumum 90 mmHg og hvis trykket overstiger 100 mmHg slukkes motoren. Hvis metoden modtager en falsk værdi slukkes motoren og ventilen åbnes.

**Metode:** `occlude()`

**Parameter:** `unsigned short pressure` **Returtype:** `unsigned short`

**Beskrivelse:** Metode der bruges i konditioneringsforløbet når blodtrykket er bestemt og der skal laves en afklemning. Først indhentes det nuværende manchet tryk. Dernæst bruges den parameteren `pressure`, som indeholder det systoliske blodtryk, til at bestemme hvor meget manchetten skal fyldes. Der søges for at trykket i manchetten mindst bliver 200mmHg og maks 300mmHg. Efter dette indeholder `pressure` afklemningstrykket og motoren begynder at pumpe indtil trykket i manchetten er større end `pressure + 10`. Plus 10 fordi at motoren drifter en anelse og ikke stopper præcist ved det tryk der specificeres. Hvis metoden modtager 0 istedet for det systoliske tryk, åbnes ventilen og manchetten tømmes indtil trykket er under 10mmHg.

### 13.3.4 Klasse: Timer

Klassen timer gør brug af en hardware Real Time Clock(RTC) med IC'en *DS1302*. For at kommunikere med denne RTC, så gør klassen brug af et biblioteket *DS1302*. Derfor oprettes et objekt af klassen *DS1302.h* ved navn timestamp. Et Time objektet indeholder hhv: år, måned, dag, time, minut, sekund og ugedag.

#### Metode: `setTimeStamp()`

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Denne metode aflæser den nuværende værdi af timeren og sætter en variable til denne værdi.

#### Metode: `getTimeStamp()`

**Parameter:**

**Returtype:** *Time*

**Beskrivelse:** Returnerer et Time objekt med værdien af timestamp.

#### Metode: `countdown()`

**Parameter:** *unsigned short totalTime*

**Returtype:** *void*

**Beskrivelse:** Denne metode modtager parameteren *totalTime*, som indeholder det ønskede antal sekunder nedtællingen skal vare. Variablen *elapsedTime* indeholder den nuværende tid og timestamp indeholder et tidsstempel der bliver sat når timeren skal starte. Hver gang metoden køres trækkes den nuværende tid i hhv timer, minutter og sekunder fra hinanden. Dernæst omregnes de forskellige difference til samlede antal sekunder, hvorefter det tal omregnes til sekunder og minutter. For at få antal sekunder tages differencen mellem *totalTime* og *elapsedTotalSeconds* udregner modulus 60 til dette tal (Se formel 13.1)

$$\text{seconds} = (\text{totalTime} - \text{elapsedTotalSeconds}) \bmod 60 \quad (13.1)$$

Dette samme gøres for minutter blot hvor *seconds* også trækkes fra. Se kode nedenfor.

```

1   timerHasEnded = false;
2   Time elapsedTime = rtc.time();
3   String elapsedTimeString;
4   unsigned short hoursToSec = (elapsedTime.hr - timestamp.hr) * 24 * 60;
5   unsigned short minutesToSec = (elapsedTime.min - timestamp.min) * 60;
6   unsigned short elapsedTotalSeconds = hoursToSec + minutesToSec + (elapsedTime.sec -
7       timestamp.sec);
8   seconds = (totalTime - elapsedTotalSeconds) % 60;
9   minutes = (totalTime - elapsedTotalSeconds - seconds)/60;
10
11  if(minutes == 0 && seconds == 0)
12      timerHasEnded = true;
```

Når det er regnet ud hvor mange minutter og sekunder der er tilbage i nedtællingen, gemmes de lokale variable minutes og seconds.

**Metode: stopWatch()****Parameter:****Returtype:** *void*

**Beskrivelse:** Metode til at styre simulere og styre et stopur under okklusionstræning, den forløbne tid udregnes på samme måde *countdown()*, blot hvor det er tidsstemplet der trækkes fra den nuværende tid. Denne metode gemmer også minutter og sekunder i to lokale variabler, når udregning er den forløbne tid er færdig.

**Metode: displayTimer()****Parameter:****Returtype:** *String*

**Beskrivelse:** Denne metode bruges til at konvertere tiden fra enten stopuret eller nedtællingen til formatet mm:ss. Desuden konverteres tiden til en string.

```

1  String minString = String(minutes, DEC);
2  String secString = String(seconds, DEC);
3  String timeString;
4
5  if(0 <=minutes && minutes < 10)
6      minString = String("0" + minString);
7  else
8      minString = String(minutes, DEC);
9
10 if(0 <=seconds && seconds < 10)
11     secString = String("0" + secString);
12 else
13     secString = String(seconds, DEC);
14 return timeString = String(minString + ":" + secString);

```

For at sikre at tiden vises på formatet mm:ss, tjekker metoden for om *seconds* eller *minutes* er større eller lig med 0 og mindre end 10. Hvis det er tilfældet tilføjes et nul foran værdien

**Metode: getTimerStatus()****Parameter:****Returtype:** *bool*

**Beskrivelse:** Metode til at returnere værdien af statussen for timeren, hvis værdien er sand er timeren slut og omvendt hvis værdien er falsk kan timeren stadig være igangværende.

**Metode: setTimerStatus()****Parameter:** *bool val***Returtype:** *void*

**Beskrivelse:** Denne metode bruges til at sætte værdien af statussen for timeren, den sættes til true for at stoppe timeren. Derfor modtager en metode en parameter af typen bool.

**Metode: timeToString()****Parameter:****Returtype:** *String*

**Beskrivelse:** For at kunne gemme tidsstempler på SD kortet, er denne metode lavet til at konvertere et tidsstempel til en string på formatet: *tt:mm:ss DD-MM-YY*. Ligesom metoden *displayTimer()* håndterer denne metode hvis enten timer, minutter eller sekunder er mindre end 10 og sætter et nul foran. Metoden returnerer en samlede string med et tidsstempel

### 13.3.5 Klasse: MemoryParser

Denne klasse er lavet for at overholde 3-lags modellen. Da informations læsning fra fx EEPROM og SD kort skal foregå i data laget, skal der en række metoder til at sende information igennem logik laget og videre til GUI laget. Derfor indeholder denne klasse som udgangspunkt kun get og set metoder.

#### Metode: getNumberOfCycles()

**Parameter:**

**Returtype:** *unsigned short*

**Beskrivelse:** Denne metode returnerer værdien fra data laget via metoden “*intMem.readFromEEPROM()*”.

#### Metode: setNumberOfCycles()

**Parameter:** *unsigned short val*

**Returtype:** *void*

**Beskrivelse:** Metode der skriver til data laget via metoden: *intMem.writeToEEPROM(200, val)*. Parameteren *val* videres til med denne metode.

#### Metode: getTimePerCycle()

**Parameter:**

**Returtype:** *unsigned short*

**Beskrivelse:** Da værdien af timePerCycle indeholder hvor mange sekunder én konditioneringscyklus skal vare, er denne værdi ofte større end 255. Denne værdi skal læses fra EEPROM og en plads kan indeholde værdier på maks 255. Derfor sørger metoden for at hente værdien fra den anden plads, hvis den er større end maks værdien.

```

1     unsigned short val = intMem.readFromEEPROM(205);
2     unsigned overloadVal = 0;
3     if(val == 255)
4         return overloadVal = val + intMem.readFromEEPROM(206);
5     else
6         return val;

```

#### Metode: setTimePerCycle()

**Parameter:** *unsigned short val*

**Returtype:** *void*

**Beskrivelse:** Beskrivelse: Når denne metode køres skrives tiden pr. cyklus til EEPROM, som forklaring i metoden *getTimePerCycle()*, skal den metode håndtere overload.

```

1     if(val > 255){
2         unsigned short rest = val % 255;
3         unsigned short valtoFit = val - rest;

```

```

4     intMem.writeToEEPROM(205, valtoFit);
5     intMem.writeToEEPROM(206, rest);
6   }
7   else
8     intMem.writeToEEPROM(205, val);

```

Det er forinden bestemt at adresserne 205 og 206 bruges til at gemme *TimePerCycle*. Metoden får en parameter som indeholder antallet af sekunder en cyklus skal vare, og for at sørge for værdien skrives korrekt til EEPROM udregnes modulus 255 af antallet af sekunder og den rest gemmes på adressen 206.

*String timeStamp, boolean occlusionComplete, unsigned short occlusionPressure, unsigned short sys, unsigned short map, unsigned short dia, boolean interruptOcclusion*

#### Metode: writeToSDCard()

**Parameter:** *String timeStamp, boolean occlusionComplete, unsigned short occlusionPressure, unsigned short sys, unsigned short map, unsigned short dia, boolean interruptOcclusion*

**Returtype:** *void*

**Beskrivelse:** Denne metode indeholder syv parametre af typerne *String, boolean* og *unsigned short*. Metoden konverterer og samler alle parametrene til en string og ved denne sendes videre til datalaget.

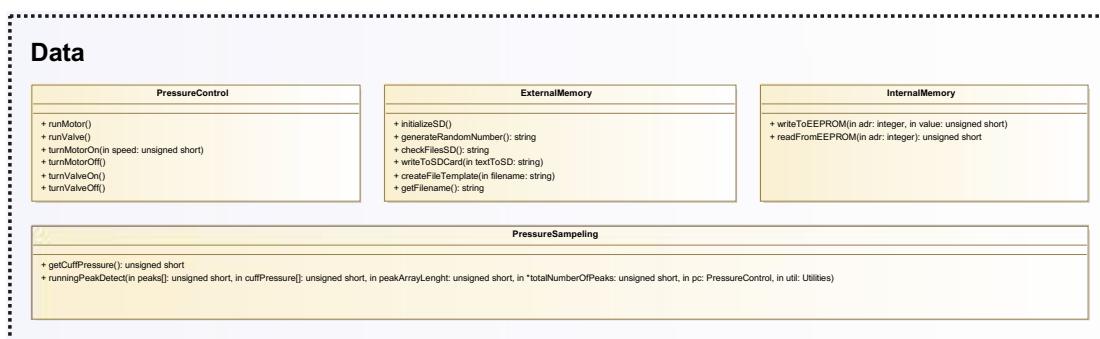
#### Metode: getID()

**Parameter:** **Returtype:** *String* **Beskrivelse:** Metode som henter filnavnet fra datalaget, og fjerner endelsen ".csv" og returner den nye String.

#### Metode: startInitSD()

**Parameter:** **Returtype:** *void* **Beskrivelse:** Metode der kalder *initializeSD()* fra datalaget.

## 13.4 Namespace: Data laget



*Figur 13.8.* Klasse diagram over namespaceset data

### 13.4.1 Klasse: PressureControl

**Metode:** runMotor()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Funktionen starter motoren og lader den kører ved 100% duty cycle indtil interrupt knappen på pin 18 ikke længere leverer en høj.

**Metode:** runValve()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Funktionen åbner for ventilen og lader den kører ved 100% duty cycle indtil interrupt knappen på pin 19 ikke længere leverer en høj.

**Metode:** turnMotorOn()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Funktionen tænder for motoren med den hastighed, angivet i parameteren (0-255).

**Metode:** turnMotorOff()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Funktionen stopper for motoren ved at sætte pin 3 lav

**Metode:** turnValveOn()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Funktionen åbner for ventilen ved at sætte pin 11 høj

**Metode:** turnValveOff()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Funktionen lukker for ventilen ved at sætte pin 11 lav

### 13.4.2 Klasse: ExternalMemory

Denne klasse gør brug af to arduino biblioteker hhv. SD og SPI. De to biblioteker muliggøre kommunikation med SD kortet via en SPI forbindelse.

**Metode:** initializeSDCard()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Simple metode der starter kommunikation med SD kortet og køre metoden *checkFilesSD()* (Se 13.4.2)

### Metode: generateRandomNumber()

**Parameter:**

**Returtype:** *String*

**Beskrivelse:** Denne metode genererer et 6-cifret tilfældigt nummer. Arduinos indbyggede funktion *random()* laver et tilfældigt nummer i det interval man specificerer, men den genererer dem i samme rækkefølge hver gang. For at gøre nummeret “mere tilfældigt” styres rækkefølgen af de genererede numre af værdien fra en analog port, som svæver. Den tilfældige HEX værdi skal bruges som ID for patienten der modtager konditioneringsbehandling

```

1 randomSeed(analogRead(A5));
2 long randNumber = random(100000, 999999); /
3 String randNumberHEX = String(String(randNumber, HEX) + ".csv");
4 return randNumberHEX;

```

Når der er lavet et 6-cifre tilfældigt nummer, bliver dette konverteret til en HEX, så værdien nu indeholder tal mellem 0-9 og bogstaver mellem A-F. Denne værdi bliver returneret som en string.

### Metode: checkFilesSD()

**Parameter:** *File dir, String val*

**Returtype:** *String*

**Beskrivelse:** Metode der kontrollere alle filer på SD kortet. Hver gang der findes en fil, tjekkes der for om de sidste 4 karakterer matcher med karaktererne “.csv”. Hvis den fundne fil matcher dette, bliver hele filnavnet gemt i en variable og returneret. Hvis der ikke findes et fil med endelsen “.csv” køres metoden *generateRandomNumber* og der oprettes en ny .csv fil.

```

1 String tempName, tempType, finalFile;
2 String valToCheck = ".csv";
3 File root = SD.open("/"); //Tell the method where to look on the SD card
4 while(true){
5   File entry = root.openNextFile();
6
7   //When a file is found, check the last four characters
8   tempName = entry.name();
9   tempType = tempName.substring(5,9);
10
11  //If a .csv is found
12  if(tempType.equalsIgnoreCase(valToCheck)){
13    Serial.print("/** A file was found with name: "); Serial.println(tempName);
14    finalFile = tempName;
15    break;
16  }
17  //If no file was found, creates a file
18  if(!tempType.equalsIgnoreCase(valToCheck) && !entry){
19    String newFileName = generateRandomNumber();
20    //Create a new file with the random HEX as filename and generate a new header
21    Serial.print("/** A new file was created: "); Serial.println(tempName);
22    createFileTemplate(newFileName);
23    finalFile = newFileName;
24    break;
25  }

```

```

26 entry.close();
27 }
28 return finalFile;
29 }
```

For at sikre at alle filer på SD kortet kontrolleres, kører metode i et loop, som kun brydes hvis den finder en .csv fil, eller hvis alle filer er kontrolleret og der ikke blev fundet en .csv fil.

#### Metode: `createFileTemplate()`

**Parameter:**

**Returtype:** `String filename`

**Beskrivelse:** Når der skal laves en ny fil på SD-kortet, skal der skrives en header til hver kolonne i .csv filen. Denne metode modtager et filnavn, som den åbner og skriver i en header til.

#### Metode: `writeToSDCard()`

**Parameter:** `String textToSD`

**Returtype:** `void`

**Beskrivelse:** Denne metode modtager en tekst, som den skriver til .csv filen på SD kortet.

#### Metode: `getFilename()`

**Parameter:** **Returtype:** `String` **Beskrivelse:** Simple metode der returnere en lokal variable, som indeholder navnet på den fundne .csv fil

### 13.4.3 Klasse: InternalMemory

Klassen gør brug af biblioteket *EEPROM*, dette bibliotek gør kommunikation muligt med EEPROMen.

#### Metode: `writeToEEPROM()`

**Parameter:** `int adr, unsigned short value`

**Returtype:** `void`

**Beskrivelse:** Simple metode der skriver `value` til adressen `adr` på EEPROM.

#### Metode: `readFromEEPROM()`

**Parameter:** `int adr`

**Returtype:** `unsigned short`

**Beskrivelse:** Metode der læser værdien på adressen `adr` på EEPROM.

### 13.4.4 Klasse: PressureSampling

#### Metode: `getCuffPressure()`

**Parameter:**

**Returtype:** `unsigned short`

**Beskrivelse:** Returnerer det aktuelle tryk i manchetten ved at sample 10 gange og tage middelværdien

**Metode: runningPeakDetect()**

**Parameter:** *unsigned short peaks[], unsigned short cuffPressure[], unsigned short peakArrayLength, unsigned short \*totalNumberOfPeaks, PressureControl pc, Utilities util*  
**Returtype:** *void*

**Beskrivelse:** Denne metode anvender en pointer til et array med peak værdier, et array med manchettryk værdier, en variabel med værdien tilsvarende længden af de to arrays, samt en pointer til en variabel hvor metoden skriver hvor mange peaks, som der er blevet fundet. De sidste parametre er bare objekter af de klasser som indeholder metoder der skal bruges i runningPeakDetect(). Metoden sampler et sample ad gangen (i alt 13) og tjekker om værdien er højere end middelværdien af de 6 samples før og de 6 samples efter. Hvis sample x(n-6) er størst og er højere end thresholdværdien for et detekteret puls signal, så gemmes peak værdien i "peaks[]" og manchettrykket i "cuffPressure[]".

```

1   if(timestamp<millis()-400 && n6>threshold && (n12+n11+n10+n9+n8+n7)/6 < n6 &&
2       (n+n1+n2+n3+n4+n5)/6 < n6)
3   {
4     peaks[tN0Peaks] = n6;
5     cuffPressure[tN0Peaks] = getCuffPressure();
6     tN0Peaks++;
}
```

If sætningen sikre også at peaks har mindst 400ms i afstand. Ved at have minimum afstand mellem detekteret pulssignal sikres at flere peaks på en puls ikke detekteres. Metoden fortsætter med at sample ind til manchet trykket når under 40mmHg, hvorefter den stopper.

```

1   while(currentPressure > util.mmHgToRaw(40))
2   {
```

Fordi de to arrays, som indeholder henholdsvis peaks og cuffpressure er prealokeret i hukommelsen, sættes de resterende ubrugte pladser = NULL. Dette sikrer at de værdier, som ellers måtte være liggende i hukommelsen ikke bliver forvekslet med en peak amplitude.

```

1   for(i = tN0Peaks; i < peakArrayLength; i++)
2   {
3     peaks[i] = NULL;
4     cuffPressure[i] = NULL;
5   }
```

Til sidst lukkes den resterende luft ud af manchetten, altså de sidste 40mmHg.

## 13.5 Namespace: Global



**Figur 13.9.** Klasse diagram over det globale namespace

### 13.5.1 Klasse: Utilities

Denne klasse tilhører det globale namespace og kan tilgås af alle namespaces. Klassen skal ses som en hjælpe klasse med funktioner der kan være brugbare flere steder

#### Metode: rawToMmHG()

**Parameter:** *unsigned short rawPressure*

**Returtype:** *double*

**Beskrivelse:** Metoden konverterer rå ADC værdi til mmHg.

#### Metode: mmHgToRaw()

**Parameter:** *unsigned short mmHgPressure*

**Returtype:** *double*

**Beskrivelse:** Metoden konverterer mmHg værdi til rå ACD værdi.

### 13.5.2 Klasse: Konditioneringsapparat.pde (Main fil)

Denne klasse er softwaren main fil, her samles funktionaliteten i to metoder: *setup()* og *loop()*, samt 5 interrupt service rutiner. Dette er begge metoder som arduino skal have. *setup()* bliver kørt som det første når arduinoen startes og derefter køres det uendelige *loop()*. Klassen kender kun til de to klasser i GUI laget, Buttons og Display Da interrupt ikke kan sættes fra andre steder end main filen, bliver der initieret 5 interrupt i denne klasse, hhv to til konditioneringsforløbet, en til okklusionstræning og to til setup.

#### Metoder: intCon\_ISR(), intBT\_ISR(), intOcc\_ISR(), intCha\_ISR() og intSel\_ISR()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Disse fem interrupt service rutiner kalder alle sammen deres respektive metode fra klassen Buttons. Alle metoder indeholder et delay på 100 ms og derefter et if statement, som tjekker om interrupt pin'en stadig er høj, dette software imødekommer debouncing, se afsnit 15.2

#### Metode: setup()

**Parameter:**

**Returtype:** *void*

**Beskrivelse:** Denne metode bruges til opsætning af arduino, og det er derfor også her at det afgøres om arduinoen skal køre hhv konditioneringsforløb, okklusionstræning eller setup. Derfor aflæses værdien af 3 digital porte for at afgøre hvilket program der skal køres. Derefter opsættes de respektive interrupt for det valgte program. Dette er nødvendigt da prototypen kun har 2 knapper og disse skifter funktion efter hvilket program der er valgt

```

1     programToRun = btt.readModeSwitch();
2
3     //Setup for i interrupts
4     pinMode(interruptPin0, INPUT);
5     pinMode(interruptPin1, INPUT);

```

```
6
7     switch(btt.readModeSwitch()){
8         case 1: //Conditioning
9             attachInterrupt(digitalPinToInterruption(interruptPin0), intCon_ISR, RISING);
10            attachInterrupt(digitalPinToInterruption(interruptPin1), intBT_ISR, RISING);
11            break;
12            case 2: //Occlusion
13                attachInterrupt(digitalPinToInterruption(interruptPin0), intOcc_ISR, RISING);
14                break;
15                case 3: //Setup
16                    attachInterrupt(digitalPinToInterruption(interruptPin0), intCha_ISR, RISING);
17                    attachInterrupt(digitalPinToInterruption(interruptPin1), intSel_ISR, RISING);
18                    break;
19    }
```

### Metode: loop()

**Parameter:**

**Returtype:** *void*

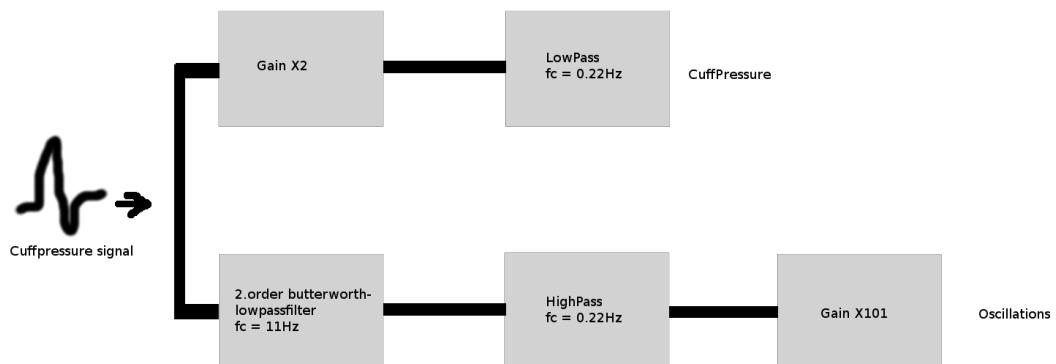
**Beskrivelse:** Uendeligt loop der eksekveres efter metoden setup() har kørt. Metoden indeholder en switch case struktur, og program valget afgøre hvilken case der køres. Der kan kun skiftes case, hvis arduinoen genstartes.



# 14 | Filter design

## 14.1 Analoge filtre

Det analoge filter design har blandt andet funktionen at isolerer og forstærke DC niveauet fra tryksensoren, som er koblet til manchetten. Tryksensoren (MPX5100) er lineær og kan derfor ud fra en enkel koefficient kalibreres.<sup>1</sup> DC niveauet skal forstærkes for at øge ADC opløsningen i forhold til mmHg per bit. Arduino MEGA 2560 har en ADC opløsning på 10bit. I volt er dette en opløsning på  $5/1023=4.9\text{mV}$ . Sensoren har en sensitivitet på  $45\text{mV}/\text{kPa}$ , svarende til  $6\text{mV}/\text{mmHg}$ . Uden DC forstærkning er opløsningen altså  $4.9/6=0.817\text{mmHg}$ . Ved at anvende gain X2 øges opløsningen til  $0.817/2=0.408\text{mmHg}$ . Ren DC opnås ved at lavpas filteret fjerner alt AC over knækfrekvensen.



**Figur 14.1.** Filter model med inputsignal til venstre og de to output signaler til højre

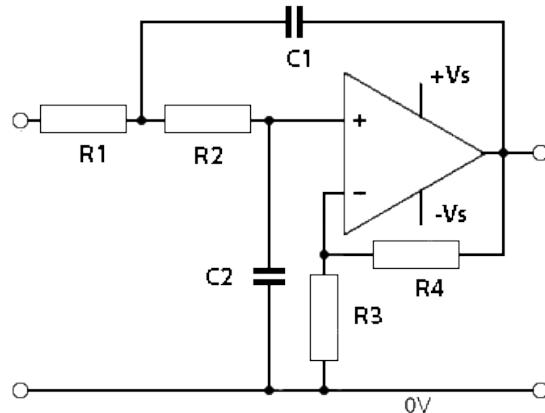
CuffPressure er forstærket DC og Oscillations er forstærket AC mellem 0.22Hz og 11Hz

Oscillationerne i manchetten isoleres med 11Hz anden ordens butterworth for at opnå god dæmpning på 50Hz brummen. L.A. Geddes<sup>1</sup> et al. Sætter knæk frekvensen til 30Hz, men efter som at puls signalet befinner sig på ca 1Hz og at Geddes med høj sandsynlighed har anvendt kaskade filtre sættes butterworth filterets knæk til 11Hz, svarende til elve afledte af puls grund frekvensen. Højpasfilteret fjerner DC ved at knække ved 0.22Hz. For at forstærke oscillationerne op i en størrelse, som arduinoen kan sample forstærkes det pulserende signal med gain X101.

<sup>1</sup>Fixme Fatal: REFERENCE

## 14.2 Komponent udregninger

### 14.2.1 Anden ordens butterworth filter



**Figur 14.2.** 2. ordens butterwoth filter

Normaliseret polynomie som producerer et anden ordens butterwoth respons<sup>2</sup>

$$s^2 + 1.414s + 1 = 0 \quad (14.1)$$

$$\zeta = \frac{1.414}{2} = 0.707 \quad (14.2)$$

Der bruges en anden metode til at bestemme  $R$  og  $C_1$  og  $C_2$ . Det bestemmes at  $R = 100k\Omega$ , for at mindske spændingsdelen mellem kredsløbet og spændingskilden.

Den ønskede knækfrekvens er  $f_c = 11Hz$  og efter som  $f_c = \frac{\omega_c}{2\pi}$  så er  $\omega_c = f_c * 2 * \pi = 22\pi$

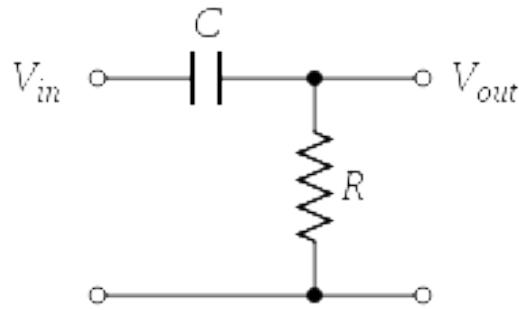
Nu løses de to ligninger med de to ubekendte  $R\sqrt{C_1 * C_2} = \frac{1}{\omega_0}$  og  $\frac{C_2}{C_1} = \zeta^2$  og  $C_1$  og  $C_2$  bliver:

$$C_1 = 2.046 * 10^{-7}F \text{ eller } C_1 = -2.046 * 10^{-7}F$$

$$C_2 = 1.023 * 10^{-7}F \text{ eller } C_2 = -1.023 * 10^{-7}F$$

<sup>2</sup>FiXme Fatal: Kilde: the analysis & design of linear circuits – side 755

### 14.2.2 Høj pas filter



**Figur 14.3.** 1. ordens høj pas filter

Knæk frekvens  $f_c = 0.3$

Modstand værdi,  $R = 1M\Omega$

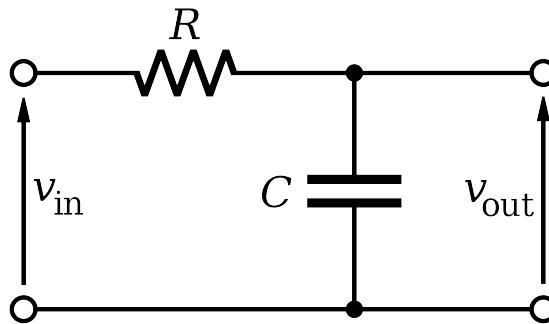
Nu udregnes værdien af kondensatoren ved at isolere C:

$$f_c = \frac{1}{2\pi * R * C} \Rightarrow C = \frac{1}{2\pi * R * f_c} = \frac{1}{2\pi * 1M\Omega * 0.3} = 5.3052 * 10^{-7} F \quad (14.3)$$

Dette bliver en kondensator på 530nF, den nærmeste komponent hedder 680nF og derfor udregnes den nye knæk frekvens til:

$$f_c = \frac{1}{2\pi * R * C} = \frac{1}{2\pi * 1M\Omega * 680 * 10^{-9}} = 0.23 Hz \quad (14.4)$$

### 14.2.3 Lav pas filter



**Figur 14.4.** 1. ordens lav pas filter

Knæk frekvens  $f_c = 0.3$

Modstand værdi,  $R = 1M\Omega$

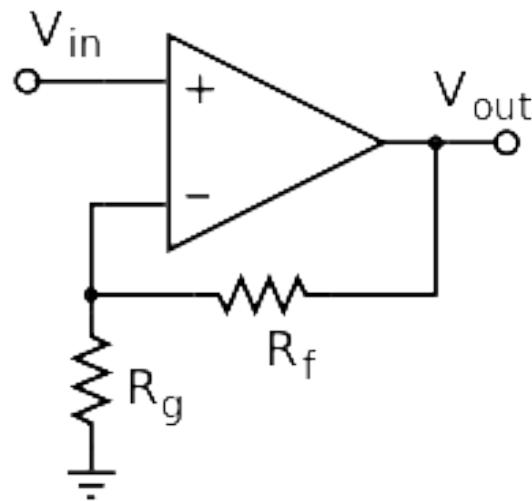
Nu udregnes værdien af kondensatoren ved at isolere C:

$$f_c = \frac{1}{2\pi * R * C} \Rightarrow C = \frac{1}{2\pi * R * f_c} = \frac{1}{2\pi * 1M\Omega * 0.3} = 5.3052 * 10^{-7} F \quad (14.5)$$

Dette bliver en kondensator på 530nF, den nærmeste komponent hedder 680nF og derfor udregnes den nye knæk frekvens til:

$$f_c = \frac{1}{2\pi * R * 680 * 10^{-9}} = 0.23Hz \quad (14.6)$$

#### 14.2.4 Gain på manchet oscillationer



**Figur 14.5.** Gain til manchet oscillationer

##### Udregning af gain:

Der ønskes et gain på 2, dvs  $A = 100$

$$A = \frac{V_0}{V_i} = 1 + \frac{R_f}{R_g} \quad (14.7)$$

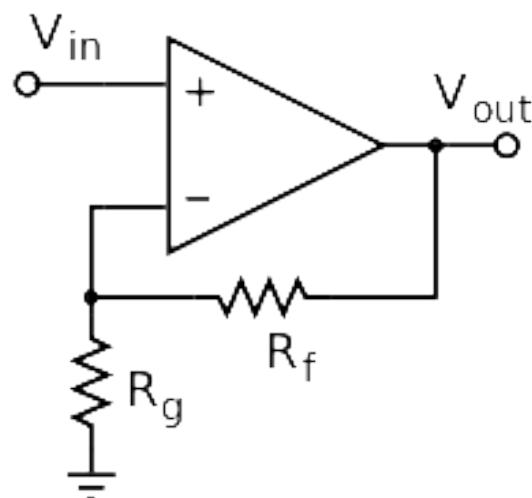
##### Udregning af komponenter:

Indsættelse af komponent værdier,  $R_f$  sættes til  $100k\Omega$  og  $R_g$  udregnes:

$$A = 1 + \frac{R_f}{R_g} \Rightarrow R_g = 1010 \quad (14.8)$$

Derfor vælges  $R_g$  til at være  $1k\Omega$

#### 14.2.5 Gain på manchettryk signal



**Figur 14.6.** Gain til manchettryk signal

**Udregning af gain:**

Der ønskes et gain på 100, dvs  $A = 2$

$$A = \frac{V_0}{V_i} = 1 + \frac{R_f}{R_g} \quad (14.9)$$

**Udregning af komponenter:**

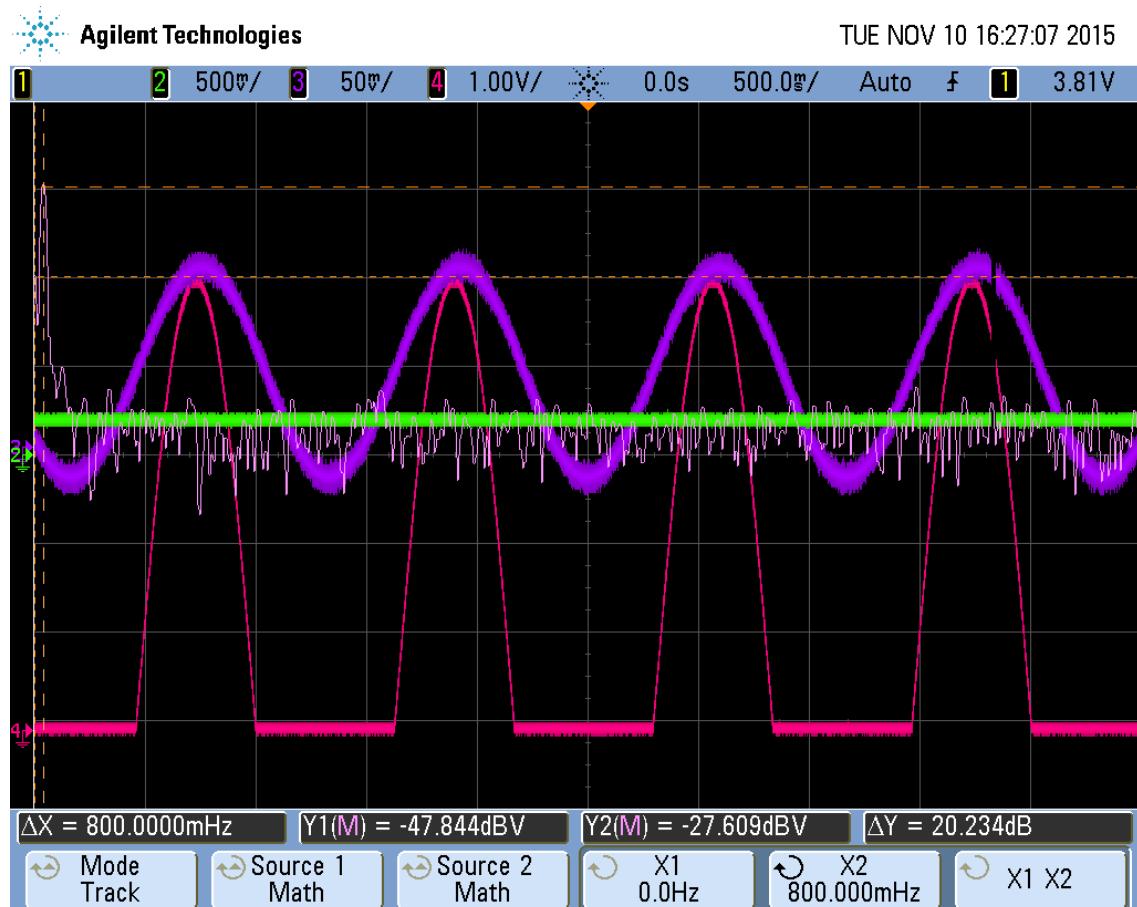
Indsættelse af komponent værdier,  $R_f$  sættes til  $100k\Omega$  og  $R_g$  udregnes:

$$A = 1 + \frac{R_f}{R_g} \Rightarrow R_g = 100000 \quad (14.10)$$

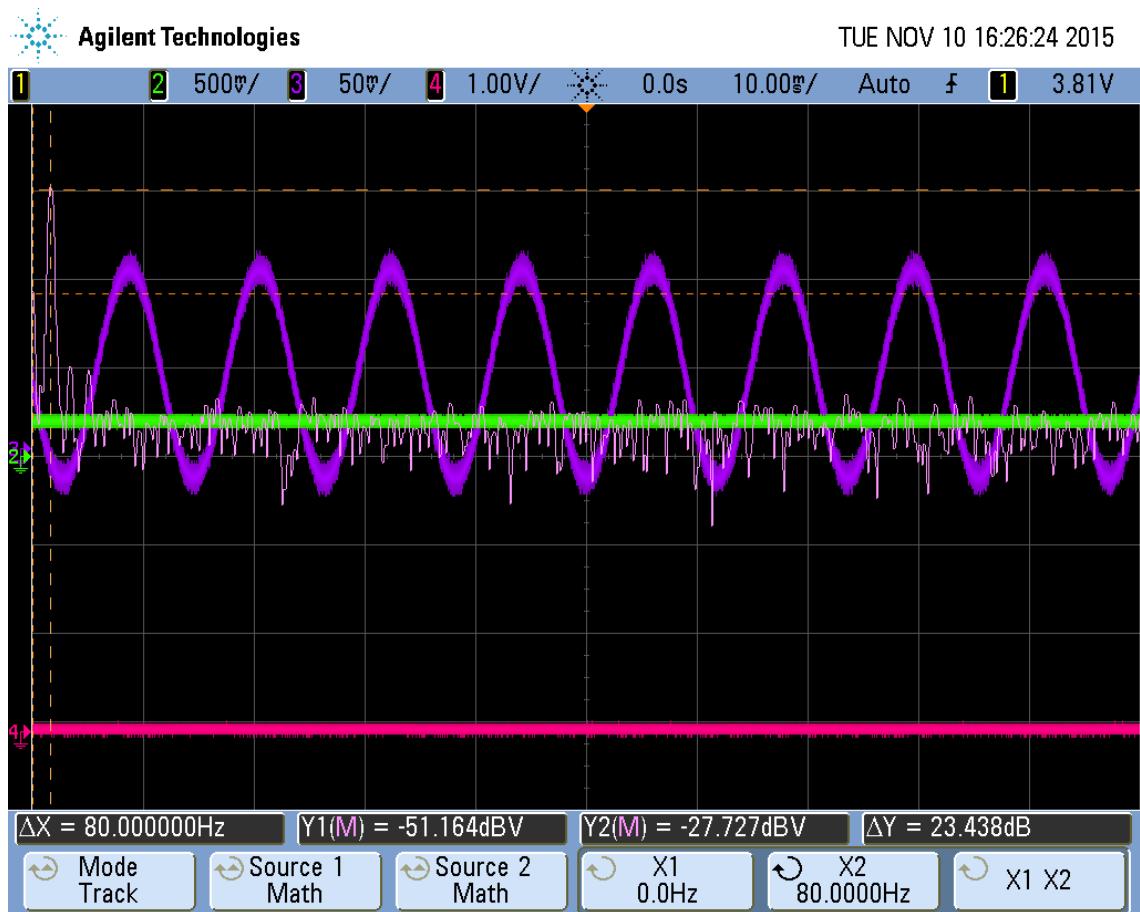
Derfor vælges  $R_g$  til at være  $100k\Omega$

### 14.3 Praksis

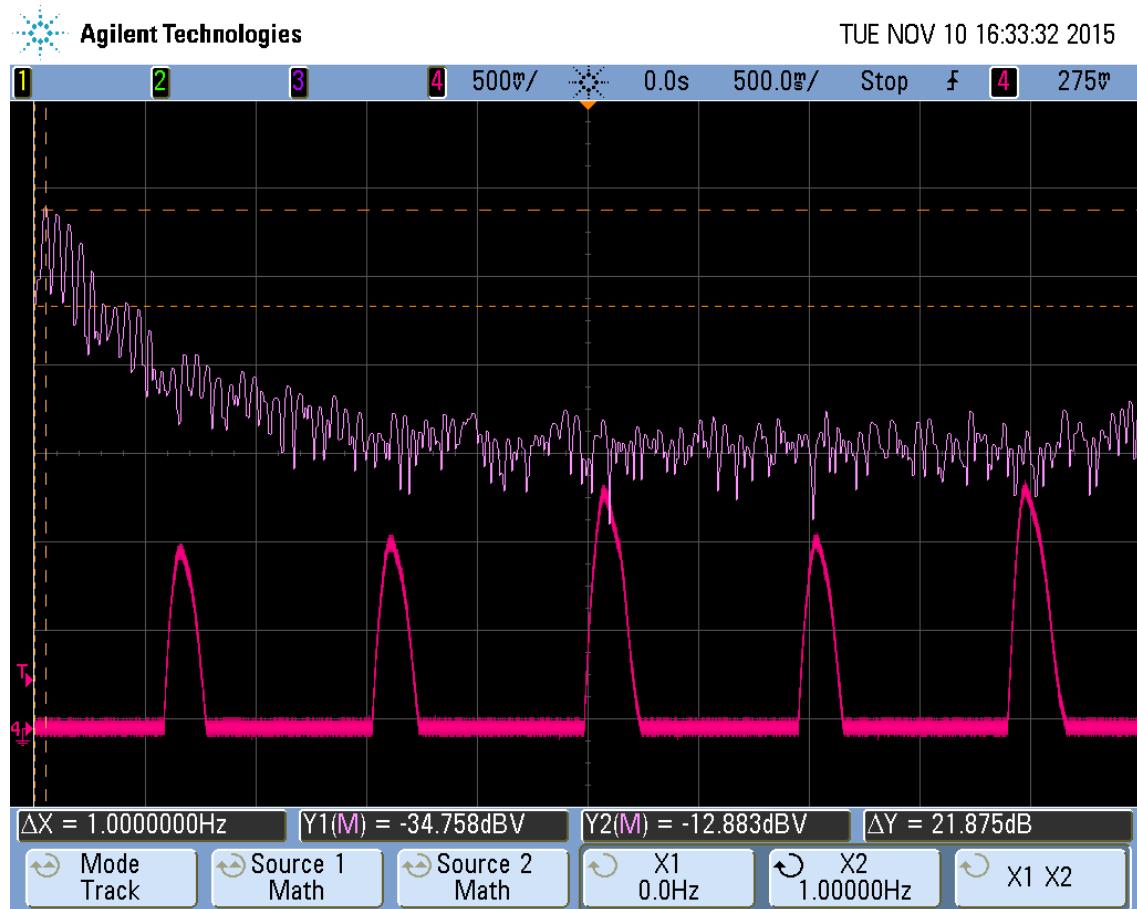
Signal generator med frekvens 0.8Hz svarende til en sund puls. Simuleringen er vist på billederne under. Den lilla kurve er inputsignalet, rød er det oscillerende udgangssignal efter filteret og den grønne kurve er udgangssignalet efter det ikke oscillerende filter.



**Figur 14.7.** Lilla er input sinus 0.8Hz. Grøn er ren DC. Rød er det filtrerede signal med oscillationerne. Den lyserøde kurve er FFT af det lilla signal, hvor det ses at det består af 0.8Hz grundtone.

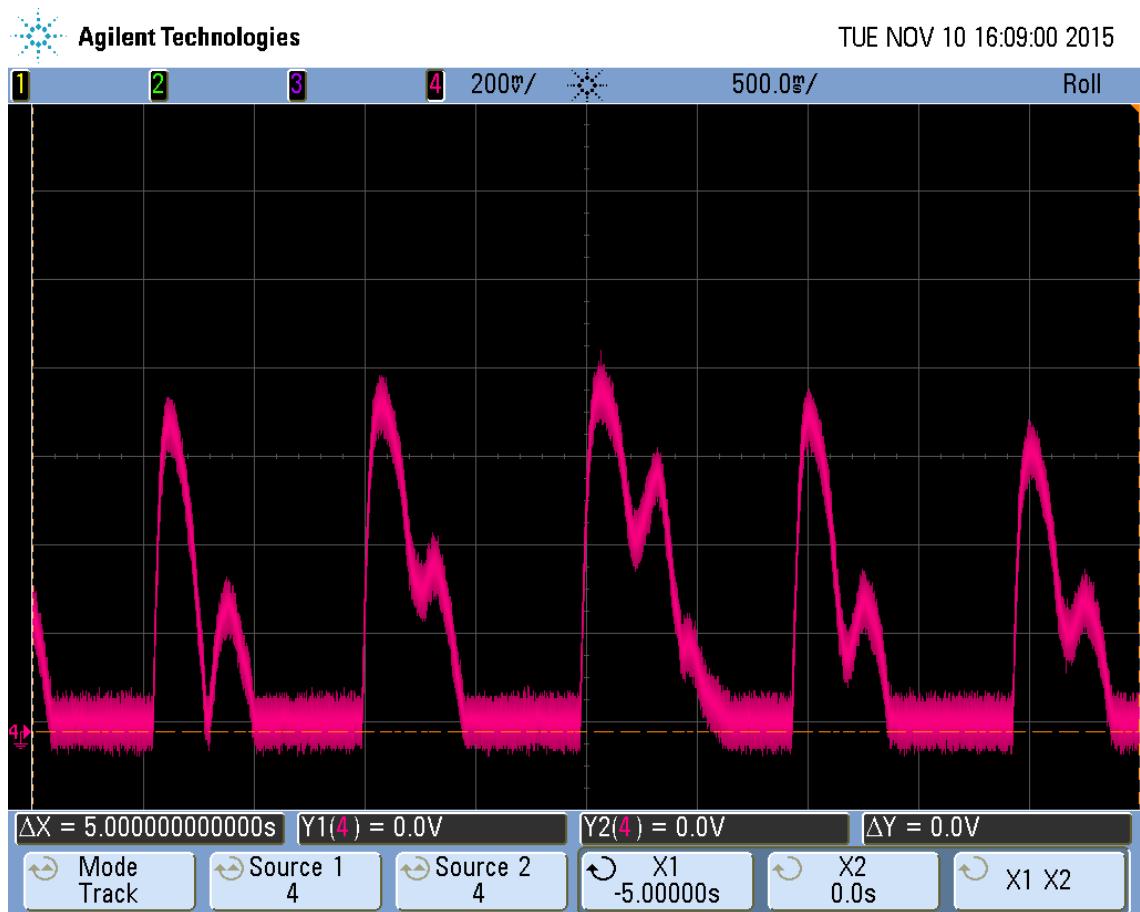


**Figur 14.8.** Lilla er input sinus 80Hz. Grøn er ren DC. Rød er det filtrerede signal med oscillationerne. Den lyserøde kurve er FFT af det lella signal, hvor det ses at det består af 80Hz grundtone.



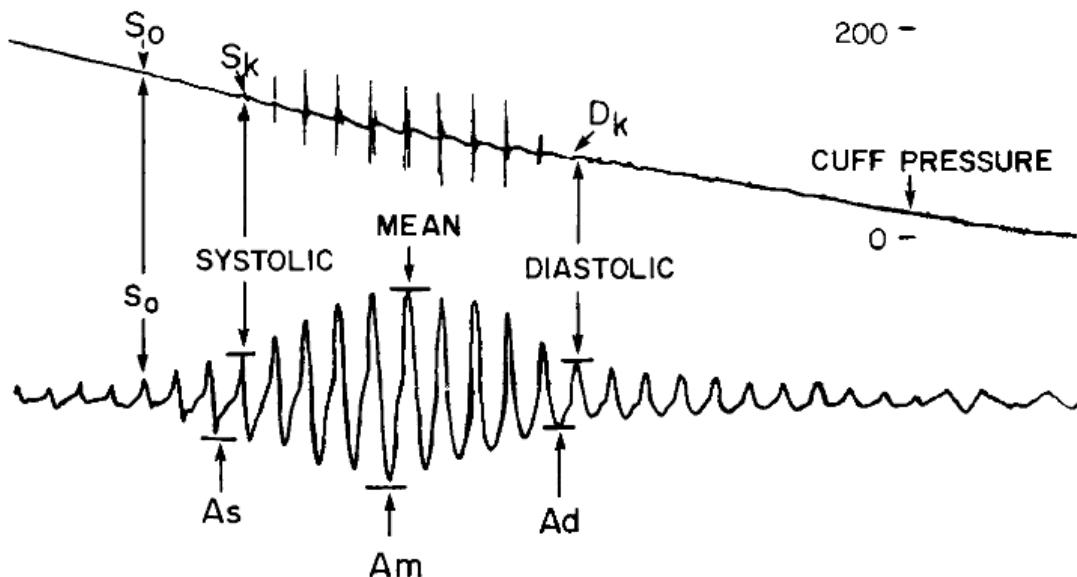
**Figur 14.9.** Oscilloskop billedet viser udgangssignalet efter den oscillérende filtrering (Rød) og en FFT af signalet (lyserød). Her ses det tydeligt at grundtonen er ca 1Hz og har mindst 4 afledte der af (2,3,4,5 Hz), som giver den spidse smalle form.

Hvis manchet signalet ikke simuleres, men i stedet måles på et rigtigt menneske ligner signalet ikke perfekte sinuser, men i stedet korte udsving, med en grundfrekvens og en masse overtoner. På nedenstående billede ses forskellige typiske udgangssignaler under en normal blodtryksmåling.



**Figur 14.10.** Når manchet trykket nærmer sig det systolliske tryk observeres der et signal som ser lidt anderledes ud

Det ses meget tydeligt på billede (se figur 14.9 og 14.10) at det pulserende signal svinger meget med hensyn til peak amplituden over tid. Fænomenet som ses kan skyldes mange ting, her iblandt respirationen og andre mekaniske bevægelse. Den analoge filtrering når sin begrænsning, da den ikke kan filtrere peak amplituden til at passe en optimal blodtryksmåling, som kan ses på figur 14.11, hvor amplituderne er konstant stigende ind til MAP og så aftagende efterfølgende. Til at håndterer dette problem anvendes der digital filtrering.



**FIGURE 1.** Cuff-pressure with superimposed Korotkoff sounds and amplified oscillations in cuff pressure. The symbols identify the measurements used to identify systolic and diastolic pressure (see text).

*Figur 14.11.* Signal fra oscilometrisk blodtryksmåling

Reference <sup>3</sup>

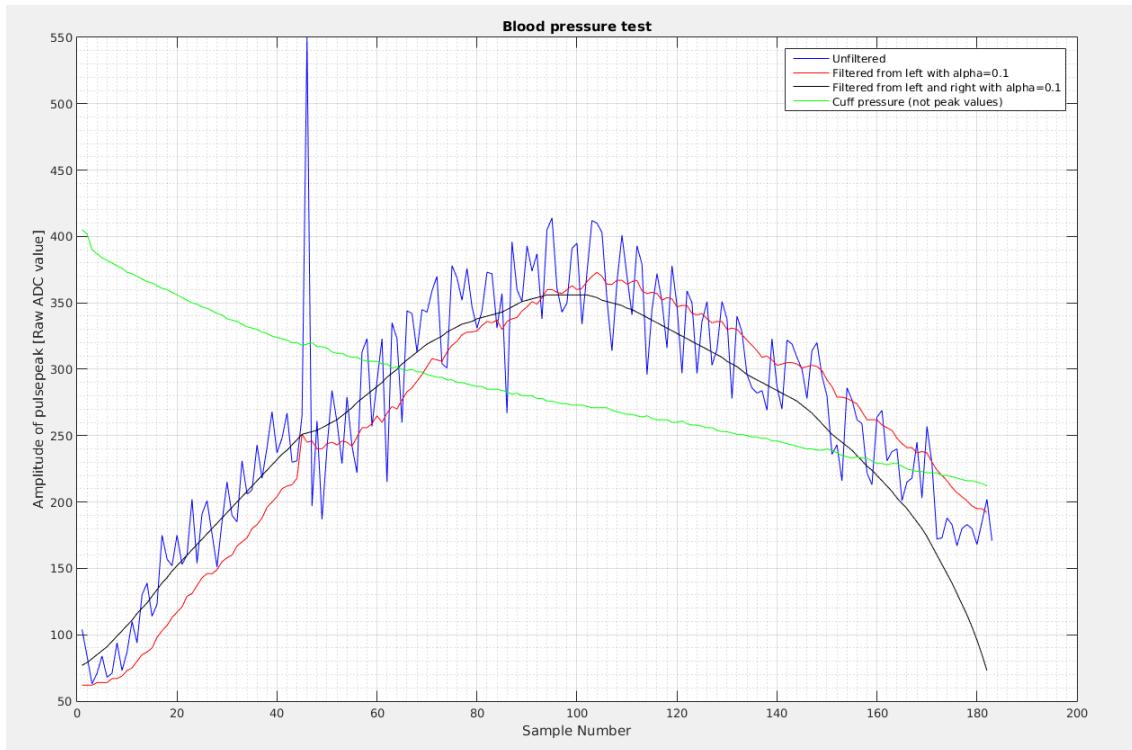
#### 14.4 Digital filtrering

Til digital filtrering er der anvendt et ekspotentiel midlingsfilter med zero group delay.

$$y(n) = \alpha x(n) + (1 - \alpha)y(n - 1) \quad (14.11)$$

Filteret midler over peak amplituderne, så den gennemsnitlige amplitude forøgelse/dæmpning kommer til udtryk. Blodtryks filteret anvender en alfaværdi på  $\alpha = 0.11$ . Ydermere filtreres data fra begge sider, altså først fra venstre mod højre og så bag efter fra højre mod venstre. Denne teknik sikrer ingen group delay, som er vigtigt for at kunne bestemme MAP, som det tryk der er i manchetten på samme tidspunkt som den maksimal målte peak amplitude. Dette er bedst illustreret på figur 14.12, hvor manchettrykket er faldende over tid og peak amplituderne er stigende og efter MAP så faldende. Når det rå signal midles fra venstre mod højre opstår et group delay på peak amplituderne, som ikke er på manchet tryk data'en. Af denne grund må der ikke være group delay på signalet.

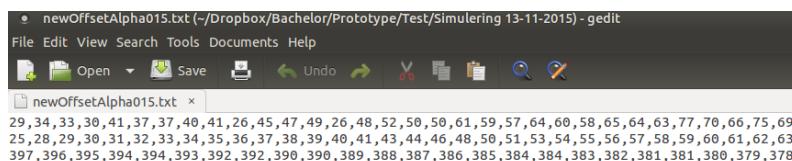
<sup>3</sup>FiXme Fatal: Billedet er taget fra CHARACTERIZATION OF THE OSCILLOMETRIC METHOD FOR MEASURING INDIRECT BLOOD PRESSURE



**Figur 14.12.** Graf med data fra en blodtryksmåling. Her ses at det rå signal er støjfyldt. Grøn er manchet trykket, blå er de rå peak amplituder, rød er filtreret en gang fra venstre mod højre, sort er filtreret fra begge sider

#### 14.4.1 Matlab

Til udarbejdelsen af det digitale filter er der anvendt matlab på det rå signal. Det rå signal med peak amplituderne og manchet trykket er udprintet gennem serielparten og derefter gemt i .txt fil for at blive importeret til matlab. Et udkast af en sådan fil kan ses i figur 14.13. Første række indeholder de rå peaks, anden række de filtrerede peaks og tredje række indeholder manchet trykket.



**Figur 14.13.** Udklip af datafil som importeres til matlab

Matlab figuren kan se på figur 13.7 og koden til fremstilling af denne kan læses under.

```

1 sysCoefficient = 0.38;           %Lokation of SYS in percentage of MAP
2 diaCoefficient = 0.48;           %Lokation of DIA in percentage of MAP
3
4 figure(1)                      %Figure number
5 plot(newOffsetAlpha015(1,:),'b') %First plot
6 hold on;
7 plot(newOffsetAlpha015(2,:),'r') %Secund plot
8 plot(newOffsetAlpha015(3,:),'black')%Third plot
9
10 filtOsc = newOffsetAlpha015(2,:); %Reads data from 2 row

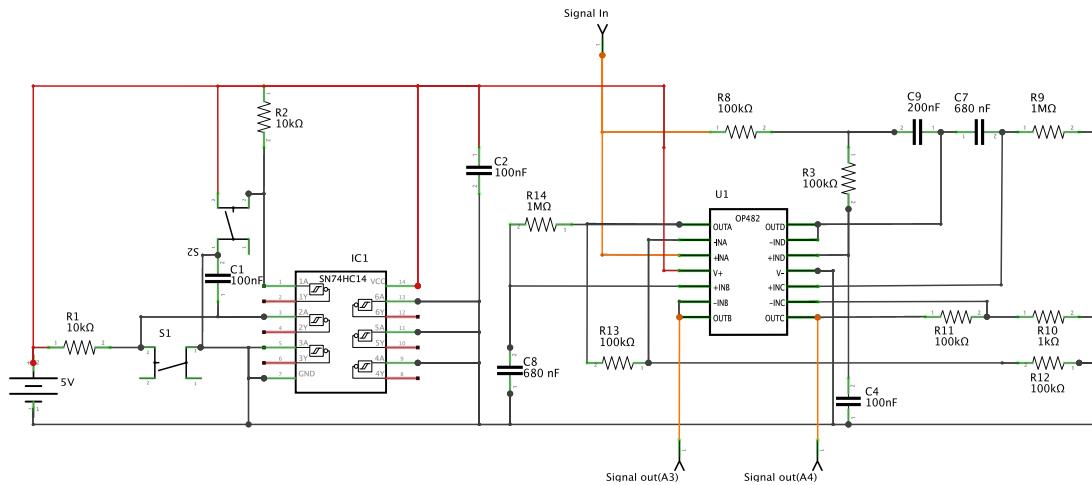
```

```
11 cuffPress = newOffsetAlpha015(3,:); %Reads data from 3 row
12 [M,I] = max(filt0sc); %Finds higest peak in oscilliations
13
14
15 [~,ISYS] = min(abs(filt0sc(1:I)-(M*sysCoefficient))); %Find location of SYS
16
17
18 [~,IDIA] = min(abs(filt0sc(I:end)-(M*diaCoefficient))); %Find location of DIA
19 IDIA = I+IDIA; %Adds the offset of MAP
20 scatter([ISYS, I+4, IDIA, ISYS, I+4, IDIA],[filt0sc(ISYS), M, filt0sc(IDIA), cuffPress(ISYS),
cuffPress(I+4), cuffPress(IDIA)],'or','LineWidth',5); %Plot SYS, MAP and DIA
21
22 %Labels for locations
23 str = ' \leftarrow SYS';
24 text(ISYS,filt0sc(ISYS),str)
25 str = ' \leftarrow SYS';
26 text(ISYS,cuffPress(ISYS),str)
27 str = ' \leftarrow MAP';
28 text(I,filt0sc(I+4),str)
29 str = ' \leftarrow MAP';
30 text(I,cuffPress(I+4),str)
31 str = ' \leftarrow DIA';
32 text(IDIA,filt0sc(IDIA),str)
33 str = ' \leftarrow DIA';
34 text(IDIA,cuffPress(IDIA),str)
35
36 %plot settings
37 grid on;
38 grid minor;
39 xlabel('Sample Number');
40 ylabel('Amplitude of pulsepeak [Raw ADC value]');
41 title('Blood pressure test');
42 legend on;
43 legend('Unfiltered','Filtered from left and right with alpha=0.15','Cuff pressure (not peak
values)','SYS, MAP and DIA');
```

# 15 | Hardware

## 15.1 Schmetic

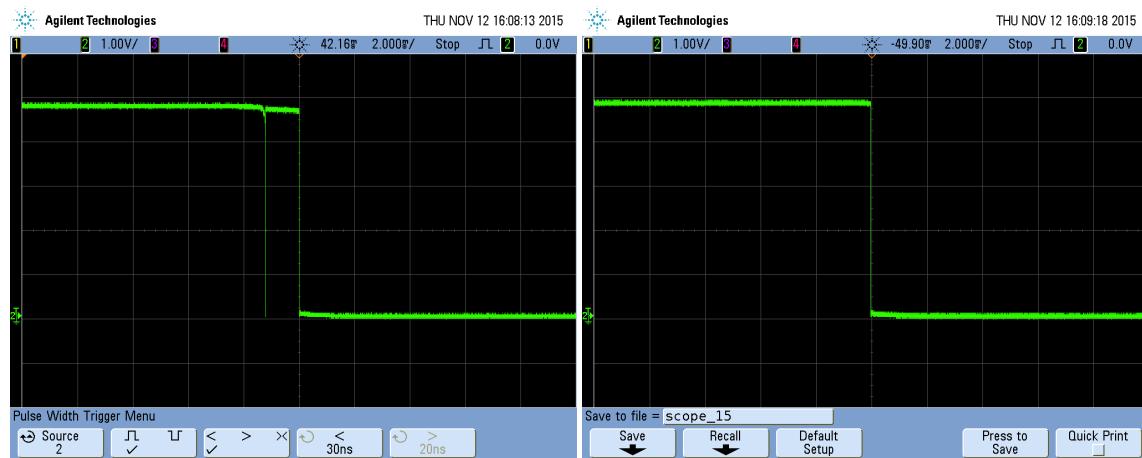
På figur 15.1 ses tegning over hardware setup. *Signal in*, *Signal out(A3)* og *Signal out(A4)* skal ses som interface med arduino og tryksensoren. Disse er undladt for ikke gøre diagrammet for kompleks



Figur 15.1.

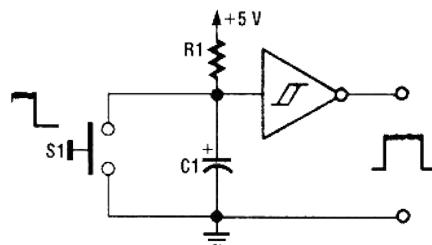
## 15.2 Knapper

For at imødegå debouncing ved tryk på knapperne, er der bygget et støttekredsløbs til disse. Debouncing kan ses på figur 15.2 hvor det mekaniske slip på knappen, giver anledning til en løs forbindelse ses på multimeteret som en lav efterfulgt af en høj, for så at blive lav igen. Microcontrolleren er hurtig nok til at registrer dette som et nyt knaptryk.



**Figur 15.2.** Figuren viser et knaptryk, hvor der **Figur 15.3.** Knaptryk hvor der er implementeret et anti debouncing kredsløb.

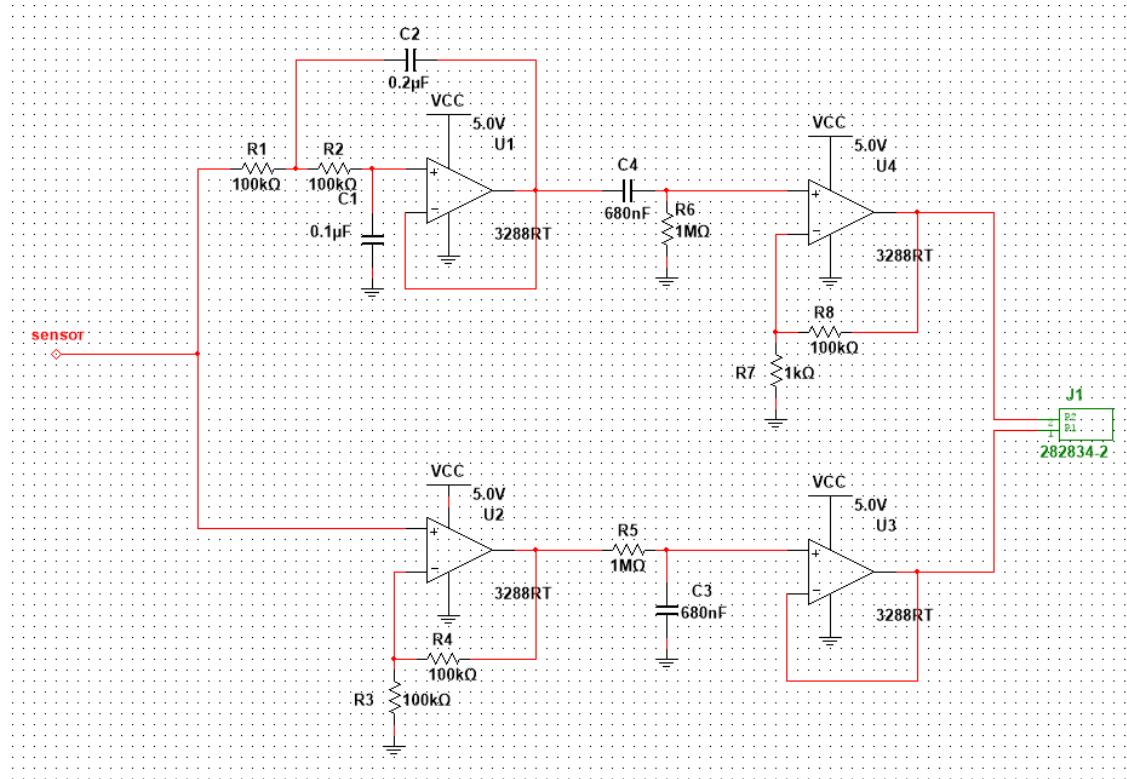
For at imødekomme er der implementeret et debounce kredsløb (figur 15.4) med en inverting schmitt trigger. Schmitt triggeren leverer en stabil høj og kapacitoren forhindrer den lav ved kortvarig open circuit under knaptryk.



**Figur 15.4.** Anti debouncing kredsløb

## 15.3 Filter

Filtreringen af signalet er opdelt i isolering af manchet tryk og isolering af pulserende signal. På figur 15.5 ses opdelingen af signalet i to til ADC'en. Dyberer forklaring af filterdesignet kan findes under afsnit 14 omkring filter design.

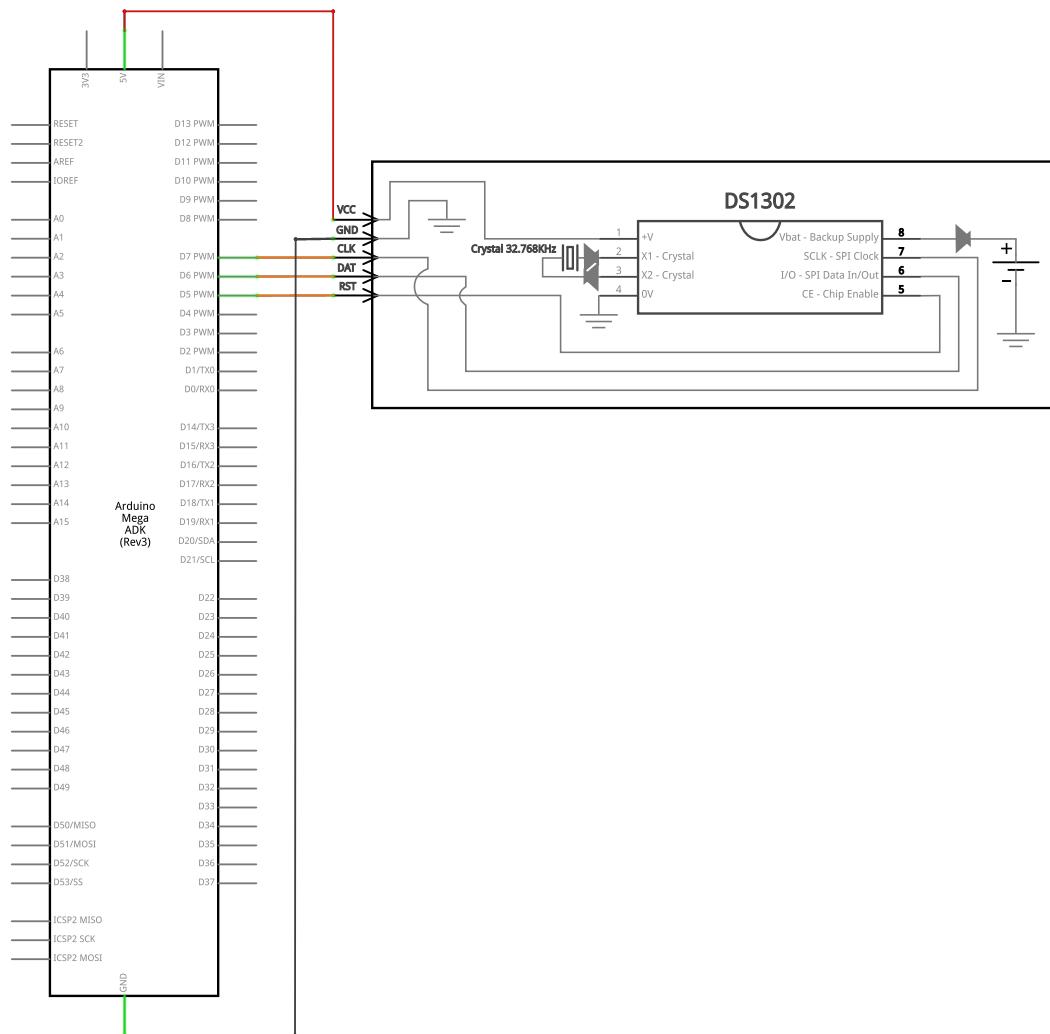


**Figur 15.5.** Schematics over filter design

## 15.4 Timer

For at kunne lave tidsstemplere til målinger med konditioneringsapparatet var prototypen nød til at have en real time clock, som kunne holde korrekt tid, selvom apparatet blev slukket. Dette er opnået ved hjælp af en *timekeeping chip* model DS1302 (Se datablad<sup>1</sup>). Når prototypen er tændt forsynes den via konditioneringsapparatets strømforsyning, og når prototypen er slukket forsynes DS1302 med et knapcelle batteri. Se figur 15.6 for at se hvordan timeren er komplet på arduino.

<sup>1</sup>FiXme Fatal: reference til datablad DS1302

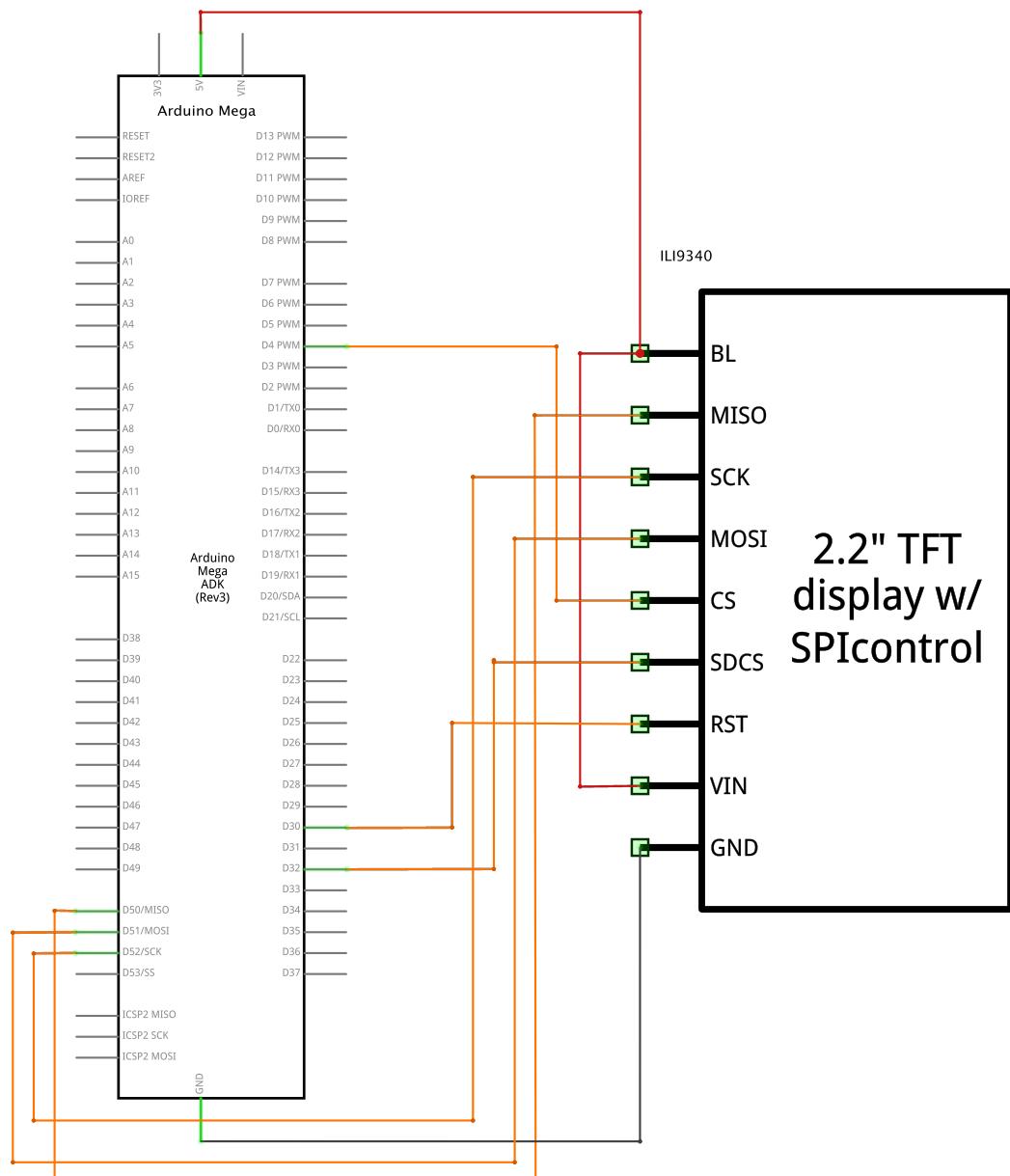


**Figur 15.6.** Oversigt over hvordan DS1302 er forbundet

## 15.5 Display

Til styring af bruger interfacet og feedback. Displayet er 2.2" med en oplosning på 320x240. Displayet styres med IC'en ILI9340 (Se datablad<sup>2</sup>). Kommunikation med displayet foregår via en SPI forbindelse som er koblet til Arduino Mega's digitale port 50, 51 og 52. For komplet interface se figur

<sup>2</sup>FIXme Fatal: Reference til ILI9340

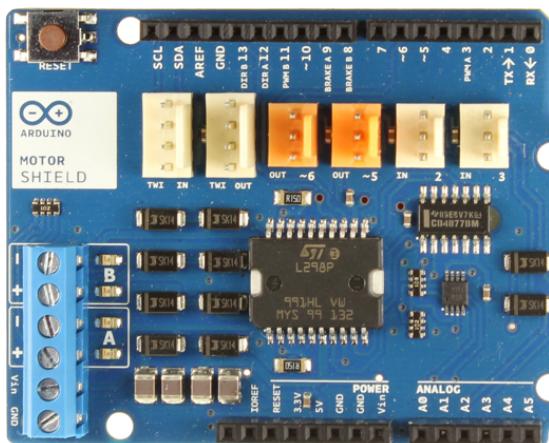


**Figur 15.7.** Oversigt over hvordan ILI9340 er forbundet

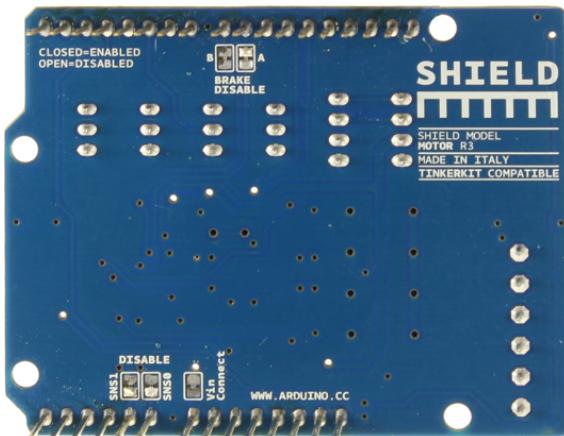
## 15.6 Motor shield

Arduino motor shield implementerer L298P (se figur 15.8 og 15.9 med selvinduktion beskyttelse og en galvanisk adskillelse af motor forsyning og den logiske strømforsyning. Schematics over boardet kan ses i bilag <sup>3</sup>. Desuden er motor shield modifieret, så det har en ekstern strøm forsyning på 12V. Derfor er 5V forbindelse mellem arduino og motor shield brudt.

<sup>3</sup>Fixme Fatal: reference til datablad



**Figur 15.8.** Arduino motorshield forside



**Figur 15.9.** Knaptryk hvor der er implementeret et anti debouncing kredsløb.