

Projektdokumentation

Semesterprojekt 3

Blodtryksmålersystem

ST3PRJ3-03

Gruppe 5

Sundhedsteknologi

Århus Universitet, IHA

Vejleder: Samuel Alberg Thyresøe

Dato: 16/12 2015

Mette Østergård Knudsen, 201404501

Ida Mark Skovbjerg, 201404669

Line Skov Larsen, 201405838

Brian Hansen, 201310502

Mohamed Hussein Mohamed, 201370525

Khaled Edwan, 200800899

Indhold

Indhold	i
1 Indledning	1
2 Kravspecifikation	2
2.1 Godkendelsesformular	2
2.2 Indledning	2
2.3 Systembeskrivelse	3
2.4 Aktør-kontekst diagram	4
2.5 Use cases	6
2.6 Ikke-funktionelle krav	10
3 Arkitektur og design	13
3.1 Hardware design	13
3.2 Software design	25
3.3 Integrationstest	65
4 Accepttest	66
4.1 Indledning	66
4.2 Accepttest for funktionelle krav	66
4.3 Accepttest for ikke-funktionelle krav	72
4.4 Godkendelses formular	76
Litteratur	77
4.5 Referencer	77

Kapitel 1

Indledning

Kapitel 2

Kravspecifikation

2.1 Godkendelsesformular

Forfattere	Ida, Line, Mette, Brian, Mohamed og Khaled
Godkendes af:	Samuel Alberg Thrysøe
Antal sider:	
Kunde:	IHA

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

Sted og dato:

Kundens underskrift

Leverandørens underskrift

2.2 Indledning

Denne kravspecifikation er blevet udarbejdet på baggrund af krav fra kunden, samt hvad leverandøren finder muligt. Kravspecifikationens formål er at specificere de krav, der er til produktet.

2.3 Systembeskrivelse

Det er blevet besluttet ud fra projektformuleringen at udvikle et blodtryksmålersystem som en prototype med perspektivering til fremtidigt arbejde. Blodtryksmålersystemet ønskes ideelt at kunne måle blodtryk, EKG og iltmætning for en patient. Ud fra blodtrykket findes systolisk, diastolisk og middeltryks værdier, dette gøres ved at finde den maksimale værdi (systole) og den minimale værdi (diastole) for blodtrykskurven. Ud fra disse to værdier bestemmes middeltrykket, dette gøres ved formlen: $middeltryk = 1/3 \cdot systole + 2/3 \cdot diastole$. [2]

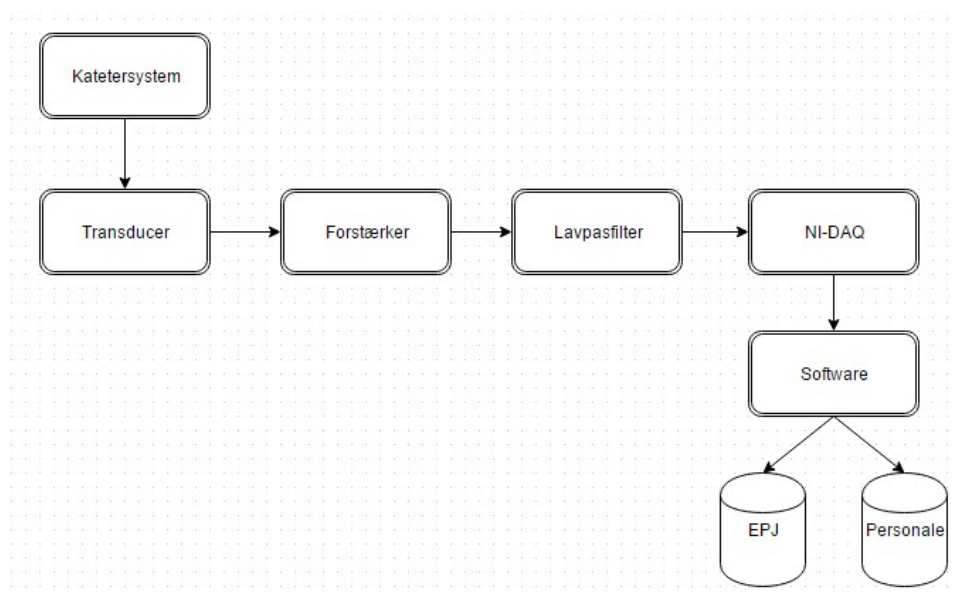
Ud fra EKG-signalet kan pulsen bestemmes, dette gøres ved at bestemme antallet af R-takker på et minut. Desuden kan pulsen bestemmes ud fra blodtrykket, da pulsen her er antallet af systoliske/diastoliske værdier på et minut. Iltmætningen er mængden af ilt i blodet. For at kunne bestemme denne værdi skal et eksternt produkt benyttes. Dette produkt skal ved hjælp en infrarød sensor bestemme iltmætningen i blodet, dette bliver ikke implementeret i denne prototype.

Forudsætninger for brug af systemet er, at det skal køres på en computer, samt overholde de opstillede krav. Systemet er opbygget af en hardwaredel og en softwaredel.

Hardwaredelen består af et aktivt 2. Ordens Butterworth Sallen-Key lavpasfilter, samt en forstærker. Forstærkeren modtager et signal fra den udleverede transducer, dette signal forstærkes op. Signalet sendes videre til lavpasfilteret hvor alle frekvenser over 50Hz bliver dæmpet. Herfra sendes signalet ind i dataopsamlingsenheden NI-DAQ.

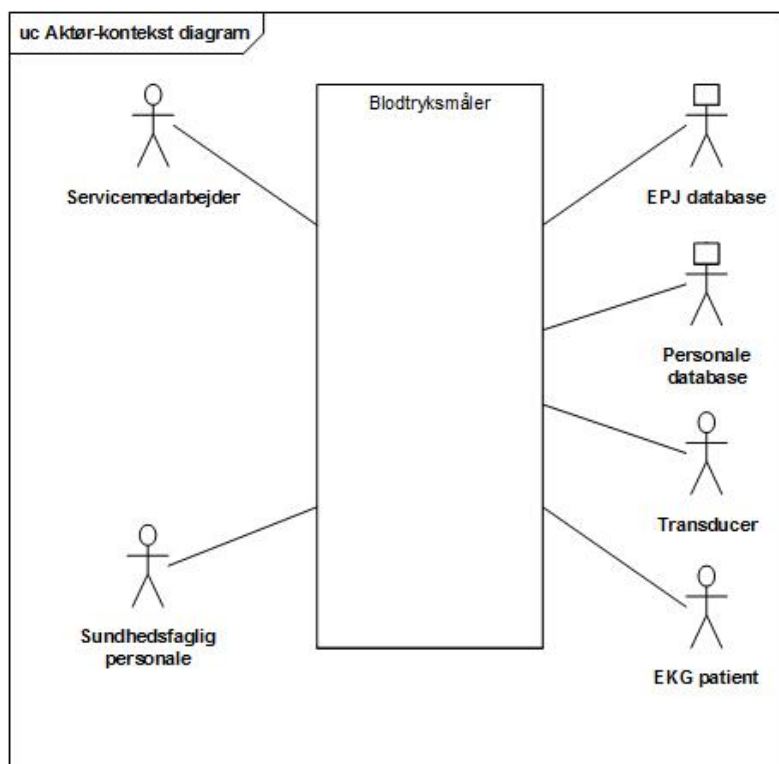
Software delen består af et window forms program udviklet i C# .net, programmet skal kunne præsentere data indhentet fra DAQ'en. Ligeledes skal programmet kunne analysere og filtrere data fra DAQ'en, samt gemme disse i en "EPJ-database. Programmet skal også kunne hente login-oplysninger fra en "personale-database.

Man har valgt at have to databaser for at simulere det at der er en adskillelse af medarbejderdata og patientdata.



Figur 2.1: Skitse af opbygning af systemet.

2.4 Aktør-kontekst diagram



Figur 2.2: Aktør-kontekst diagram

Af dette diagram ses de interagerende aktører: *Sundhedsfaglig personale*, *Transducer*, *EKG patient*, *EPJ database*, *Personale database* og *Servicemedarbejder*.

Herunder er der en detaljeret beskrivelse af hver aktør.

Navn:	Sundhedsfaglig personale
Type:	Primær aktør
Beskrivelse:	Det sundhedsfaglige personale er aktøren, der sætter måleudstyret til transduceren, samt starter målingen. Det er det sundhedsfaglige personale, der interagerer med systemet og dermed har tilgang til de viste målinger på brugergrænsefladerne (startskærm og hovedskærm).

Navn:	Transducer
Type:	Sekundær aktør
Beskrivelse:	Transduceren er kilden til måleresultaterne, og fungerer dermed som patienten og forbindelsen til systemet. I dette tilfælde er det en in-vitro maskine, der leverer et tryk til transduceren, som sender dette tryk videre til en forstærker, hvorefter signalet sendes igennem et lavpasfilter, for derefter at sendes ind i systemet via NI-DAQ.

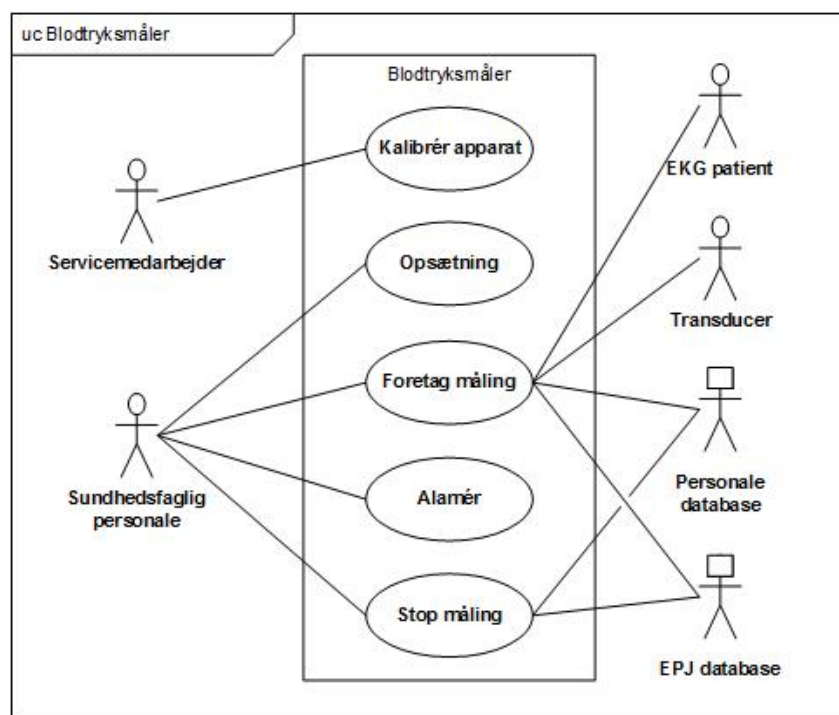
Navn:	EKG patient
Type:	Sekundær aktør
Beskrivelse:	EKG patienten er den aktør hvorfra værdierne til EKG-kurven fås fra. Dermed er det denne aktør der er kilden til pulsen. Disse værdier hentes fra PhysioBank ATM.

Navn:	Personale database
Type:	Sekundær aktør
Beskrivelse:	Personale database er der, hvori det sundhedsfaglige personales login informationer opbevares, hvilket benyttes til at tilgå systemet.

Navn:	EPJ database
Type:	Sekundær aktør
Beskrivelse:	EPJ database er den database, hvor patientdata ligger, samt der hvori analyseresultaterne, der opnås ved målingerne i systemet, samt signalerne bliver gemt. Disse data er grafer for EKG, arterietryk, samt talværdier for puls, systole, diastole og middeltryk. Denne EPJ database skal simulere den EPJ database der fungerer på sygehusene i virkeligheden.

Navn:	Servicemedarbejder
Type:	Primær aktør
Beskrivelse:	Servicemedarbejderen er aktøreren der igangsætter og foretager kalibreringen.

2.5 Use cases



Figur 2.3: Use case diagram

De fire Use cases kan ses ud fra Use case diagrammet, disse er: *Kalibrér apparat*, *Foretag måling*, *Alarmér* og *Stop måling*. Hver enkel af disse Use cases beskrives detaljeret herunder i et fully-dressed Use case skema.

Systemet består af en computer med programkode, en NI-DAQ samt hardware. Hardwaren består af et lavpasfilter, en forstærker og en transducer desuden er der hertil tilkoblet to 9 V batterier.

Systemet gør det muligt at hente data fra transducere. Her går data fra transducere igennem forstærkeren, hvilken bliver forsynet af spænding fra to 9 V batterier. Fra forstærkeren går signalet videre til lavpasfilteret og derefter ind i NI-DAQ, som så sender data videre til computeren og ind i programkoden.

Det digitale filter er pr. default slået til når systemet startes, dog vil det være muligt at slå dette filter fra under målingen.

Når programmet startes skal computeren have VPN-forbindelse til "ASE IHA VPN", desuden skal der være en forbindelse til "webhotel10.F15ST2ITS2201404669.db_owner" (Personaledatabasen) og "webhotel10.F15ST2ITS2201405838.db_owner" (EPJ database).

Programmet skal have to skærme; en startskærm, der fungerer som EPJ systemet, og en hovedskærm, der fungerer som selve blodtryksmålerens grænseflade.

Tabel 2.1: Use case 1

Use case 1	Kalibrér apparat
Mål:	Få kalibreret apparatet
Initiering:	Startes af Servicemedarbejder
Aktører:	Servicemedarbejder (primær)
Referencer:	-
Samtidige forekomster:	én kalibrering pr. apparat
Forudsætninger:	Blodtryksmålersystemet er tændt og tilsluttet kalibreringsudstyret.
Resultat:	Apparatet er kalibreret
Hovedscenarie:	<i>[Undtagelse 1: Undlad kalibrering]</i> 1. Servicemedarbejder foretager kalibreringen, ved at indtaste spændingen og trykket for målepunktet fra væskesøjlen. 2. Systemet beregner kalibreringsværdien.
Udvidelse/undtagelser:	<i>[Undtagelse 1: Undlad kalibrering]</i> 1.1 Undlad kalibrering 1.2 Fortsæt til Use case 2. 1.3 Kalibreringsværdien hentes fra den forrige måling.

Tabel 2.2: Use case 2

Use case 2	Foretag måling
Mål:	Den valgte patients målinger foretages og værdierne gemmes i EPJ database
Initiering:	Startes af Sundhedsfaglig personale
Aktører:	Sundhedsfaglig personale (primær), Personale database (sekundær), EPJ database(sekundær), Transducer (sekundær), EKG patient (sekundær)
Referencer:	Use case 2
Samtidige forekomster:	En sundhedsfaglig person og én transducer pr. system
Forudsætninger:	VPN, Personale database og EPJ databasen er tilsluttet korrekt
Resultat:	Målingerne for den valgte patient er foretaget
Hovedscenarie:	<p>1. Sundhedsfaglig personale logger på ved at indtaste brugernavn og kode. <i>[Undtagelse 1: Brugernavn og/eller kode indtastet forkert]</i></p> <p>2. Besked: "Logget på" vises</p> <p>3. Liste med patienter kommer frem</p> <p>4. Den ønskede patient vælges</p> <p>5. Sundhedsfaglig personale indtaster nulpunktjusterings værdien og starter nulpunktsjusteringen.</p> <p>6. Systemet foretager nulpunktsjusteringen.</p> <p>7. Sundhedsfaglig personale starter målingen</p> <p>8. Systemet indhenter data fra transduceren og måler, hvor lang tid målingen foretages</p> <p>9. EKG og arteriestryk præsenteres kontinuert på hver sin graf. Puls, systole, diastole og middeltryk vises som talværdier. Data gemmes automatisk kontinuert i EPJ database. <i>[Udvidelse 1: Slå digitalt filter til/fra]</i> <i>[Udvidelse 2: Juster systolens/diastolens grænseværdi]</i></p>
Udvidelse/undtagelser:	<p><i>[Undtagelse 1: Brugernavn og/eller kode indtastet forkert]</i></p> <p>1.1 Besked: "Brugernavn og/eller kode indtastet forkert"</p> <p>1.2 Use case 3 starter forfra</p> <p><i>[Udvidelse 1: Slå digitalt filter til/fra]</i></p> <p>1.1 Sundhedsfaglig personale vælger "Digitalt filter OFF"</p> <p>1.2 Systemet slår det digitale filter fra</p> <p>1.3 Sundhedsfaglig personale vælger "Digitalt filter ON"</p> <p>1.4 Systemet slår det digitale filter til</p> <p><i>[Udvidelse 2: Juster systolens/diastolens grænseværdi]</i></p> <p>2.1 Sundhedsfaglig personale justerer grænseværdierne for systole og/eller diastole.</p>

Tabel 2.3: Use case 3

Use case 3	Alarmér
Mål:	Få startet alarmeringen ved overskridelse af en grænseværdi
Initiering:	Systemet starter denne Use case
Aktører:	Sundhedsfaglig personale (sekundær)
Referencer:	Use case 2
Samtidige forekomster:	-
Forudsætninger:	Målingen i Use case 2: Foretag måling, er kørt succesfuldt
Resultat:	Alarmen starter
Hovedscenarie:	1. Grænseværdi overskrides 2. Alarmering starter. <i>[Udvidelse 1: Anden grænseværdi overskrides]</i> <i>[Udvidelse 2: Udsæt alarm]</i> 3. Alarmen stopper når grænseværdien ikke længere er overskredet.
Udvidelse/undtagelser:	<i>[Udvidelse 1: Anden grænseværdi overskrides]</i> 1.1. Endnu en grænseværdi overskrides 1.2. Første alarm fortsætter, og den nye alarmering igangsætter. 1.3 Use case 3 fortsætter fra punkt 3. <i>[Udvidelse 2: Udsæt alarm]</i> 2.1 Sundhedsfaglig person udsætter alarm 2.2 Systemet stopper alarmens lyd i et minut 2.3 Use case 3 fortsætter ved punkt 3.

Tabel 2.4: Use case 4

Use case 4	Stop måling
Mål:	Få stoppet målingen og logget ud
Initiering:	Startes af Sundhedsfaglig personale
Aktører:	Sundhedsfaglig personale (primær)
Referencer:	Use case 2
Samtidige forekomster:	-
Forudsætninger:	Use case 2: Foretag måling, er kørt succesfuldt
Resultat:	Signalet er stoppet, sundhedsfaglig personale er logget ud og vendt tilbage til startskærm.
Hovedscenarie:	1. Sundhedsfaglig personale stopper måling 2. Systemet stopper målingen. 3. Sundhedsfaglig personale logger ud
Udvidelse/undtagelser:	-

2.6 Ikke-funktionelle krav

De ikke-funktionelle krav er opsat efter FURPS+ metoden. De er prioriteret efter MoSCoW metoden:

- **Must** (skal være med)
 - De krav der dermed bliver markeret med et **(M)**, er altså de krav til funktioner der skal være til produktet.
- **Should** (bør være med, hvis muligt)
 - Kravene kan også markeres med et **(S)**. Disse krav er funktioner produktet bør have.
- **Could** (kunne have med, hvis det ikke går i vejen for noget andet)
 - Markeret et krav med **(C)**, behøver produktet ikke at have funktionen, men det kunne være en funktion der kunne være god at have til produktet.
- **Won't/Would** (tager det ikke med nu, men kan komme med i fremtidige opdateringer)
 - Dermed er de krav der bliver markeret med et **(W)**, altså de krav til funktioner der ville være gode til produktet, men ikke bliver implementeret i produktet. Grunden til dette kan være at der ikke er tid, eller at funktionen ikke er vigtig for produktet.

FURPS+ med MoSCoW

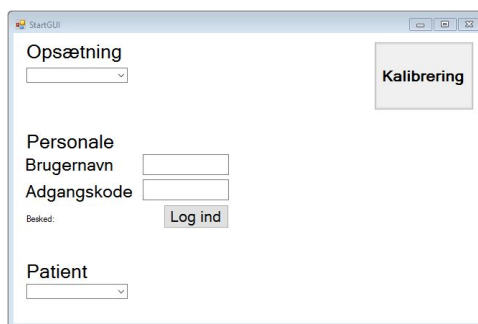
1. Functionality

- 1.1. **(M)** Programmet skal have et digitalt filter til udglatning af blodtrykssignal.
- 1.2. **(S)** Programmet skal give alarm, når grænseværdier for blodtrykket overskrides, med lyd og "Udsæt alarm"knappen blinker skiftevis mellem rød og hvid.
- 1.3. **(M)** Programmet skal kunne gemme blodtrykssignalet i en database.

2. Usability

- 2.1. **(S)** Programmet skal have to window forms: startskærm, der fungerer som EPJ systemet og hovedskærm, der fungerer som selve blodtryksmålerens grænseflade.
- 2.2. **(M)** Programmet skal have en "Log ind"knap på startskærmen.
- 2.3. **(M)** Programmet skal have en "Kalibrering"knap på startskærmen.
- 2.4. **(M)** Sundhedsfagligt personale skal kunne ændre "devicename/enhedsnavn"i dropdown på startskærmen.
- 2.5. **(S)** Programmet skal indeholde en dropdown, hvor patienten kan vælges, på startskærmen.
- 2.6. **(M)** Programmet skal have en "Nulpunkts indstilling"knap på hovedskærmen.
- 2.7. **(M)** Programmet skal have en knap til at slå det digitale filter fra og til på hovedskærmen.
- 2.8. **(S)** Programmet skal have knapper til at justere systolisk og diastolisk grænseværdiintervaller op og ned, på hovedskærmen.
- 2.9. **(S)** Programmet skal have en "Udsæt alarm"knap på hovedskærmen.

- 2.10. (M) Programmet skal have en "Tænd"knapp på hovedskærmen.
- 2.11. (M) Programmet skal have en "Sluk"knapp på hovedskærmen.
- 2.12. (M) Programmet skal have en "Log ud"knapp på hovedskærmen.
- 2.13. (M) Teksten på startskærmen skal kunne læses fra 2 meters afstand ved synsstyrke i intervallet på ± 1 .
- 2.14. (M) Teksten og graferne på hovedskærmen skal kunne læses fra 2 meters afstand ved synsstyrke i intervallet på ± 1 .
- 2.15. (M) Programmet skal præsentere arterietryk kontinuert, herudover vise systolisk værdi, diastolisk værdi og middeltryk som talværdier.
- 2.16. (S) Programmet skal præsentere EKG og puls.
- 2.17. (W) Programmet skal præsentere iltmætning både som graf og talværdi.
- 2.18. (M) Programmet skal præsentere data på grafer på følgende måde (Se afsnit nedenfor).
 - EKG vises i lysegrøn.
 - Arterietryk vises i rød.
 - Iltmætning/saturation i lyseblå.
- 2.19. (M) Programmet skal præsentere data i tal på følgende måde (Se afsnit nedenfor)
 - Hjerterefrekvens (puls) i lysegrøn.
 - Systolisk samt diastolisk tryk i rødt, ligeledes middelblodtrykket i parentes ved siden af i rødt.
 - Iltmætningsværdien i lyseblå.



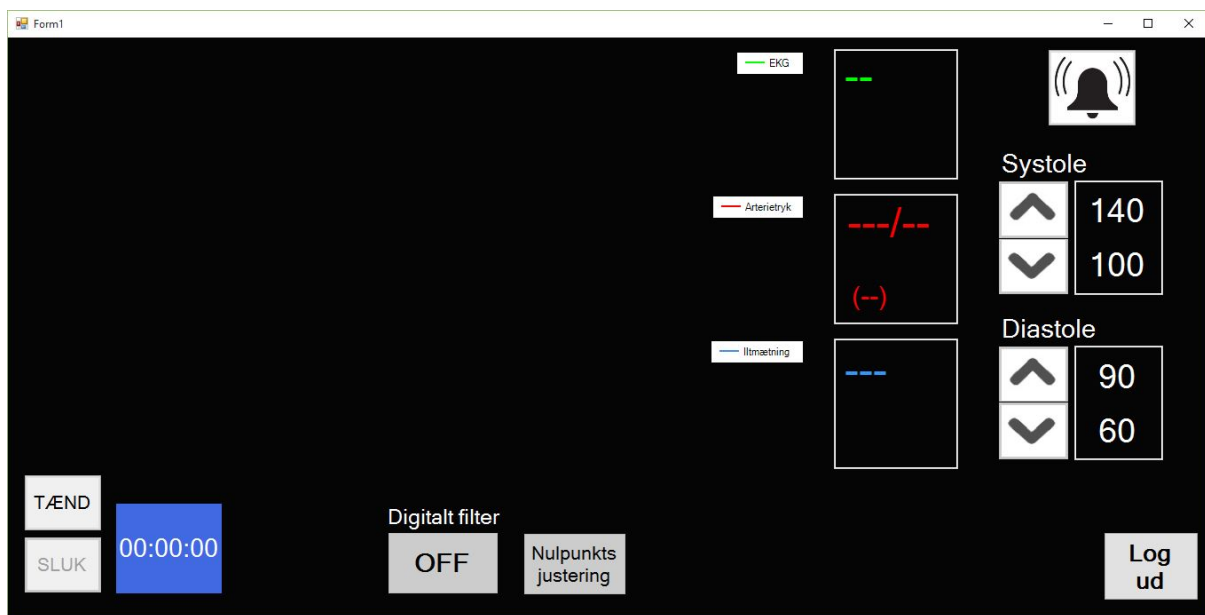
Figur 2.4: Skitse af startskærmen, som repræsenterer EPJ systemet

3. Reliability

3.1. (S) INGEN RELIABILITY KRAV ENDNU

4. Performance

- 4.1. (S) Tiden der går før måling af data påbegynder/vises i grafer må maksimalt være 2 sek.
- 4.2. (S) Tiden der går fra at data, herunder puls, diastolisk tryk, systolisk tryk og middeltryk, er analyseret til at data er gemt i EPJ database må være 2 sek. med en tolerance på $\pm 15\%$



Figur 2.5: Skitse af hovedskærmen, hvilken repræsenterer en blodtryksmålers brugergrænseflade

- 4.3. (S) Ved justering af grænseværdi for systole og diastole ændres grænseværdien 1 mmHg op eller ned.

5. Supportability

- 5.1. (M) Programmet skal skrives i C# kode
 5.2. (M) Softwaren skal være opbygget efter trelagsmodellen (Data-View-Model)
 5.3. (S) I softwaren benyttes Observer/Subject mønsteret.
 5.4. (S) I softwaren benyttes PUSH mønsteret

6. + Test conditions

- 6.1. (M) Der skal være adgang til en computer med Windows 7, 8 eller 10 - computeren skal minimum have 4 GB RAM.
 6.2. (M) Der skal være adgang til en computer hvor National Instruments er installeret.

Kapitel 3

Arkitektur og design

Følgende beskriver arkitekturen for systemet herunder både hardware og software. Systemarkitektur er udviklingsrammen for den videre udvikling af design og implementering. Der vil igennem dette afsnit startes med at se systemet overordnet og hvorefter der arbejdes ned gennem systemet i mindre brudstykker. Der benyttes diagrammer for at kunne specificere og klarlægge systemkravene. Disse diagrammer beskrives desuden i tekst. Igennem dette afsnit bliver designet af produktet dermed bestemt.

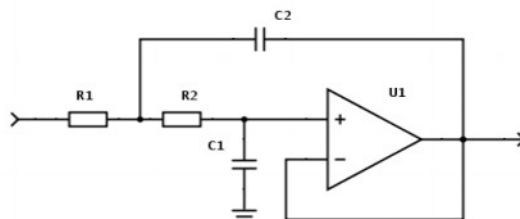
3.1 Hardware design

Udviklingsfase og iterationer

I dette afsnit redegøres der for udviklingsprocessen på hardware-teamet, som har bestået af tre mand. Der redegøres for de løsnings-iterationer gruppen har været igennem, samt gruppens tankeproces mens der er arbejdet på projektet. Udviklingen af hardwaren til projektet har foregået som et teknologistudie baseret på krav til projektet og tidligere viden fra Analog Signal Behandling. Processen har været en iterativ og agil udviklingsproces, som er præget af en generel mangelfuld viden på området.

Første iteration - komponenter i lavpasfilteret

Der blev besluttet, at tage udgangspunkt i beregningsværktøjer på nettet [4] til den første iteration. Dette gav nogle værdier der lå meget tæt på det ønskede resultat. Kredsløbet blev derefter bygget op på et fumlebræt og fastsatte komponent værdierne ud fra beregningerne fra nettet. Udregningen fra nettet var baseret på en cut-off frekvens på 50Hz og $C1 = C2$.



Figur 3.1: Butterworth Sallen-key filter.

Komponent	Værdi
R1	4.7 k Ω
R2	4.7 k Ω
C1	680 nF
C2	680 nF

Det viste sig at det ikke levede op til kravene omkring den nødvendige dæmpning ved 500Hz, ligeledes opførte filteret sig ikke som ønsket. I starten var der blandt andet arbejdet med filteret som passivt, noget som der først fik rettet op på til sidste iteration. Desuden fik hjælpen fra en ASB-underviser, vist at ovenstående filter ikke overholdte kravene til et 2. ordens Butterworth Sallen-key filter.

Anden iteration - komponenter i lavpasfilteret

Den anden iteration er beskrevet udførligt i hardware delen, her blev der brugt en mere matematisk proces til at designe filteret. Der blev gættet på en C1 værdi på 330nF, og herefter blev de resterende komponentværdier skaleret ud fra dette. Neden for ses de udregnede komponentværdier for lavpasfilteret version 2.

Komponent	Værdi
R1	6.7 k Ω
R2	6.7 k Ω
C1	330 nF
C2	680 nF

Filteret blev analyseret. Analysen viste, at der ikke kom den nødvendige dæmpning ved 500Hz. Igen kan dette være et resultat af det ikke var aktivt.

Tredje iteration - komponenter lavpasfilteret, forstærker og forsyningsspænding

I den tredje iteration blev der opdaget at C1 skulle være halvdelen af C2 for at give et filter der overholdte projektets krav. Modstandene er beregnet ud fra den nye kondensatorværdi.

Komponent	Værdi
R1	6.6 k Ω
R2	6.6 k Ω
C1	340 nF
C2	680 nF

Der blev besluttet at benytte en INA-114 forstærker, efter anbefaling fra vejleder og medstuderende. Der blev beregnet forstærkningen, samt blev der sikret at forstærkeren opfyldte projektets krav. Beslutningen om hvilken forsyningsspænding der skulle benyttes, faldt på to 9V-batterier. Det blev undersøgt hvorvidt det var muligt at bruge andre spændingsforsyninger, blandt andet USB og dedikerede spændingsforsyninger. USB som spændingsforsyning blev fravalgt da der ikke kunne opnå tilstrækkelig forstærkning i INA-114 forstærkeren, så det ville gå ud over præcisionen. De udleveret spændingsforsyninger blev fravalgt af praktiske årsager, da de var meget uhåndterbare.

Fjerde iteration - aktivt filter

Den fjerde iteration benytter samme komponentværdier som tredje iteration, men det var blevet overset at filteret skulle være aktivt. Derfor blev der bygget et kredsløb op forfra på fumlebræt, hvor der var tilsluttet spænding på filteret.

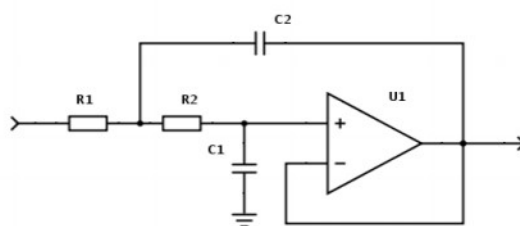
Design af lavpasfilteret

Frekvenserne der skal arbejdes med i blodtryksmåleren, ligger op til 50Hz, derfor skal der realiseres et 2. ordens lavpasfilter med følgende dimensioner:

Cut-off Frequency: 50Hz

Ligeledes skal der dæmpes med 40 dB/decade ved 500Hz.

C2 er givet til at være på 680nF, ligeledes er operationsforstærkeren givet til at være OP27.



Figur 3.2: Butterworth Sallen-key filter.

Overføringsfunktionen for ovenstående filter:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{s^2 + \frac{R1 + R2}{R1 \cdot R2 \cdot C2} \cdot s + \frac{1}{R1 \cdot R2 \cdot C1 \cdot C2}} \quad (3.1)$$

Omskrevet til standardformel:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{\omega^2}{s^2 + 2\zeta\omega + \omega^2} \quad (3.2)$$

Her isoleres cut-off frekvensen (ω) i overføringsfunktionen for lavpasfilteret:

$$\omega = 2\pi\sqrt{\frac{1}{R1 \cdot R2 \cdot C1 \cdot C2}} \quad (3.3)$$

I formelen for cut-off frekvensen indsættes komponentværdierne. Der er valgt en værdi for $C1$ til at være 330nF, dette er et gæt. Cut-off frekvensen er sat til 50Hz, da dette er blevet givet som et krav til projektet.

$$C1 = 330nF, C2 = 680nF \text{ og } R1 = R2$$

$$\begin{aligned}
 C1 &:= 330 \cdot 10^{-9} & C2 &:= 680 \cdot 10^{-9} \\
 50 \cdot 2\pi &= \sqrt{\frac{1}{R \cdot R \cdot C1 \cdot C2}} \text{ solve, R} \rightarrow \begin{pmatrix} \frac{500000 \cdot \sqrt{561}}{561 \cdot \pi} \\ -\frac{500000 \cdot \sqrt{561}}{561 \cdot \pi} \end{pmatrix} \\
 \frac{500000 \cdot \sqrt{561}}{561 \cdot \pi} &= 6.72 \times 10^3
 \end{aligned}$$

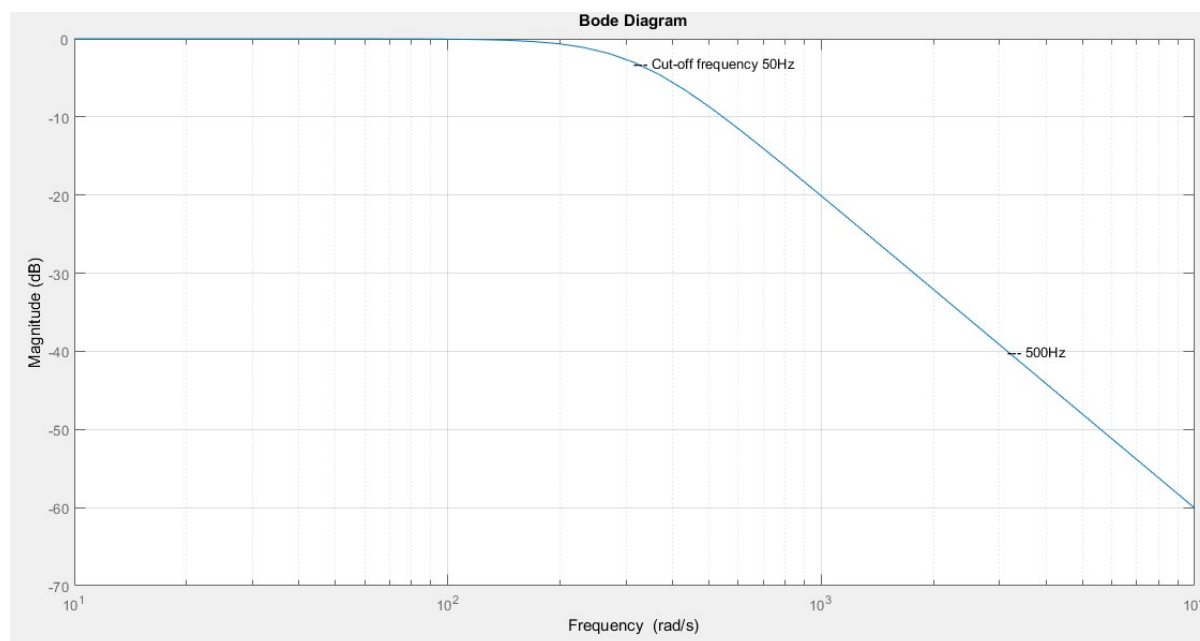
Figur 3.3: Mathcad beregninger.

Ud fra disse beregninger skal modstanden altså bestemmes til at have en værdi på ca. $6.7k\Omega$. Da der er blevet brugt et Butterworth Sallen-key filter vides der, at $C1$ skal være halvdelen af $C2$. Derfor er der lavet nye beregninger hvor $C1 = 340nF$, hvilket svarer til halvdelen af $C2$.

$$\begin{aligned}
 C1 &:= 340 \cdot 10^{-9} & C2 &:= 680 \cdot 10^{-9} \\
 50 \cdot 2\pi &= \sqrt{\frac{1}{R \cdot R \cdot C1 \cdot C2}} \text{ solve, R} \rightarrow \begin{pmatrix} \frac{250000 \cdot \sqrt{2}}{17 \cdot \pi} \\ -\frac{250000 \cdot \sqrt{2}}{17 \cdot \pi} \end{pmatrix} \\
 \frac{250000 \cdot \sqrt{2}}{17 \cdot \pi} &= 6.62 \times 10^3
 \end{aligned}$$

Figur 3.4: Mathcad beregninger.

Der er efterfølgende lavet en analyse i matlab for at sikre at dæmpningen er min. 40 dB pr. decade ved 500Hz. Dette er gjort med modstande på $6.6k\Omega$.

Figur 3.5: Matlab bodeplot med $6.6k\Omega$ modstande.

Ud fra dette kan det konkluderes at dæmpningen ved 550Hz er tilstrækkelig.

Design af operationsforstærkeren

Det elektriske signal der skal bruges i systemet, kommer fra tryktransducere TruWaveTM og ind i dataopsamlingsmodel (NI-DAQ6009). Signalet fra TruWaveTM skal forstærkes op, så der kommer bedre målesignaler i DAQ'en. I databladet for NI-DAQ6009 kan der ses den maksimale spænding for indgangsportene er $\pm 10V$, det vil sige at der ikke må forstærkes mere op end 10V. Der er valgt at forstærke op til $\pm 8V$, for at give lidt buffer, så der ikke overstiges de 10V og derved mister data. Ligeledes er "projektet" hæmmet af valget af strømforsyning som er to 9-volts batterier, som praktisk leverer omkring $\pm 8V$.

Fra TruWaveTM er der valgt at fokusere på et måleområde der hedder 0-250mmHg.

Maksimalt output for transducer

$$V_{max} = 250mmHg \cdot 16V \cdot 5\mu V \quad (3.4)$$

Ud fra dette kan der bestemmes, hvor meget gain (forstærkning), der skal bruges fra forstærkeren:

$$16V = 20 \cdot 10^{-3}V \cdot gain \quad (3.5)$$

$$gain = \frac{16V}{20 \cdot 10^{-3}V} = 800 \quad (3.6)$$

Båndbredde og valg af forstærker

Produktet af gain og båndbredde, BW , er konstant, derfor er det vigtigt vores båndbredde ligger over knæfrekvensen på 50Hz. Der er valgt at benytte en INA-114 forstærker da den opfylder behovene. Ved $gain = 1$ kan INA-114 levere 1MHz båndbredde, det vil sige, at der kan

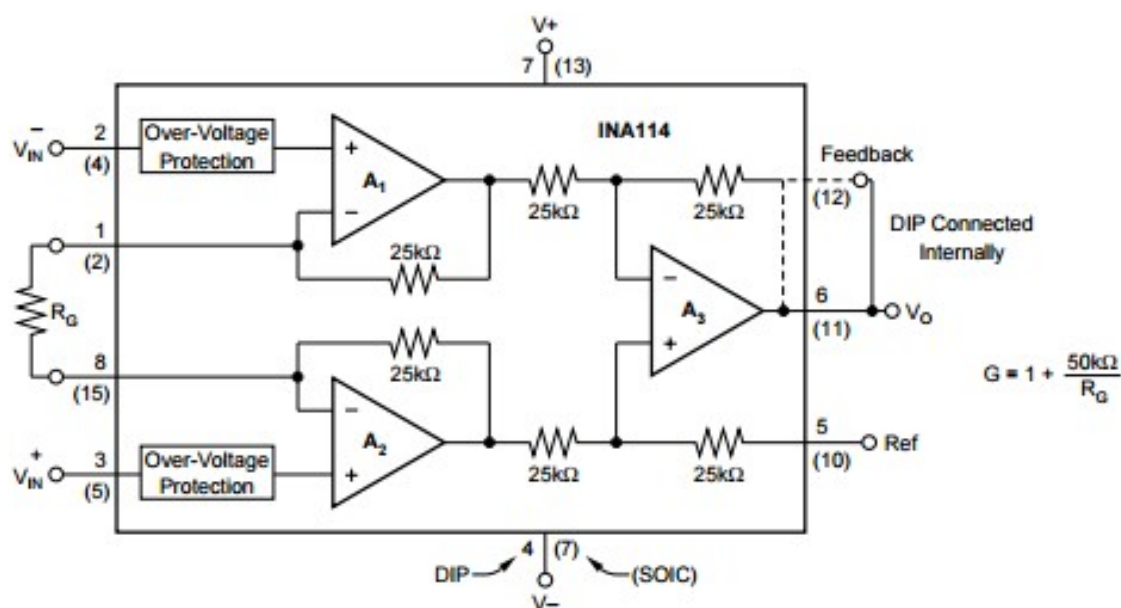
opstilles følgende ligning for beregning af båndbredden.

$$1000000 = 800 \cdot BW \quad (3.7)$$

$$BW = \frac{1000000 \text{ Hz}}{800} = 1250 \text{ Hz} \quad (3.8)$$

Da 1250 Hz er over knækfrekvensen på 50Hz har forstærkeren tilstrækkelig båndbredde til at leve op til kravene til projektet.

Beregning af modstand til forstærker



Figur 3.6: Denne viser opbygningen og delene i operationsforstærkeren INA114

Forstærkerens gain er bestemt ved modstanden der sidder på port 1 og 8. Denne modstand kan beregnes med følgende formel, hvor R_G er modstanden og gain = 800.

$$G = 1 + \frac{50k\Omega}{R_G} \quad (3.9)$$

$$800 = 1 + \frac{50k\Omega}{R_G} \quad (3.10)$$

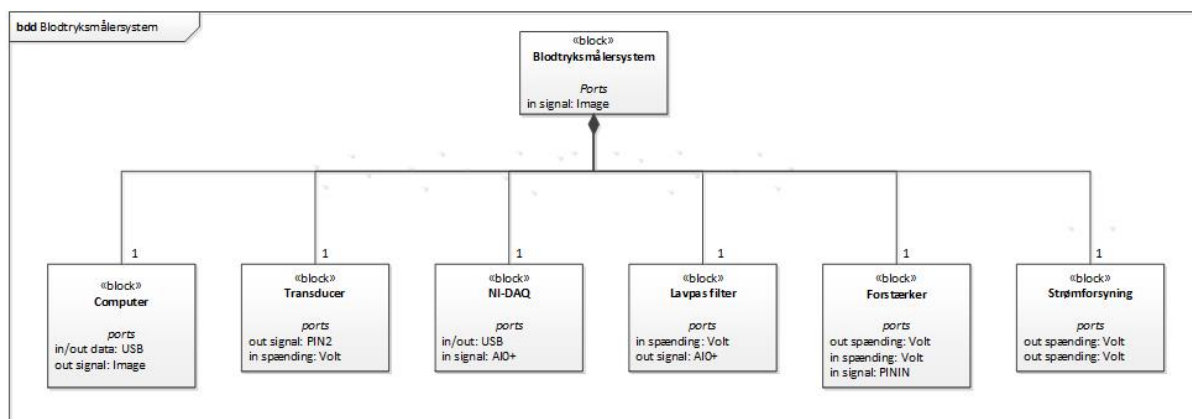
$$R_G = 62.5\Omega \quad (3.11)$$

Et potentiometer er blevet valgt at benytte som denne modstand, da der helst skulle kunne reguleres på systemet, og den ovenstående værdi er beregnet med ideelle komponenter.

Implementering

Block definition diagram

På nedenstående figur bliver systemets hardware illustreret i et BDD. Heraf ses det at systemet har seks hardware blokke: Computer, transducer, NI-DAQ, lavpas filter, forstærker og en strømforsyning. Disse blokke udgør tilsammen blodtryksmålersystemet.

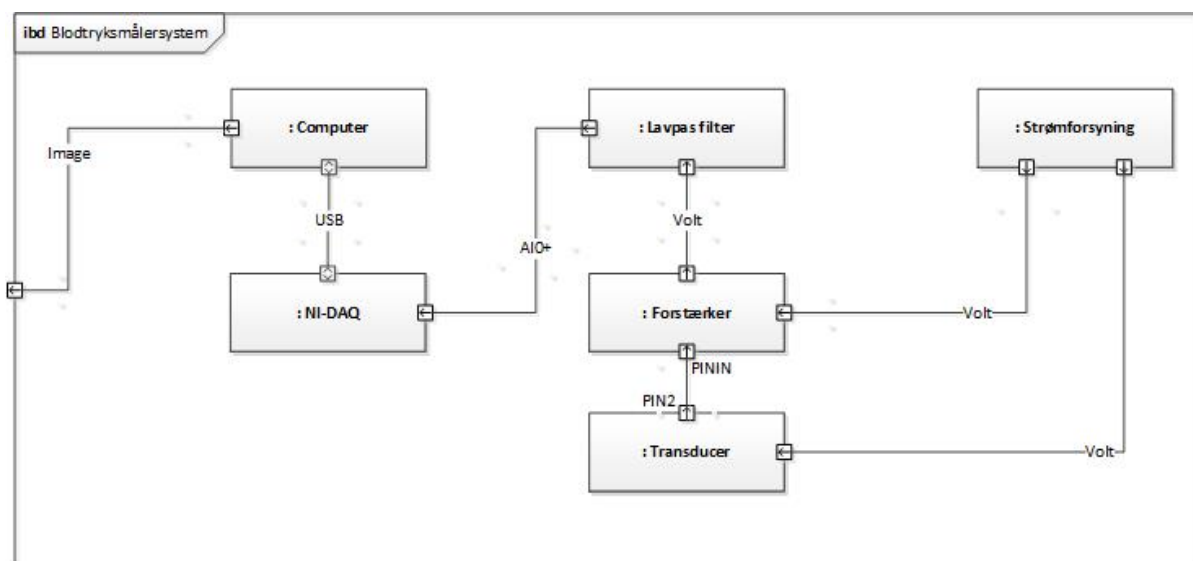


Figur 3.7: **Block definition diagram.** Dette diagram viser hardware delene i systemet, samt sammenhængen mellem disse.

Internal block definition

Ud fra BDD kan et IBD diagram udarbejdes. IBD diagrammet viser koblingen mellem blokkene:

- Strømforsyningen, denne er to 9V batterier som forsyner forstærkeren og transduceren med $\pm 9V$
- Transduceren omdanner tryksignalet fra kateteret til et strømsignal i ZZ mV tilbage til forstærkeren.
- Forstærkeren forstærker signalet til lavpas filteret.
- Lavpas filteret dæmper de høje frekvenser og sender signalet til NI-DAQ
- NI-DAQs formål er, at omdanne signalet fra et analogt signal til et digitalt signal.
- Computerens funktion er at få analyseret og vist blodtryksignalet på en brugergrænseflade, vha. Visual Studio.

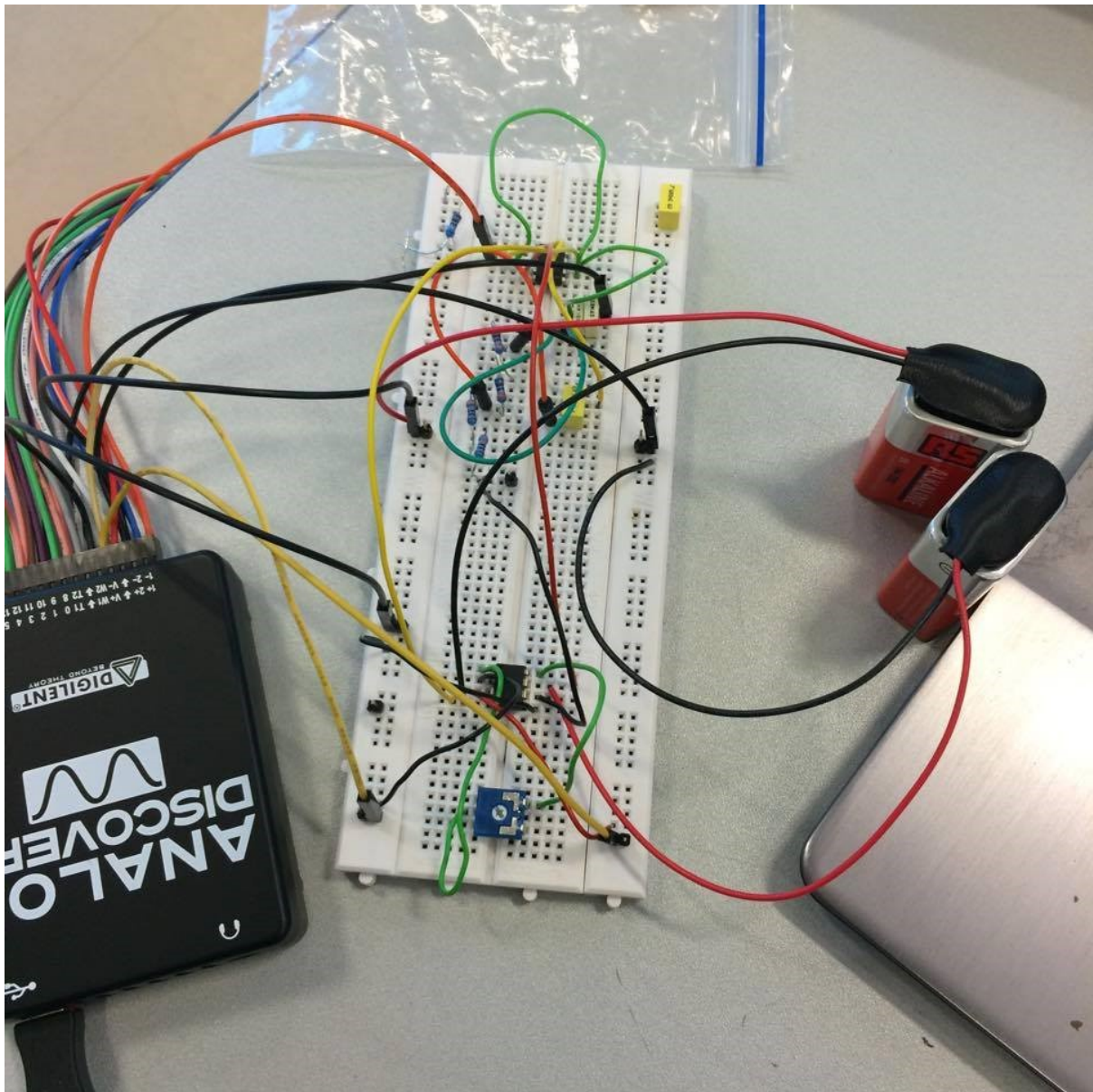


Figur 3.8: **Internal block diagram.** Dette diagram viser signalerne imellem blokkene.

Modultest

Modultesten omfatter kun en operationsforstærker. De resterende komponenter, som er anvendt i den samlede hardwaressystem, tages ikke i modultesten, da de er udleverede af vejlederen. De udleverede dele af systemet består af et filter, en transducer, et kateter og en dataopsamler i form af NI-USB-6009. Selvom filtret er udleverede, indgår den i modultesten, da man skal sikre at det dæmper de høje frekvenser. Modultesten foretages enkeltvis dvs. de enkelte blokke testes hver for sig. Til sidst testes systemet med en vandsøjle for at identificere om operationsforstærkeren og filteret kommunikere med hinanden. Rækkefølgen som blokkene skal testes er som følgende:

- Test af operationsforstærker.
- Test af filter.
- Den enelige test med vandsøjle.

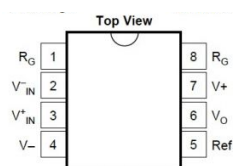


Figur 3.9: Opstilling af hardware systemet simuleret med Analog Discovery.

Test af operationsforstærker

Operationsforstærkerens funktion er at forstærke et lavt signal til et ønsket højt signal. Der er brugt INA-114 som operationsforstærker, da det er muligt at få den ønskede forstærkning ved den ønskede båndbredde. Der er i alt 8 pin og det forklares som følgende:

Pin 1 = R_G	Pin 5 = Ref
Pin 2 = V_{in}^-	Pin 6 = V_0
Pin 3 = V_{in}^+	Pin 7 = V^+
Pin 4 = V_-	Pin 8 = R_G



Figur 3.10: Op. Amp. INA-114

Vha. Analog Discovery er der påført INA-114 et differentielt signal dvs. to signaler med hver 10mV, hvor der varieres i frekvensen fra 1 til 500Hz. Det forventet resultat er at de 20mV bliver forstærket. De to signaler har samme stelpunkt. Outputtet af signalet måles vha. et oscilloskop. Resultatet af denne simulering er som følgende:

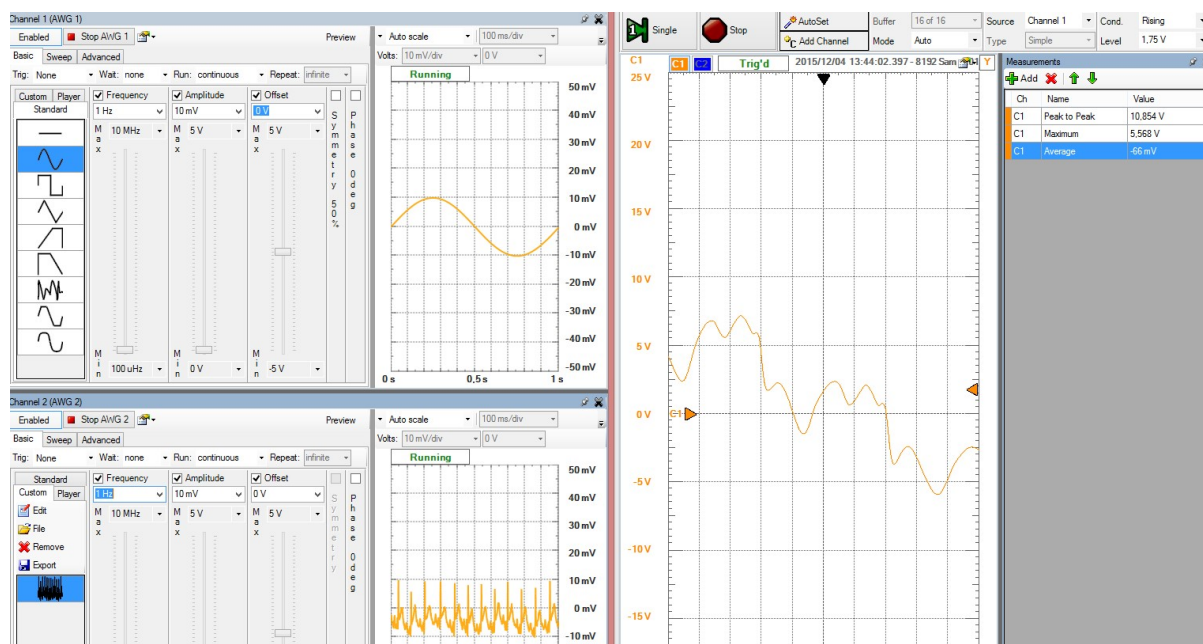
Frekvens [Hz]	Forstærket [V]. Peak to peak.
1	16,308
10	17,512
25	17,558
49	17,512
70	17,258
100	17,308

Test af filter

Systemets filtre er et 2. ordens lavpasfilter af typen Sallen-Key og designes som et Butterworth med cut off frekvens på 50Hz. Om filtret lever op til de opstillet krav kan testes ved at sende signaler med høje frekvenser igennem. De høje frekvenser skal blive dæmpet. Der er brugt Analog Discovery til at generer et signal med forskellige frekvenser. Da cut off frekvensen ligger på 50Hz, forventes der at alt over 50 bliver dæmpet og ved 500 skal der dæmpes minimum 40 dB pr. dekade.

Frekvens [Hz]	Spænding [V]
1	13,504
10	13,220
25	12,412
49	10,644
70	6,658
100	3,662
250	0,666
500	0,220

Nedstående figur viser et meget lille signal input på 20mV. De 20 mV bliver herefter forstærket til 10,854V. Dette tal er ikke en 11 volts forstærkning, men peak to peak forstærkning.

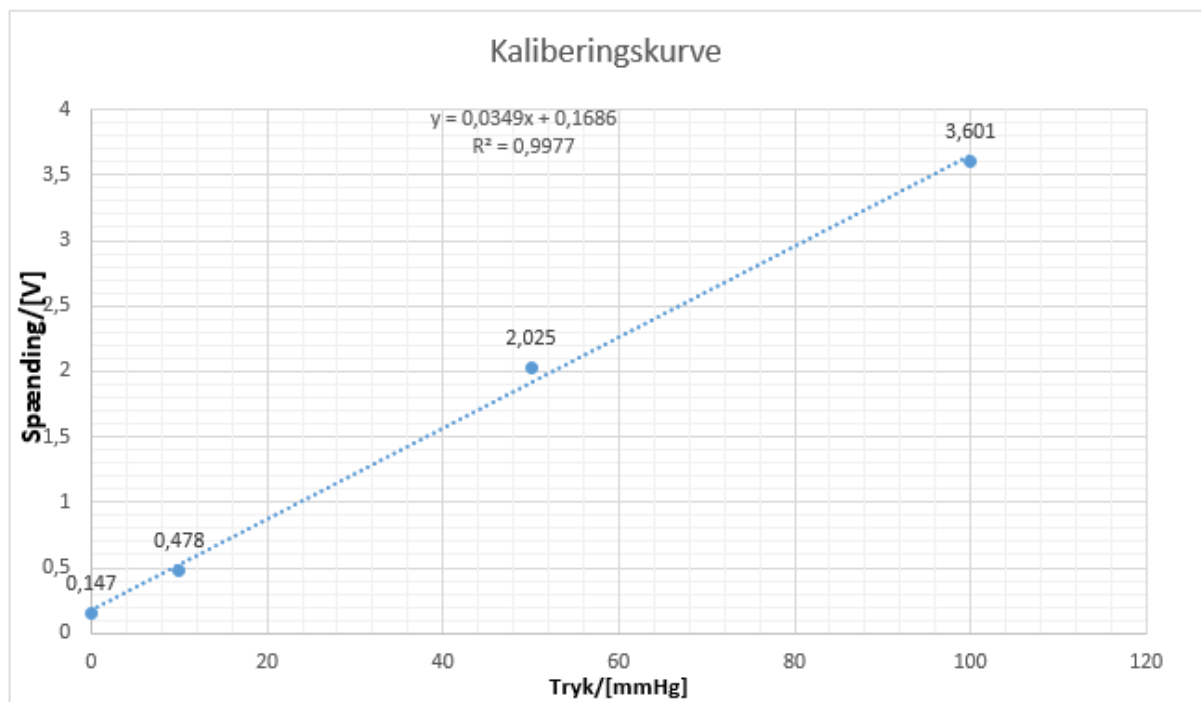


Figur 3.11: Output af lavpasfilteret ved 1 Hz.

Test med vandsøjle

Den endelig måling er foretaget vha. vandsøjlen som signalgenerator i form af tryk til transduceren. Det genererede signal skal herefter igennem forstærkeren, hvor der forventes at operationsforstærkeren forstærker signalet. Tabellen viser trykket fra vandsøjlen til transduceren og det forstærket tryk i V. Nedstående målinger er det der er kalibreret efter. Hældning af den nedstående kurve er kalibreringskoefficienten i V/mmHg. For at lave det om til mmHg/V skal hældning ganges med 30,21 ved fx tryk på 10mmHg.

Tryk [mmHg]	Spænding [V]
0	0,147
10	0,478
50	2,025
100	3,601



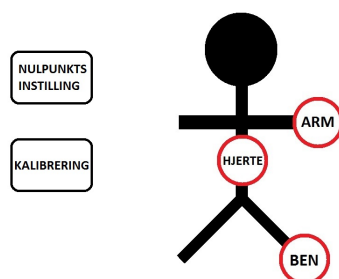
Figur 3.12: Sammenhængen mellem tryk og spænding.

3.2 Software design

I dette afsnit beskrives softwaredesign på baggrund af systembeskrivelsen og kravspecifikationen. Denne beskrivelse opnås ved, at der benyttes relevante diagrammer og modeller, hvilke kan bruges til at beskrive softwaren. Overvejelser og valg, der er blevet gjort i forbindelse med design og implementering af softwaren, vil i dette afsnit blive præsenteret.

Problemidentifikation

Ved identifikation af produktet, og herunder hvad dette skal kunne, er der opstået nogle vanskeligheder, som der har skullet tages hensyn til. I første omgang blev der udarbejdet en idé, om at have en startskærm, hvor det kunne vælges hvorpå målingen skulle foretages. På startskærmen skulle det desuden være muligt at kalibrere systemet og foretage nulpunkts justeringen. Stederne hvor målingerne skulle foretages blev identificeret til at skulle være tre målesteder; hjertet, armen og benet. Efter målestedet var valgt, skulle man komme videre til en anden skærm, hvor selve målingen skulle foretages. Idé med denne startskærm viste sig dog ikke at være brugbar i praksis, hvilket fik os til at gå væk fra denne. Grunden til at denne idé ikke ville kunne bruges i praksis, ved en invasiv blodtryksmåling, var at denne idé var tænkt som et produkt til måling af blodtrykket på diabetes patienternes underekstremiteter. Herefter ville værdierne kunne sammenlignes, og situationen vurderes for patienten. Men ved denne patientgruppe er det dog ikke hensigtsmæssigt at lave invasiv blodtryksmåling på underekstremiteterne, da blodtrykket her vil være lavt. Ved invasiv blodtryksmålingen laves der hul på karrene og helingen af dette hul vil ikke kunne foregå optimalt og der vil være risiko for infektioner.



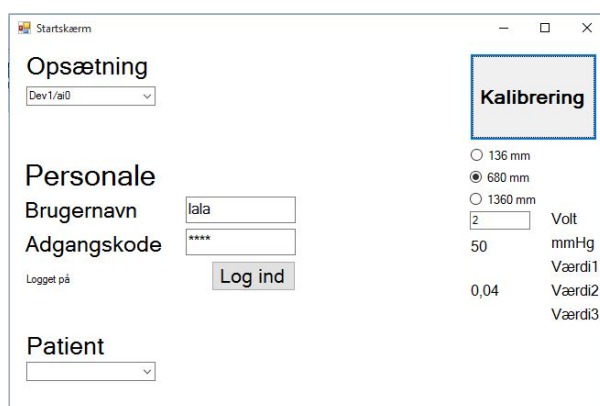
Figur 3.13: Idé til projekt. Startskærmen hvor det vil være muligt at vælge målested.

Herefter blev idéen ændret til et produkt, der ligner det der fungerer i dagligdagen på hospitalerne. Denne idé blev udarbejdet efter et besøg på Herning Sygehus, hvor et møde med anæstesi sygeplejerske Charlotte Høj blev afholdt. Her blev det muligt at se et scenarie med en blodtryksmåling. Ved scenariet loggede Charlotte Høj først ind på EPJ computeren, hvorefter selve blodtryksmålingen kunne foretages fra en anden skærm.

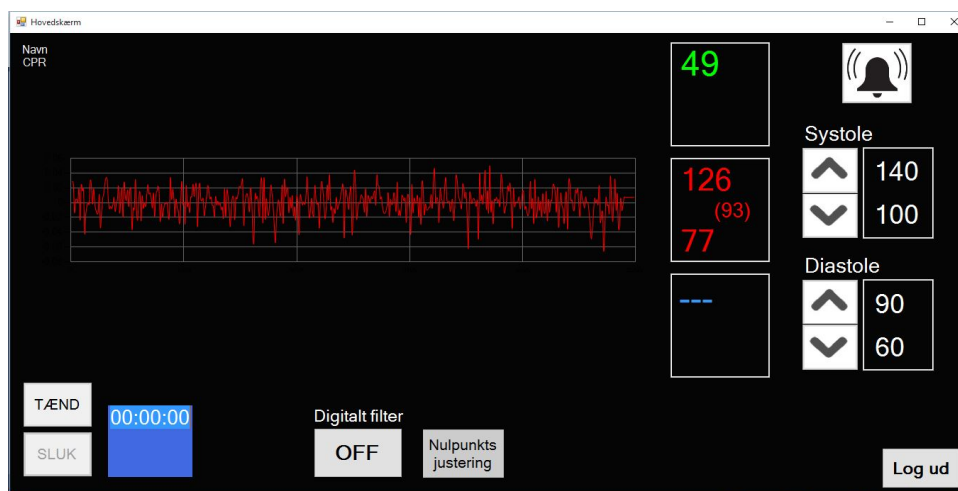
Ud fra denne idé blev en startskærm, som skal simulere EPJ-systemet (EPJ computeren), udarbejdet. EPJ-systemet er der, hvor det sundhedsfaglige personale kan tilgå patientens data, og dermed evt. se hvilket behandling patienten har været igennem før. Startskærmen der her er udarbejdet, er blot en prototype af dette og derfor er det kun den del, hvor det sundhedsfaglige personale logger på, samt vælger patient, der er blevet implementeret. Dermed er det ikke muligt at tilgå tidligere behandlinger osv. Desuden blev nulpunkts justeringen, hvilken hertil var placeret på startskærmen, flyttet ind på hovedskærmen, altså skærmen, hvor selve målingen foretages. Dette blev denne, da der var en antagelse om at der skulle sendes et signal igennem

systemet før denne kunne foretages. Dog er det senere blevet klart at dette ikke er aktuelt, idet denne sagtens kan udføres før ved denne prototype. Kalibreringen foretages med en væskesøjle, hvor der er tre punkter, som fører til at en spænding kan aflæses. Ud fra det tryk der leveres af væskesøjlen samt den målte spænding, kan en hældningskoefficient bestemmes, hvilken systemet skal gemme til at regne på alle data.

Ud fra alle disse faktorer kan grundstrukturen, til hvordan startskærmen og hovedskærmen skal se ud, dannes:



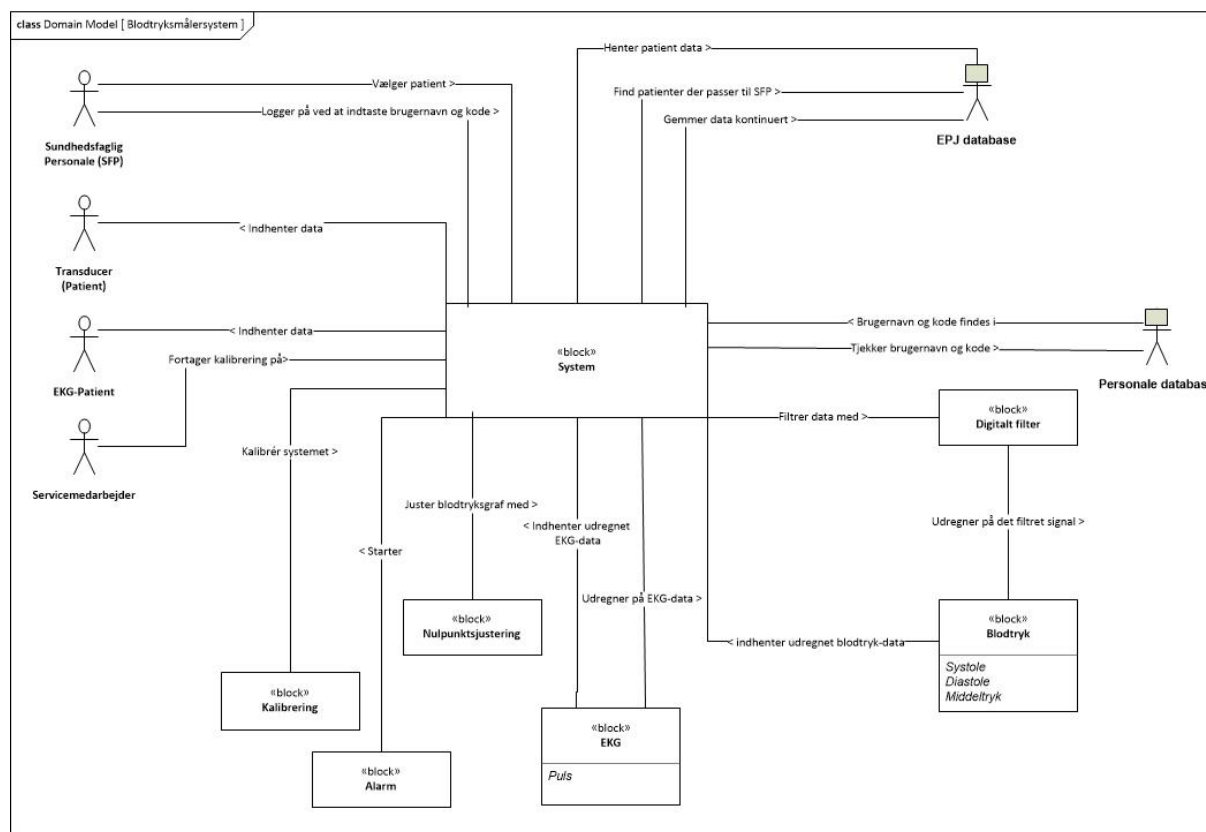
Figur 3.14: Startskærmen, hvilken fungerer som EPJ-systemet



Figur 3.15: Hovedskærmen, hvilken fungerer som blodtryksmålerens grænseflade

Domænemodel

Først skal der klarlægges hvilke klasser som systemet skal bestå af, hvilket er det første skridt i processen. For at kunne identificere disse klasse udarbejdes en domænemodel, hvilken har sit udgangspunkt i Use cases. Det er i de konceptuelle klasser fra Use cases som indeholder den information som systemet skal holde styr på. Derfor findes de konceptuelle klasser i Use cases og disse indføres i domænemodellen som klasser. Domænemodellen opstilles derfor for at finde frem til hvad problemet er i softwaren i forhold til, hvad der skal holdes styr på.



Figur 3.16: Domænemodel for blodtryksmålersystemet.

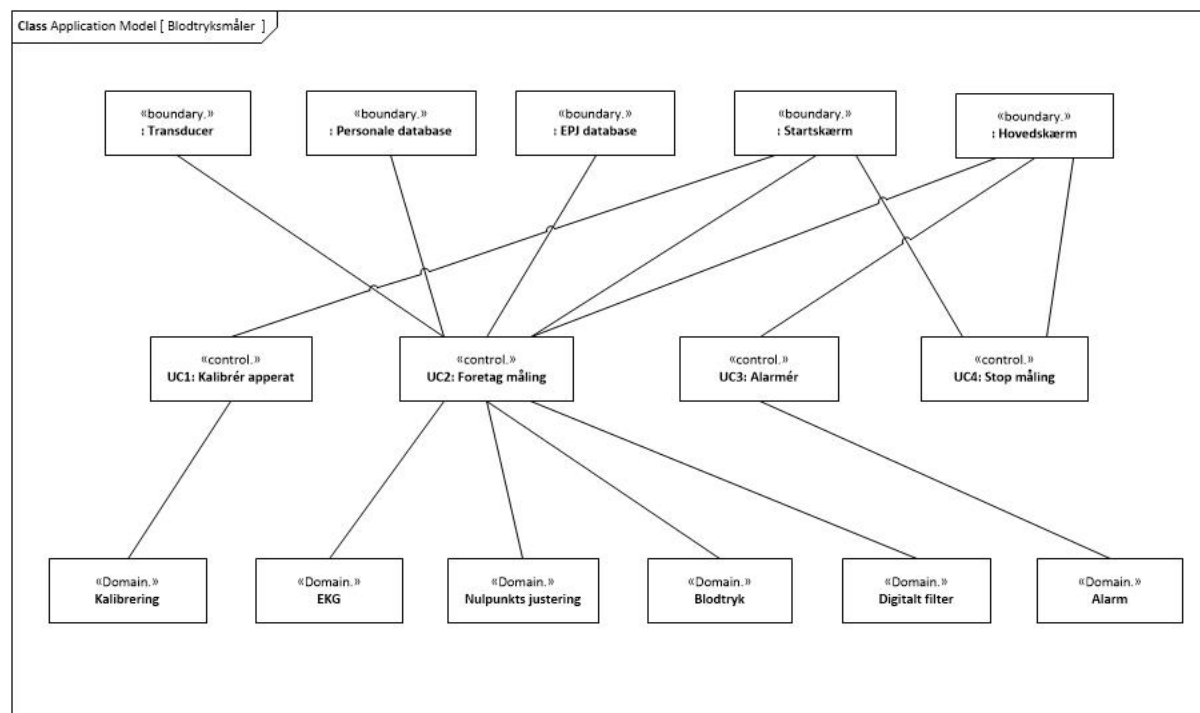
Denne domænemodel viser det sundhedsfaglige personales interaktion med systemet, samt hvilke handlinger der startes af denne interaktion. Det sundhedsfaglige personale udfører en handling, der medfører, at en række processer igangsættes i systemet. Disse processer sørger for at hente data fra transduceren og EKG patient, samt sørger for at starte beregningen af puls, systolisk, diastolisk og middeltryks værdierne. Efter beregningerne viser systemet disse værdier på brugergrænsefladen, samt sørger for at disse data bliver gemt i en database.

Klasseidentifikation

Applikationsmodel

Ud fra domænemodellen kan en applikationsmodel opstilles. Denne model tager afsæt i domænemodellens klasser. Dette betyder derfor at denne model således også tager udgangspunkt i alle Use cases.

Modellen bruges til at bestemme de interagerende klasser i blodtryksmålersystemet.



Figur 3.17: Applikationsmodel for blodtryksmålersystemet.

Ud fra denne model ses klasserne og databaserne, der skal implementeres i softwaren, samt interaktionen mellem disse. Altså hvordan klasserne taler sammen på kryds og tværs. Idet det vides at trelagsmodellen skal benyttes, kan applikationsmodellen bruges til at identificere hvor disse klasser ligger i det tilhørende lag. Databaserne vil være en boundary klasse (grænse klasse) og vil blive tilgået fra datalaget, domain klasserne (domæne klasse) er de klasser der skal ligge i logiklaget og det er disse klasser der findes fra domænemodellen og i præsentationslaget vil startskærmen og hovedskærmen, hvilke også er grænse klasser, ligge hvorfra de vil blive præsenteret for det sundhedsfaglige personale. Control klasserne (kontrol klasserne) er systemets Use cases. Grænse klasserne er de klasser der repræsenterer aktørerne fra Use casene og disse aktøreres grænseflader. Domæne klasserne er de klasser hvori data bliver behandlet og bearbejdet. Kontrol klasserne er de klasser der udfører Use casene ved at interagere med grænse og domæne klasserne.

Trelagsmodellen Trelags modellen er en software model, der gør det muligt at inddele software kodeN op i tre lag. De tre lag er præsentationslaget, logiklaget og datalaget.

Fordelen ved at kunne dele softwaren op på denne måde, er at man kan rette i de forskellige lag, uden at det får indflydelse i de andre lag. Man kan herved implementere nye klasser og

deres funktioner, og blot henvise til dem i de andre lag. Hvert lag har deres egen funktionalitet og ansvar.

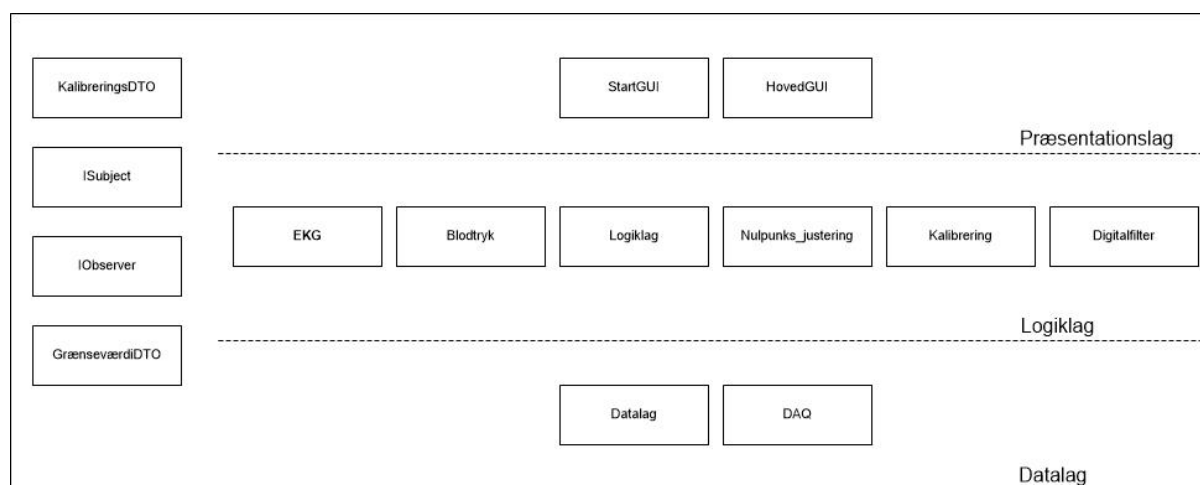
Præsentationslaget er det øverste lag i trelags modellen, og kan kun snakke sammen med logiklaget. I præsentationslaget må der ikke laves nogen form for beregninger, disse skal komme fra logiklaget. Med andre ord fungerer præsentationslaget, kun til at præsentere data på en brugergrænseflade, hvorfra data kan præsenteres for brugeren. Præsentationslaget er derfor også det eneste lag, som brugeren kommer til at integrerer med.

Logiklaget er det midterste lag, og er et bindeled mellem præsentationslaget og datalaget. I logiklaget forgår alt behandling af data fra præsentationslaget og datalaget. Det betyder at alle algoritmer skal ligge i logiklaget.

Datalaget er det nederste lag, og her indhentes eller gemmes data. Der kan indlæses data fra en fil, eller et måleapparat eller en database. Datalaget kan gemme data fra logiklaget i en fil eller database.

I trelagsmodellen kan der skabes et flydende bindeled, der kan snakke sammen med alle lagene, dette kaldes for en DTO. En DTO står for Data transfer object, og transporter objekter imellem alle lagene, og kan derved tilgås af alle lagene.

I dette projekt er trelagsmodellen blevet opfyldt ved at der er oprettet en solution der hedder PulsMaalerSystem, under denne solution ligger der fire projekter; PulsMaalerSystem, PulsMaalerSystem.Logiklag, PulsMaalerSystem.Datalag og PulsMaalerSystem.DTOlag. Opdeling af softwarekoden i trelagsmodel kan ses på figuren nedenfor.



Figur 3.18: Trelagsmodellen. Her ses hvor de forskellige klasser ligger i der forskellige lag.

Det første projekt PulsMaalerSystem fungerer som præsentationslaget og består af to Windows forms, StartGUI og HovedGUI. Begge forms sender informationer ned til logiklaget og henter data, som blodtryksværdier, og viser det på en graf.

Det andet projekt PulsMaalerSystem.Logiklag er logiklaget i projektet. I dette logiklag, er der implementeret seks klasser: Blodtryk, Kalibrering, Digitalfilter, Nulpunkts_justering og EKG, Logiklag.

Klassen Blodtryk tager sig af at udregne systolen, diastolen og middeltrykket med algoritmer ud fra blodtrykket.

Kalibrerings klassen sørger for at beregne kalibreringsværdien ud fra de indtastede værdier for

tryk og spænding og sender denne værdi over i Logiklag klassen, hvilken kalibrér blodtrykssignalet ved at gange kalibreringsværdien på signalet.

Digitaltfilter klassen implementer et glidende middelværdisfilter, der sørger for at udglatte blodtrykssignalet.

Nulpunktsjustering klassen sørger for at sætte signalets baseline ved 0 V. EKG klassen skulle have sørget for at finde R-takker og udregne pulsen. Grundet tidspres har det ikke været muligt at implementer EKG i projektet. Der er dog gjort klar til det med denne klasse.

Logiklags klassen binder alle de andre klasser sammen, ved at hente data nede fra datalaget til klasserne og sende metoderne med videre til præsentationslaget.

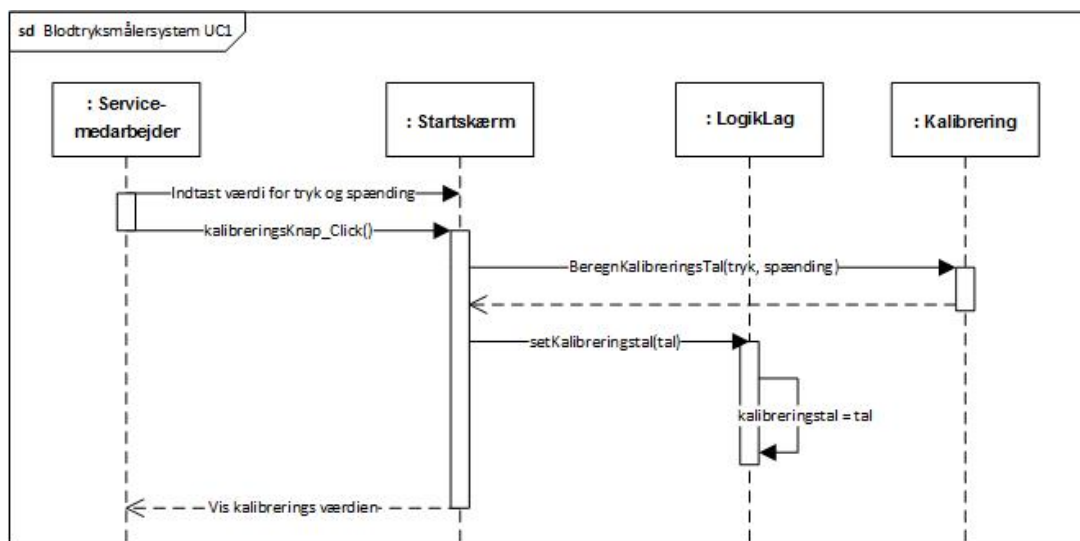
Det tredje projekt PulsMaalerSystem.Datalag består af to klasser: DAQ og datalag. DAQ klassen sørger for at hente data op fra DAQ'en. Datalag klassen sørger for at hente informationer om patienten og personalet i databaserne; Personale database og EPJ database. Datalag klassen sørger også for at gemme blodtrykket for patienten løbende i EPJ databasen.

Det sidste projekt PulsMaalerSystem.DTOlag består af tre klasser: GrænseværdiDTO, IObserver, ISubject og KalibreringsDTO. GrænseværdiDTO klassen indeholder get og set metoder for grænseværdien, og sender grænseværdierne videre med en metode. IObserver kigger ned på subjectet, hvor klassen ISubject . KalibreringsDTO klassen har get og set metoder for spænding og tryk, og en metode til at sende disse videre.

Metodeidentifikation

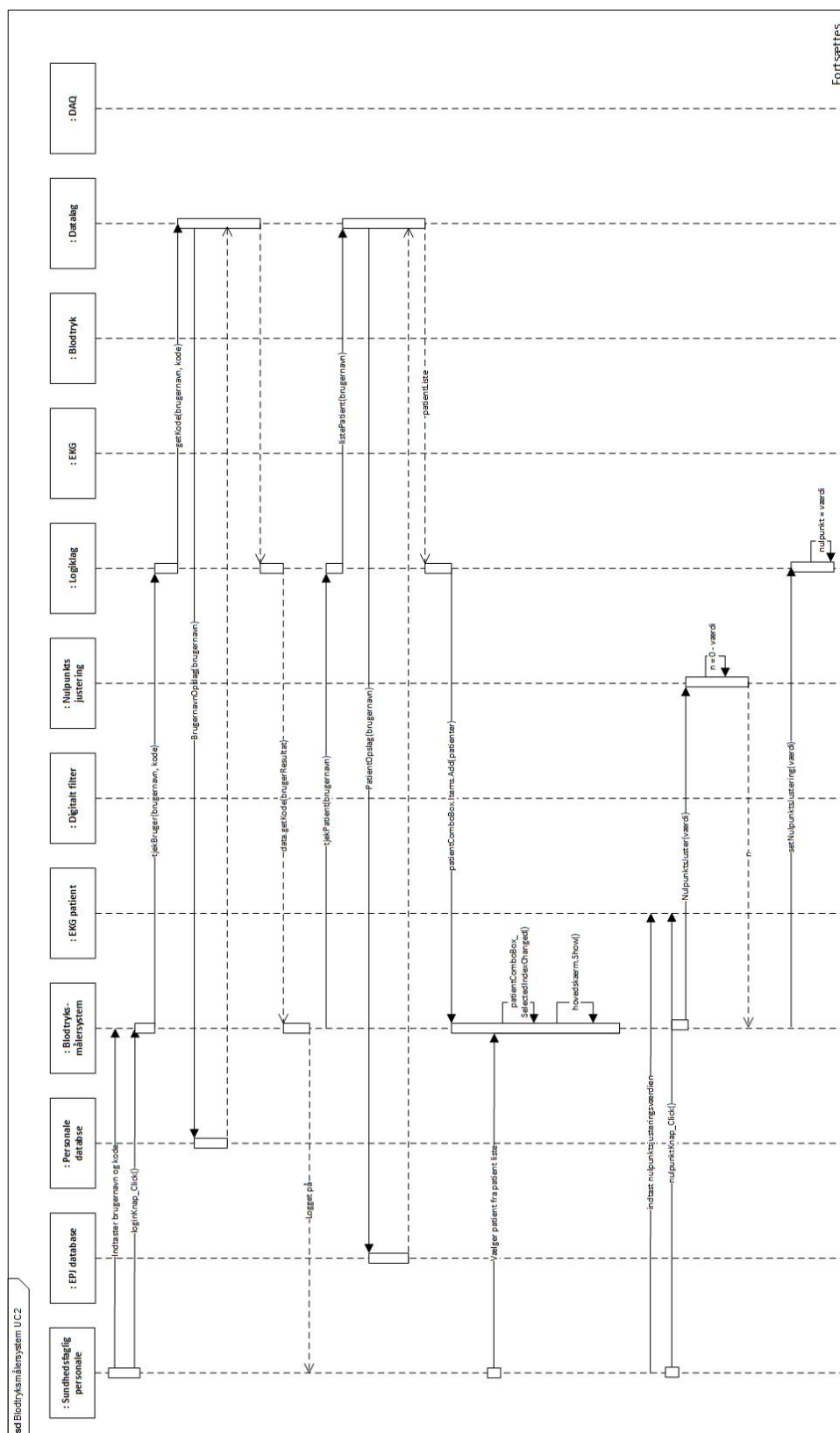
Sekvensdiagrammer

Nedenfor er vist sekvensdiagrammer for systemet. Der er lavet sekvens diagrammer for alle Use case. Vores Use cases er henholdsvis Use case 1: Kalibrér apparat, Use case 2: Foretag måling, Use case 3: Alarmér og Use case 4: Stop måling. Sekvens diagrammet er et interaktionsdiagram, som viser hvorledes processerne forløber i forhold til hinanden. Ud fra sekvensdiagrammerne kan det altså ses hvornår og hvordan de forskellige dele i systemet forløber og interagerer.

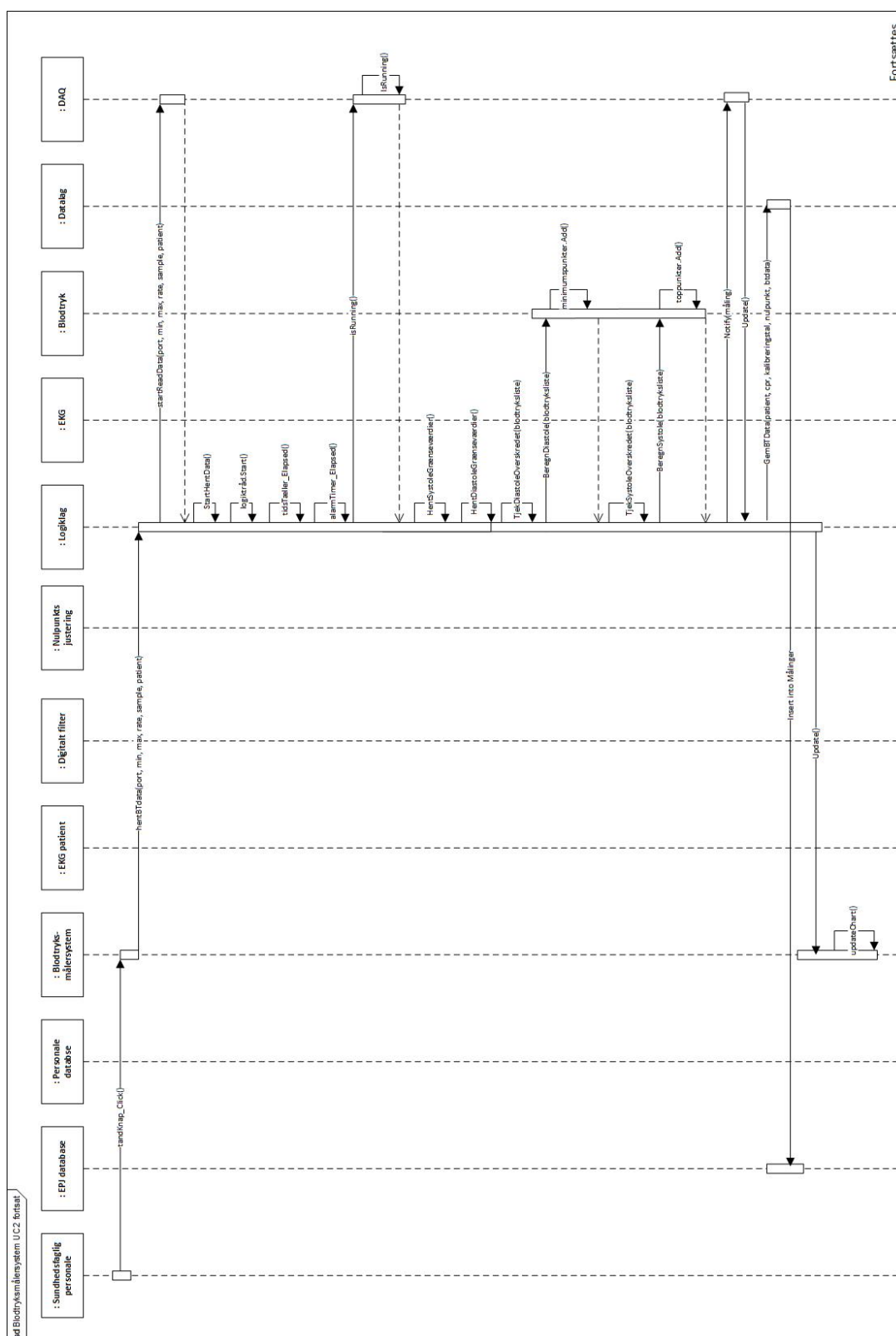


Figur 3.19: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 1

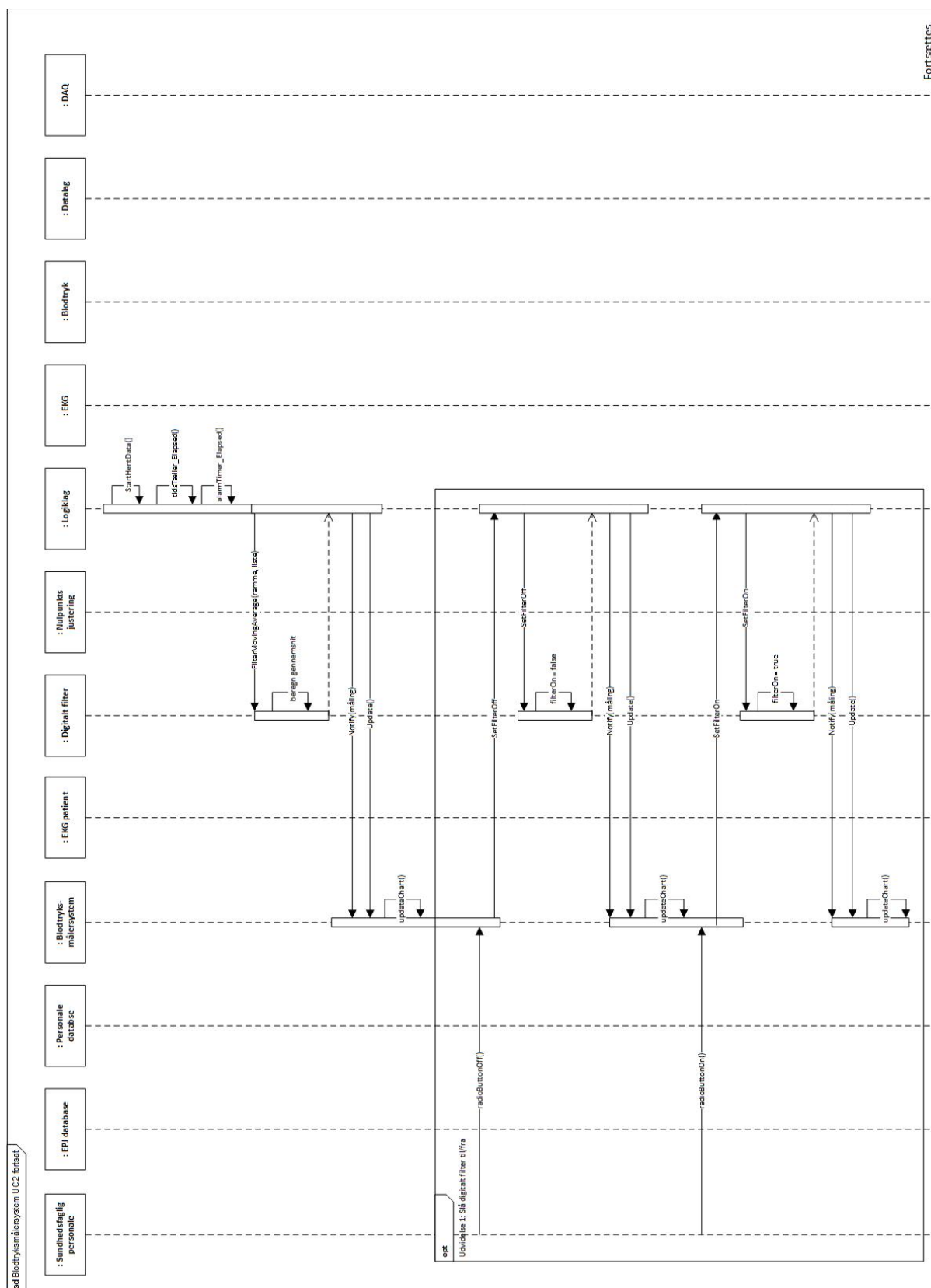
I sekvens diagrammet for Use case 1 interagerer servicemedarbejder med blodtryksmålersystemet. Her er det servicemedarbejderen som starter kalibreringen og blodtryksmålersystemet som foretager kalibreringen igennem lagene og klassen Kalibrering. Her ses det at servicemedarbejderen indtaster værdierne der aflæses for tryk og spænding, hvorefter der trykkes på knappen, hvorefter systemet foretager beregningen for kalibreringen.



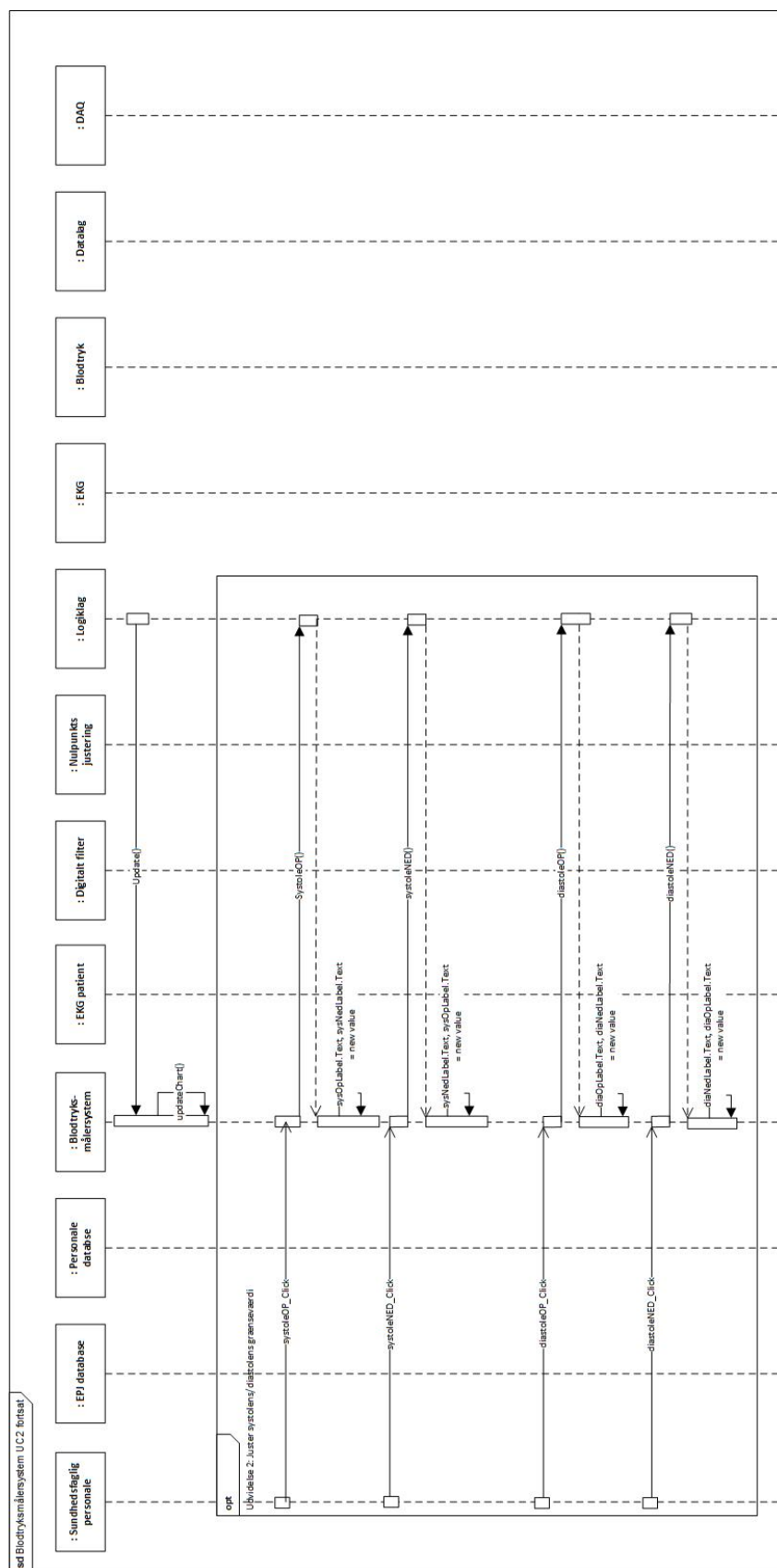
Figur 3.20: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 2, del 1 af 4



Figur 3.21: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 2, del 2 af 4

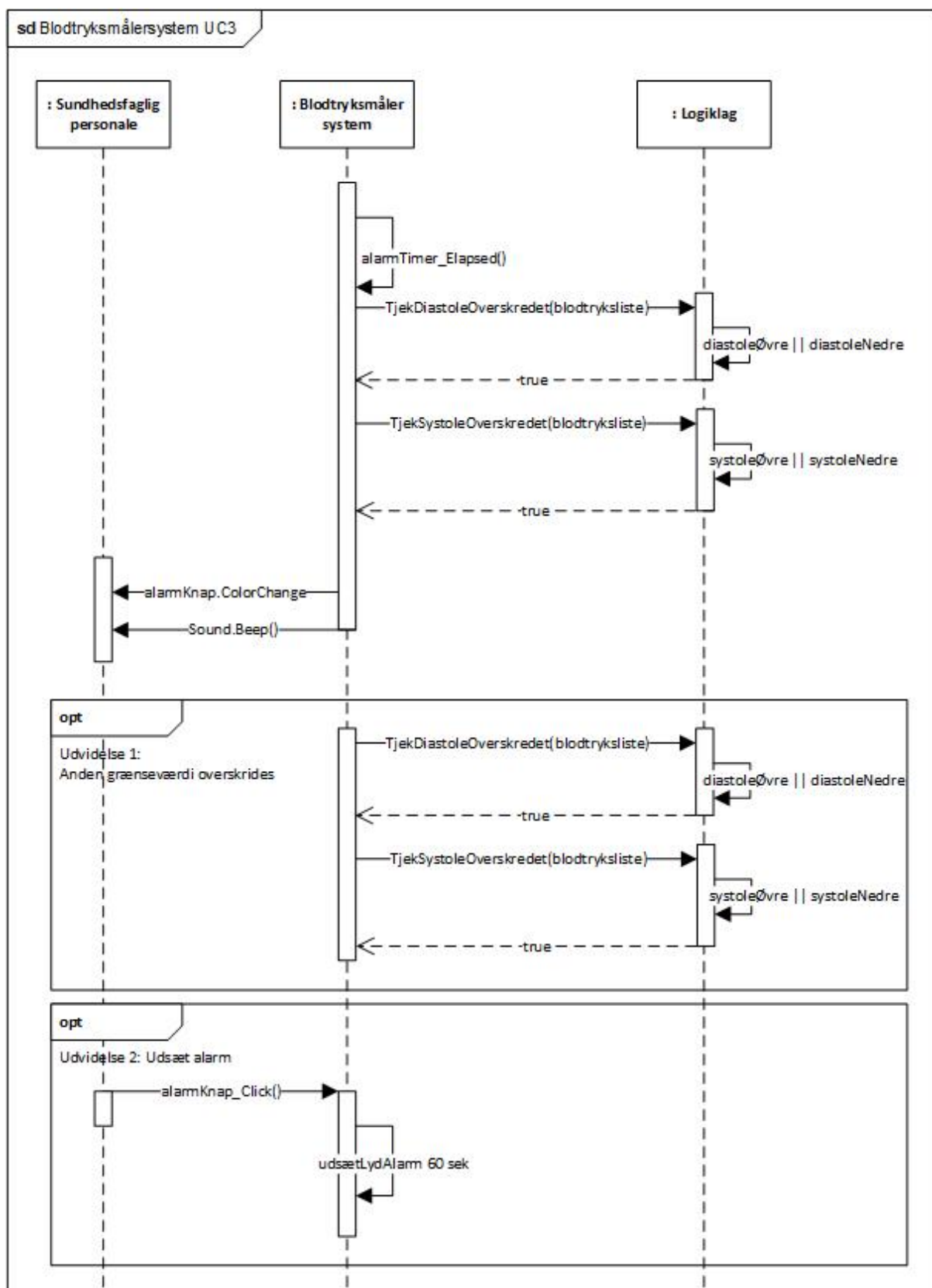


Figur 3.22: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 2, del 3 af 4



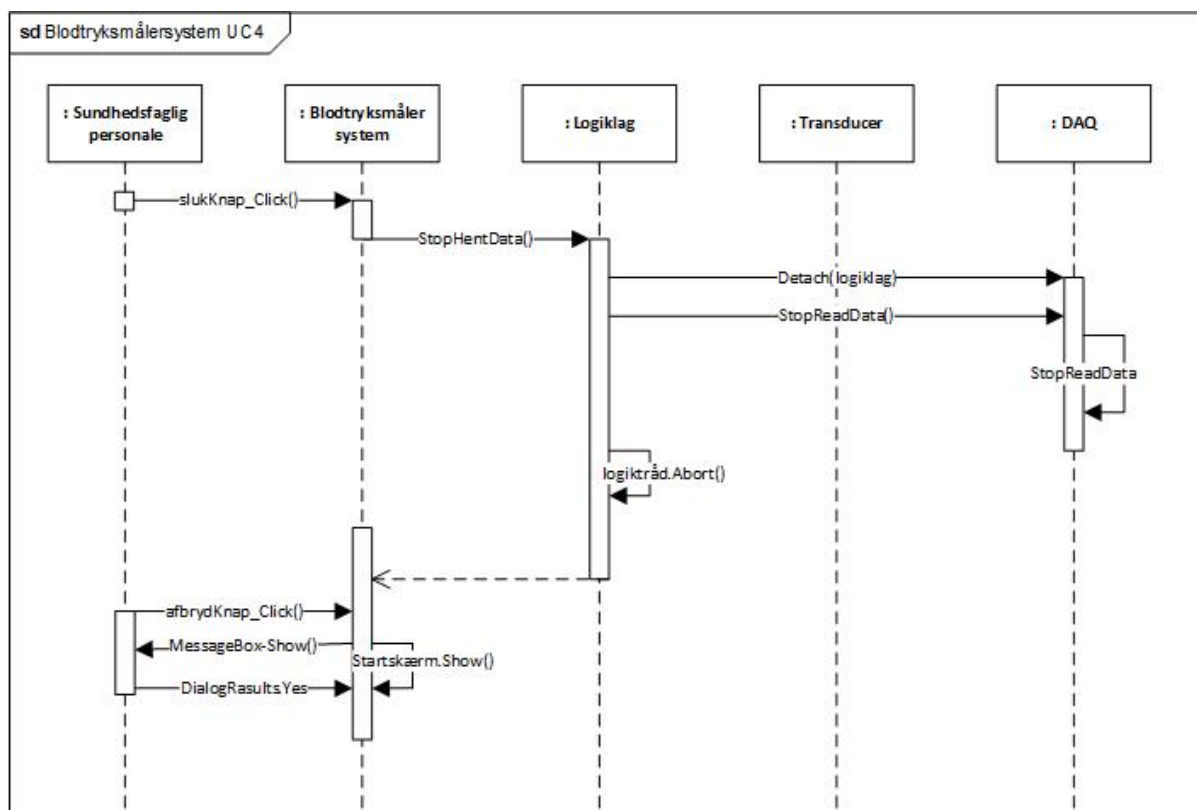
Figur 3.23: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 2, del 4 af 4

I sekvens diagrammet for Use case 2 ønsker det sundhedsfaglig personale at foretage måling, dette gøres ved at det sundhedsfaglig personale interagerer med blodtryksmålersystemet. For at målingen forløber, foregår den videre interaktion via blodtryksmålersystemet og de klasser, som indgår i Use casen. Transducer, EKG patient sender sine data til datalaget, EPJ database og Personale database får sine data fra datalaget. I dette sekvens diagram ses det sundhedsfaglige personales interaktion med blodtryksmålersystemet. Denne interaktion igangsætter processerne, som medfører at blodtryksmålingen forløber. Det ses også, at præsentrationslaget, som er vores blodtryksmålersystem, kommunikerer med logiklaget og logiklaget kommunikerer med datalag, altså er trelagsmodellen opfyldt, idet datalaget ikke direkte kan kommunikere med præsentrationslaget, men skal igennem logiklaget.



Figur 3.24: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 3

I dette sekvens diagram for Use case 3 tjekker logiklaget hvorvidt grænseværdierne for systole og diastole er overskredet. Hvis en grænseværdi overskrides starter alarmeringen. I denne Use case har det sundhedsfaglig personale mulighed for at udsætte alarmen, dette sker ved en interaktion mellem sundhedsfaglig personale og blodtryksmålersystemet.



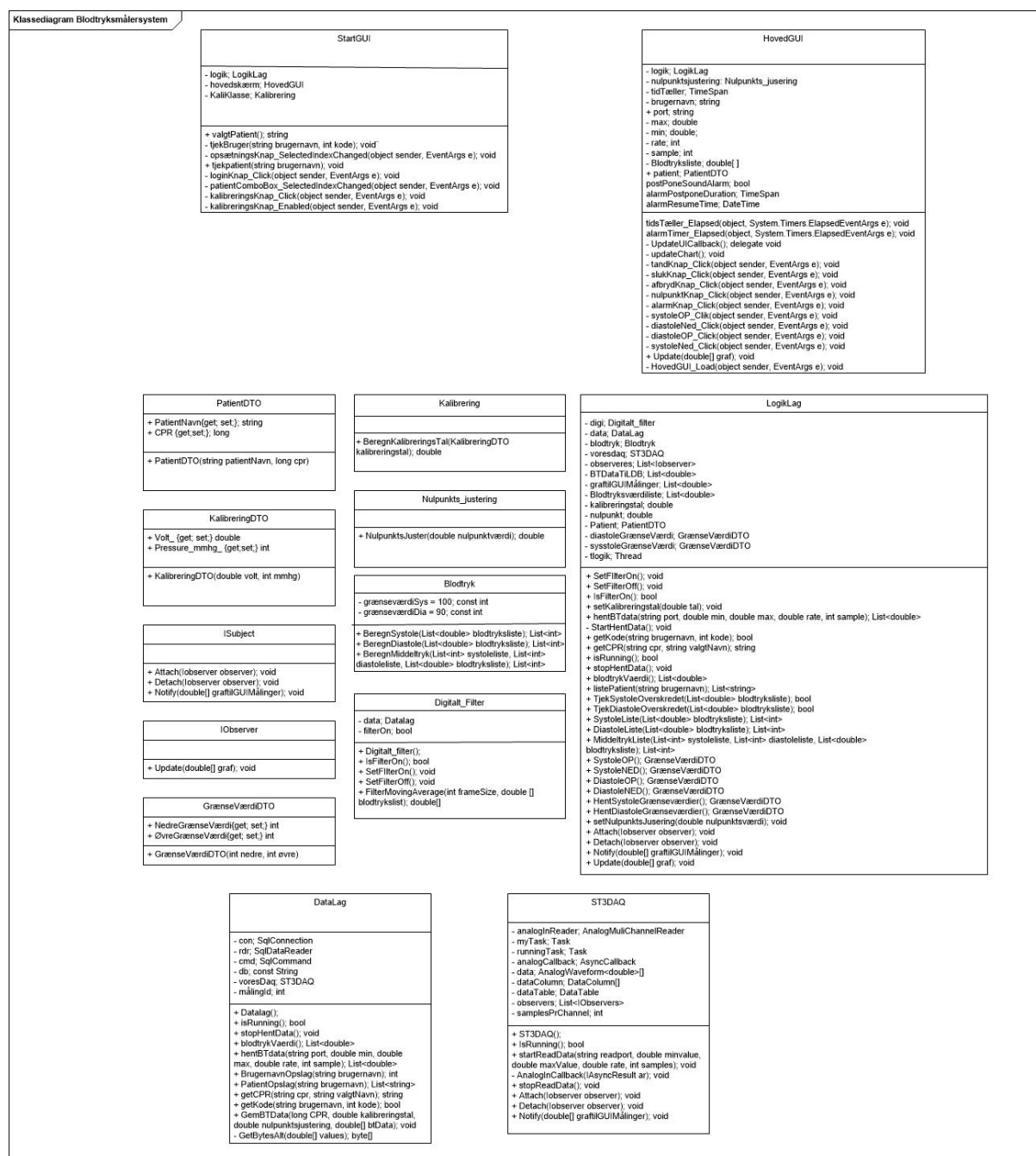
Figur 3.25: Sekvensdiagram for blodtryksmålersystemet. Denne viser adfærden for Use case 4

I sekvens diagrammet for Use case 4 ønsker sundhedsfaglig personale, at stoppe målingen. Dette gøres ved, at sundhedsfaglig personale interagerer med blodtryksmålersystemet i præsenteringslaget. Denne interaktion medfører en videre kommunikation mellem logiklaget og data-laget, som medfører at DAQ'en ikke længere henter data fra transduceren. Når blodtryksmålingen er afsluttet kan det sundhedsfaglige personale logge ud.

Klassediagram

Ud fra domænemodellen er der fundet ud af hvilke klasser blodtryksmålersystemet skal bestå af. Ud fra applikationsmodellen er samhørigheden af klasserne bestemt. Der skal være lav kobling og høj samhørighed, dette betyder at klasserne ikke må være koblet sammen på tværs i de forskellige lag. Ud fra sekvensdiagrammerne analyseres det hvilke metoder, der tilhører hver klasse.

Ud fra informationerne fra domænemodellen, applikationsmodellen og sekvensdiagrammet kan der laves et klassediagram, der indeholder attributterne og metoderne for hver klasse fra domænemodellen.

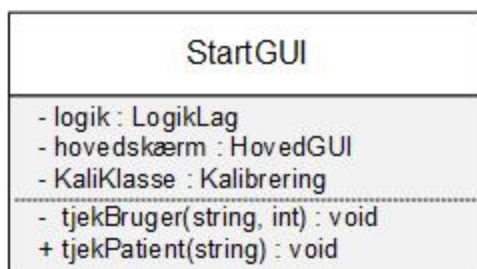


Figur 3.26: Heraf ses klasserne med de tilhørende attributter og metoder.

Metodebeskrivelse

Metoderne der identificeres i klassediagrammet vil her beskrives, hvor det beskrives, hvilken parameter metoden tager imod, hvilken returnværdi metoden sender videre og en kort beskrivelse af metodens funktion. Event metoderne der ikke er i klassediagrammet, vil dog her ligeledes blive beskrevet.

StartGUI



Figur 3.27: Klassediagram for StartGUI.

public StartGUI()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Denne metode sørger for forbindelse til LogikLag klassen og Kalibrering klas- sen.

private void tjekBruger(string, int)	
Parameter:	string, int
Returværdi:	Ingen
Beskrivelse:	Metoden går ind og kalder metoden getKode() fra LogikLag klassen for at tjekke om det indtastede brugernavn og kode hører sammen.

private void opsætningKnap_ SelectedIndexChanged(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden gør det muligt at skrive tekst i idTextBox og passwordTextBox og gør loginKnap mulig at trykke på.

public void tjekPatient(string)	
Parameter:	string
Returværdi:	Ingen
Beskrivelse:	Denne metode går ind og kigger på hver patient i patientlisten fra LogikLag, og ser hvilke patienter som er koblet sammen med det indtastede personale brugernavn og tilføjer patienterne til listen af patienter i PatientComboBox.

private void loginKnap_ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden kalder metoden tjekBruger(), hvor det indtastede brugernavn og pas- sword sendes med. Ryder PatientComboBoxens liste. Kalder metoden tjekPa- tient(), som sender brugernet med.

private void patientComboBox__ SelectedIndexChanged(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode kalder PatientDTO klassen og bruger denne til at vælge patient fra patientComboBox. Sender den valgte port videre til HovedGUI. Viser HovedGUI og skjuler StartGUI.

private void kalibreringKnap__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode kalder KalibreringsDTO og henter syntaksen for værdierne indtastet i Volttextbox og mmHgTextBox. Kalder metoden BeregnKalibreringsTal() i Kalibrering klassen, hvilken bruger de indtastede værdier og kalder metoden setKalibreringstal() fra LogikLag klassen, hvilken bruger den beregnede værdi. Værdien udskrives herefter på startskærmen.

private void kalibreringsKnap__ Enabled(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode gør kalibrerings knappen enabled, altså gør det muligt at trykke på knappen.

HovedGUI

HovedGUI	
-	logik : LogikLag
-	nulpunktsjustering : Nulpunkts_Justering
-	tidTæller : TimeSpan
-	brugernavn : string
+	port : string
-	max : double
-	min : double
-	rate : int
-	sample : int
-	blodtrykliste : double[]
+	patient : PatientDTO
	alarmTimer : System.Timers.Timer
	tidsTimer : System.Timers.Timer
	postPoneSoundAlarm : bool
	alarmPostponeDuration : TimeSpan
	alarmResumeTime : DateTime
	tidsTæller_Elapsed(object, System.Timers.ElapsedEventArgs) : void
	alarmTimer_Elapsed(object, System.Timers.ElapsedEventArgs) : void
-	UpdateUICallback() : delegate void
-	updateChart() : void
+	Update(double[]) : void

Figur 3.28: Klassediagram for HovedGUI

public HovedGUI(string, LogikLag)	
Parameter:	string, LogikLag
Returværdi:	Ingen
Beskrivelse:	Denne metode bruges til sætter tiden hvormed alarmen skal udsættes. Sætter det digitale filter på signalet pr. default og hvis radioButtonOFF vælges bliver det digitale filter slået fra, hvorefter det bliver muligt at slå digitalt filter til ved at vælge radioButtonON.

void tidsTæller_ Elapsed(object, System.Timers.ElapsedEventArgs)	
Parameter:	object, System.Timers.ElapsedEventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden henter tidTæller tråden og tæller den én værdi op hvert sekund og viser værdien på HovedGUI.

void alarmTimer_ Elapsed(object, System.Timers.ElapsedEventArgs)	
Parameter:	object, System.Timers.ElapsedEventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden kalder TjektDiastoleOverskredet() metoden fra LogikLag og kobler blodtrykslisten på denne, hvilken går ind og ser om der er en grænseværdi der er overskredet. Hvis en grænseværdi er overskredet skifter alarmKnap baggrundsbillede mellem to billeder; et rødt og et hvidt og starter en Beep.Play(). TjektSystoleOverskredet() metoden fra LogikLag kaldes og kobles til blodtrykslisten, hvilken går ind og ser om der er en grænseværdi der er overskredet. Hvis en grænseværdi er overskredet skifter alarmKnap baggrundsbillede mellem to billeder; et rødt og et hvidt, og starter en Beep.Play().

private delegate void UpdateUICallback()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	En delegate der bliver koblet til updateChart().

private void updateChart()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden er en tråd, hvilken går ind og ser om chart Blodtryk ligger i en anden tråd og starter tråden. Metoden går ind og sender værdier fra blodtrykslisten ind i Blodtryk chart. Henter systoliske, diastolisk og middeltryks værdier, og sender disse værdier til labels på startskærmen.

private void tandKnap_ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden kalder hentBTdata() og Attach() metoderne fra LogikLag. Kalder HentSystoleGrænseværdier() og HentDiastoleGrænseværdier() metoderne for at hente øvre og nedre grænseværdier.

private void slukKnap__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne kalder metoderne StopHentData() og Detach() fra LogikLag klassen.

private void afbrydKnap__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden åbner en MessageBox som spørger "Er du sikker?" hvis "ja" vises StartGUI og HovedGUI skjules.

private void nulpunktKnap__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden kalder NulpunktsJuster() fra Nulpunkts__justering klassen, og sender den indtastede værdi videre for at beregne værdien. setNulpunktsJustering() metoden fra LogikLag kaldes og den beregnede værdi sendes til LogikLag.

private void alarmKnap__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode udsætter alarmens lyd ved at kalde postPoneSoundAlarm og udsættelsens varighed sættes.

private void systoleOP__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode gør at de systoliske grænseværdier forøges. Metoden SystoleOP() fra LogikLag kaldes, hvorfra ØvreGrænseværdi og NedreGrænseVærdi ændres.

private void systoleNED__ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode gør at de systoliske grænseværdier nedsænkes. Metoden SystoleNED() fra LogikLag kaldes, hvorfra ØvreGrænseværdi og NedreGrænseVærdi ændres.

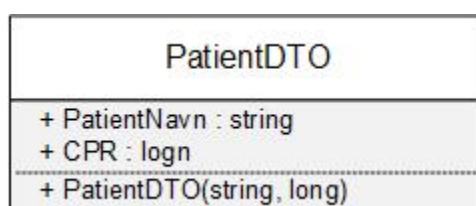
private void diastoleOP_ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode gør at de diastoliske grænseværdier forøges. Metoden DiastoleOP() fra LogikLag kaldes, hvorfra ØvreGrænseværdi og NedreGrænseVærdi ændres.

private void diastoleNED_ Click(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Denne metode gør at de diastoliske grænseværdier sænkes. Metoden DiastoleNED() fra LogikLag kaldes, hvorfra ØvreGrænseværdi og NedreGrænseVærdi ændres.

public void Update(double[])	
Parameter:	double[]
Returværdi:	Ingen
Beskrivelse:	Metoden sætter blodtrykslisten lig med et double array og kalder metoden updateChart().

private void HovedGUI_ Load(object, EventArgs)	
Parameter:	object, EventArgs
Returværdi:	Ingen
Beskrivelse:	Metoden sætter navn og cpr på HovedGUI ved at kalde PatientDTO, hvilken indeholder denne information.

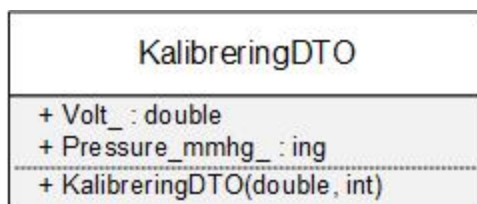
PatientDTO



Figur 3.29: Klassediagram for PatientDTO.

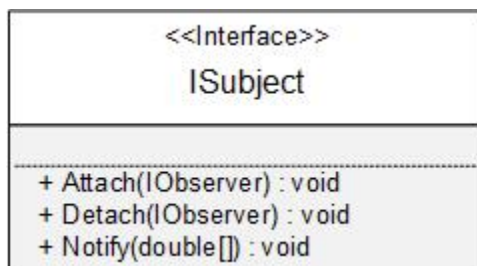
public PatientDTO(string, long)	
Parameter:	string, long
Returværdi:	Ingen
Beskrivelse:	Denne sætter syntaksen for patienten og dermed hvilke parametre denne indeholder.

KalibreringDTO



Figur 3.30: Klassediagram for KalibreringDTO.

public KalibreringDTO(double, int)	
Parameter:	double, int
Returværdi:	Ingen
Beskrivelse:	Denne sætter syntaksen for kalibrering og dermed hvilke parametre der skal benyttes til kalibrering.

ISubject

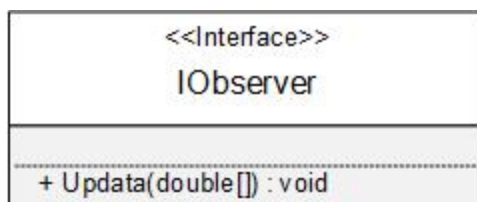
Figur 3.31: Klassediagram for ISubject.

void Attach(IObserver)	
Parameter:	IObserver
Returværdi:	Ingen
Beskrivelse:	Denne metode modtager en observer.

void Detach(IObserver)	
Parameter:	IObserver
Returværdi:	Ingen
Beskrivelse:	Denne metode modtager en observer.

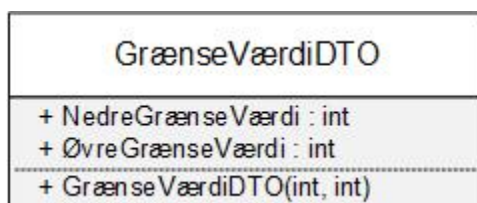
void Notify(double[])	
Parameter:	double[]
Returværdi:	Ingen
Beskrivelse:	Denne metode modtager et array med grafmålinger.

IObserver



Figur 3.32: Klassediagram for IObserver.

void Update(double[])	
Parameter:	double[]
Returværdi:	Ingen
Beskrivelse:	Denne metode modtager et array, hvori grafen ligger.

GrænseVærdiDTO

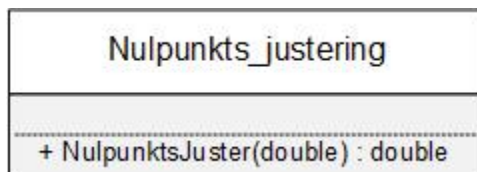
Figur 3.33: Klassediagram for GrænseVærdiDTO.

public GrænseVærdiDTO(int, int)	
Parameter:	int, int
Returværdi:	double
Beskrivelse:	Metoden sætter syntaksen sådan at der skal være en øvre og nedre grænseværdi.

Kalibrering

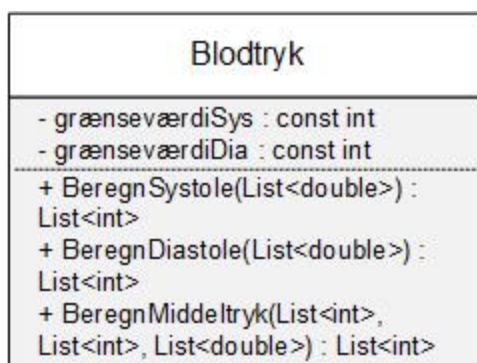
Figur 3.34: Klassediagram for Kalibrering.

public double BeregnKalibreringsTal(KalibreringDTO)	
Parameter:	KalibreringDTO
Returværdi:	double
Beskrivelse:	Denne metode beregner kalibrerings værdien ved at modtage syntaksen fra KalibreringsDTO: tryk divideret med spænding. Og returnerer denne som en double værdi.

Nulpunkts__justering

Figur 3.35: Klassediagram for Nulpunkts__justering.

public double NulpunktsJuster(double)	
Parameter:	double
Returværdi:	double
Beskrivelse:	Metoden sætter nulpunktsværdien, ved at trække den indtastede værdi fra nul og returnere denne værdi som en double.

Blodtryk

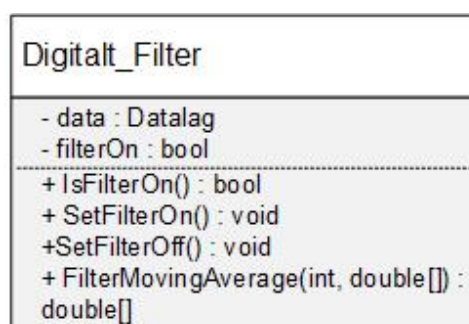
Figur 3.36: Klassediagram for Blodtryk.

public List<int> BeregnSystole(List<double>)	
Parameter:	List<double>
Returværdi:	List<int>
Beskrivelse:	Metoden går ind og kigger på samtlige værdier i blodtrykssliste. Metoden ser om værdien er over en sat baseline, hvis den er det, går metoden ind og ser om den næste værdi er højere, når værdien bliver lavere igen tilføjer metoden den højeste værdi til listen toppunkter og returnerer denne liste.

public List<int> BeregnDiastole(List<double>)	
Parameter:	List<double>
Returværdi:	List<int>
Beskrivelse:	Metoden går ind og kigger på samtlige værdier i blodtrykssliste. Metoden ser om værdien er under en sat baseline, hvis den er det, går metoden ind og ser om den næste værdi er lavere, når værdien bliver højere igen tilføjer metoden den laveste værdi til listen minimumspunkter og returnerer denne liste.

public List<int> BeregnMiddeltryk(List<int>, List<int>, List<double>)	
Parameter:	List<int>, List<int>, List<double>
Returværdi:	List<int>
Beskrivelse:	Metoden går ind og kigger på samtlige værdier i diastolelisten og systolelisten. Metoden tager her en værdi for diastole og en værdi for systole og beregner middeltryk ved formlen: $\frac{2}{3}diastole + \frac{1}{3}systole$. Tilføjer værdien til en middeltrykliste og returnerer denne.

Digitalt__ filter



Figur 3.37: Klassediagram for Digitalt__ Filter.

public Digitalt__ filter()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Denne sætter boolean filterOn lig med true.

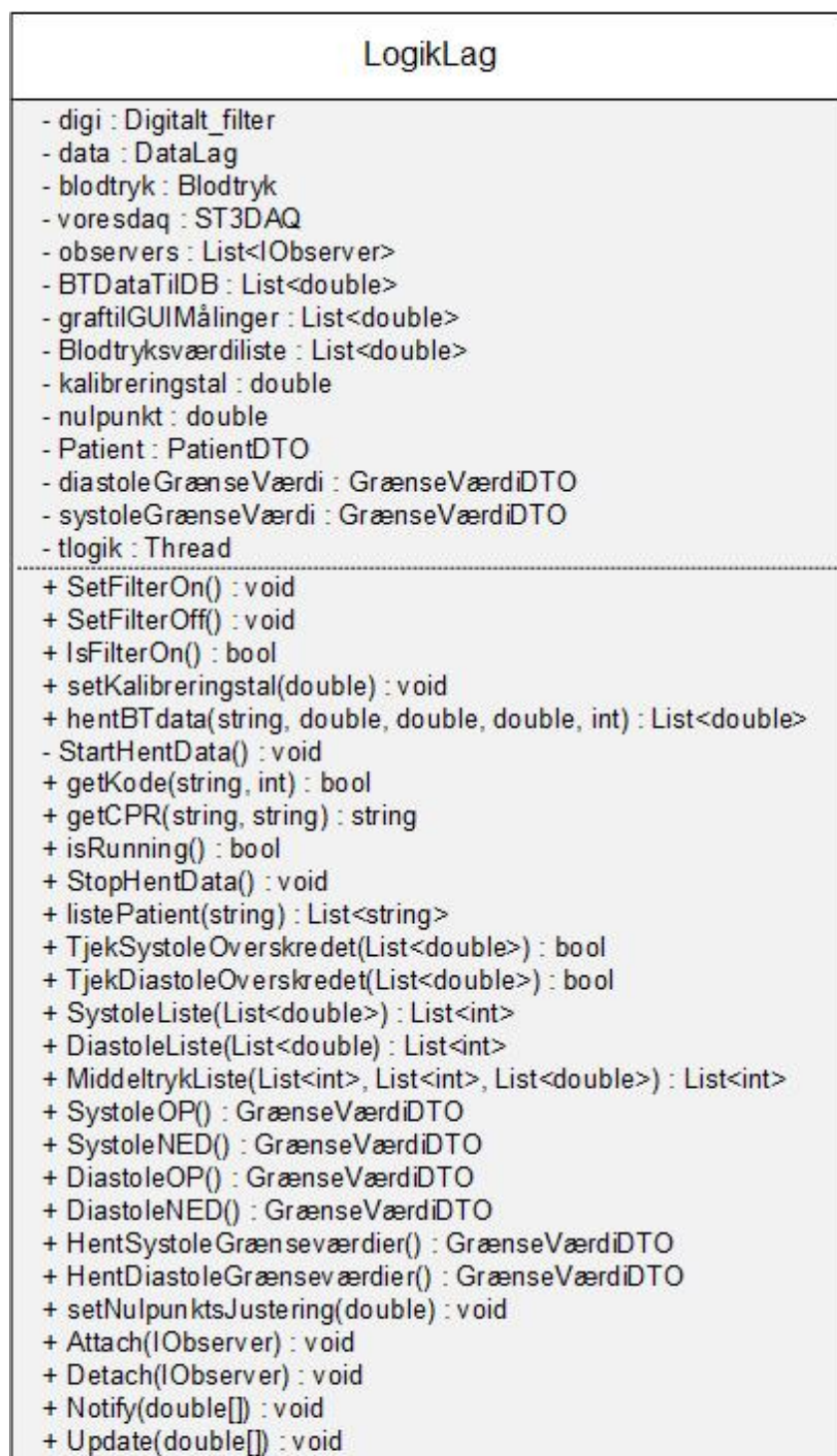
public bool IsFilterOn()	
Parameter:	Ingen
Returværdi:	bool
Beskrivelse:	Metoden returnerer filterOn

public bool SetFilterOn()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden sætter filterOn lig true.

public bool SetFilterOff()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden sætter filterOn lig false.

public double[] FilterMovingAverage(int, double[])	
Parameter:	int, double[]
Returværdi:	double[]
Beskrivelse:	Metoden gør sådan at blodtryklisten kører igennem en ramme, hvor gennemsnittet beregnes af rammens indhold og rammen rykker én plads efter beregningen, sådan at beregningen sker glidende på blodtryklisten. Metoden returnerer det beregnede gennemsnit i et double array.

LogikLag



Figur 3.38: Klassediagram for LogikLag.

public void SetFilterOn()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Denne metode kalder Digitalt_filter klassens metode SetFilterOn()

public void SetFilterOff()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Denne metode kalder Digitalt__ filter klassens metode SetFilterOff()

public bool IsFilterOn()	
Parameter:	Ingen
Returværdi:	bool
Beskrivelse:	Denne metode returnerer Digitalt__ filter metoden IsFilterOn()

public void setKalibreringstal(double)	
Parameter:	double
Returværdi:	Ingen
Beskrivelse:	Metoden sætter kalibreringstallet til et double tal.

public void hentBTdata(string, double, double, double, int, PatientDTO)	
Parameter:	string, double, double, double, int, PatientDTO
Returværdi:	Ingen
Beskrivelse:	Metoden kalder ST3DAQ metoden startReadData(). Implementere og starter tråden af typen StartHentData i logiklaget.

Private void StartHentData()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden laver en kø-metode med 500 pladser. Kalder Attach() metoden fra ST3DAQ. Så længe metoden isRunning gælder, og der kommer værdier ind, sendes der værdier ind i køen indtil der er 500 objekter i denne, hvor efter der sendes et objekt ud inden der sendes et nyt ind. Inden objekterne tilføjes til køen, bliver nulpunktværdien lagt til, og kalibreringsværdien bliver ganget på hvert tal i Blodtryksværdilisten. Herefter kaldes tælleren sådan at der kun tages hvert tiende tal i Blodtryksværdilisten. Digitalt__ filter metoden IsFilterOn() kaldes, hvilket tjekker om filteret er slået til, hvis filteret er det kaldes metoden Digitalt__ filter FilterMovingAverage() hvilket gør at værdierne i kø-metoden bliver filtreret. Notify metoden kaldes til sidst for at sende arrayet med den behandlede data op til HovedGUI

public bool getKode(string, int)	
Parameter:	string, int
Returværdi:	bool
Beskrivelse:	Metoden returnerer koden, hvor denne er lig med koden, som hentes med brugernavn parameteren, fra metoden getKode fra DataLag.

public string getCPR(string, string)	
Parameter:	string, sting
Returværdi:	string
Beskrivelse:	Metoden returnerer cpr.

public bool isRunning()	
Parameter:	Ingen
Returværdi:	bool
Beskrivelse:	Metoden returnerer ST3DAQ klassens metode IsRunning().

public void StopHentData()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden kalder ST3DAQ klassens metoder Detach() og StopReadData(). Metoden stopper tråden der startes i hentBTdata().

public List<PatientDTO> listePatient(string)	
Parameter:	string
Returværdi:	List<PatientDTO>
Beskrivelse:	Metoden returnerer metoden PatientOpslag() fra DataLag, hvilken har syntaksen fra PatientDTO.

public bool TjekSystoleOverskredet(List<double>)	
Parameter:	List<double>
Returværdi:	bool
Beskrivelse:	Metoden henter systoleværdierne ved at kalder BeregnSystole() metoden fra Blodtryk. Metoden tjekker om systole værdien er over den øvre grænseværdi eller under den nedre grænseværdi og hvis én af værdierne har en højere/lavere værdi er grænseværdien overskredet. Metoden returnerer resultatet.

public bool TjekDiastoleOverskredet(List<double>)	
Parameter:	List<double>
Returværdi:	bool
Beskrivelse:	Metoden henter diastoleværdierne ved at kalder BeregnDiastole() metoden fra Blodtryk. Metoden tjekker om diastole værdien er over den øvre grænseværdi eller under den nedre grænseværdi og hvis én af værdierne har en højere/lavere værdi er grænseværdien overskredet. Metoden returnerer resultatet.

public List<int> SystoleListe(List<double>)	
Parameter:	List<double>
Returværdi:	List<int>
Beskrivelse:	Metoden returnerer en liste med systoler ved at koble blodtrykslisten sammen med metoden BeregnSystole() fra Blodtryk.

public List<int> DiastoleListe(List<double>)	
Parameter:	List<double>
Returværdi:	List<int>
Beskrivelse:	Metoden returnerer en liste med diastoler ved at koble metoden BeregnDiastole() fra Blodtryk sammen med blodtrykslisten.

public List<int> Middeltrykliste(List<int>, List<int>, List<double>)	
Parameter:	List<int>, List<int>, List<double>
Returværdi:	List<int>
Beskrivelse:	Metoden returnerer en liste med middeltryksværdier ved at kalde metoden BeregnMiddeltryk() fra Blodtryk og koble denne sammen med systolelisten, diastolelisten og blodtrykslisten.

public GrænseVærdiDTO SystoleOP()	
Parameter:	Ingen
Returværdi:	GrænseVærdiDTO
Beskrivelse:	Metoden kalder den nedre og øvre grænseværdi og gør begge grænseværdier én større. Metoden returnerer de nye grænseværdier med syntaksen fra GrænseVærdiDTO.

public GrænseVærdiDTO SystoleNED()	
Parameter:	Ingen
Returværdi:	GrænseVærdiDTO
Beskrivelse:	Metoden kalder den nedre og øvre grænseværdi og gør begge grænseværdier én mindre. Metoden returnerer de nye grænseværdier med syntaksen fra GrænseVærdiDTO.

public GrænseVærdiDTO DiastoleOP()	
Parameter:	Ingen
Returværdi:	GrænseVærdiDTO
Beskrivelse:	Metoden kalder den nedre og øvre grænseværdi og gør begge grænseværdier én større. Metoden returnerer de nye grænseværdier med syntaksen fra GrænseVærdiDTO.

public GrænseVærdiDTO DiastoleNED()	
Parameter:	Ingen
Returværdi:	GrænseVærdiDTO
Beskrivelse:	Metoden kalder den nedre og øvre grænseværdi og gør begge grænseværdier én mindre. Metoden returnerer de nye grænseværdier med syntaksen fra GrænseVærdiDTO.

public GrænseVærdiDTO HentSystoleGrænseværdier()	
Parameter:	Ingen
Returværdi:	GrænseVærdiDTO
Beskrivelse:	Metoden returnerer grænseværdierne for systole.

public GrænseVærdiDTO HentDiastoleGrænseværdier()	
Parameter:	Ingen
Returværdi:	GrænseVærdiDTO
Beskrivelse:	Metoden returnerer grænseværdierne for diastole.

public void setNulpunktsJustering(double)	
Parameter:	double
Returværdi:	Ingen
Beskrivelse:	Metoden sætter nulpunkt værdien til et double tal.

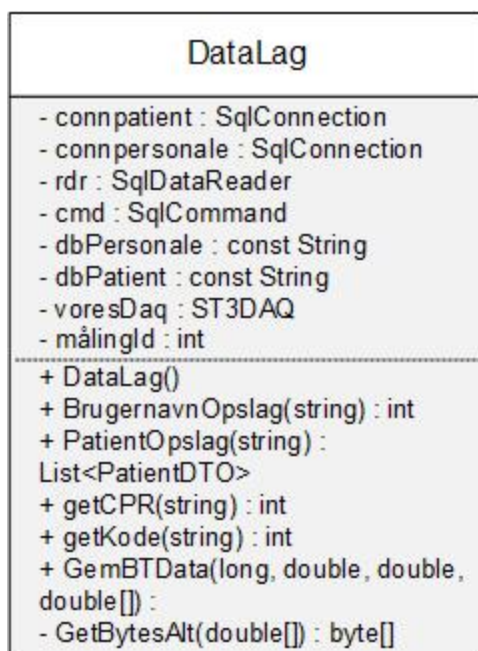
public void Attach(IObserver)	
Parameter:	IObserver
Returværdi:	Ingen
Beskrivelse:	Metoden tilføjer en observer til IObserver.

public void Detach(IObserver)	
Parameter:	IObserver
Returværdi:	Ingen
Beskrivelse:	Metoden fjerner en observer fra IObserver.

public void Notify(double[])	
Parameter:	double[]
Returværdi:	Ingen
Beskrivelse:	Metoden kigger på hver observer i IObserver og kalder her metoden Update().

public void Update(double[])	
Parameter:	double[]
Returværdi:	Ingen
Beskrivelse:	Metoden tilføjer et double array til blodtryksværdiliste og BTDataTilDB der bruges til at gemme data. Denne metode kalder metoden GemBTData() fra DataLag og gemmer når BTDataTilDB listen indeholder 1000 elementer og ryder derefter listen for denne kan tage 1000 nye elementer.

DataLag



Figur 3.39: Klassediagram for DataLag.

public DataLag()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden skaber forbindelse til de to databaser der benyttes og laver forbindelse til ST3DAQ.

public int BrugernavnOpslag(string)	
Parameter:	string
Returværdi:	int
Beskrivelse:	Metoden kalder en metode der åbner Personale databasen, og oprettet en SQL kommando, hvor der hentes alt fra Personale tabellen hvor brugernavn er lig med brugernavn parameteren. En metode til at udføre kommando kaldes. Metoden undersøger om brugernavnet passer, lukker databasen og returnerer svaret.

public List<PatientDTO> PatientOpslag(string)	
Parameter:	string
Returværdi:	List<PatientDTO>
Beskrivelse:	Metoden laver en patientliste der indholder syntaksen fra PatientDTO. Kaldet en metode der åbner Patient databasen, og oprettet en SQL kommando, hvor der hentes alt fra Patient tabellen hvor den sundhedsfaglige er lig med brugernavn parameteren. En metode til at udføre kommando kaldes. Metoden undersøger om der findes en patient med tilkobling til den sundhedsfaglige og tilføjer CPR og Navn til patientlisten. Metoden lukker databasen og returnerer patientlisten.

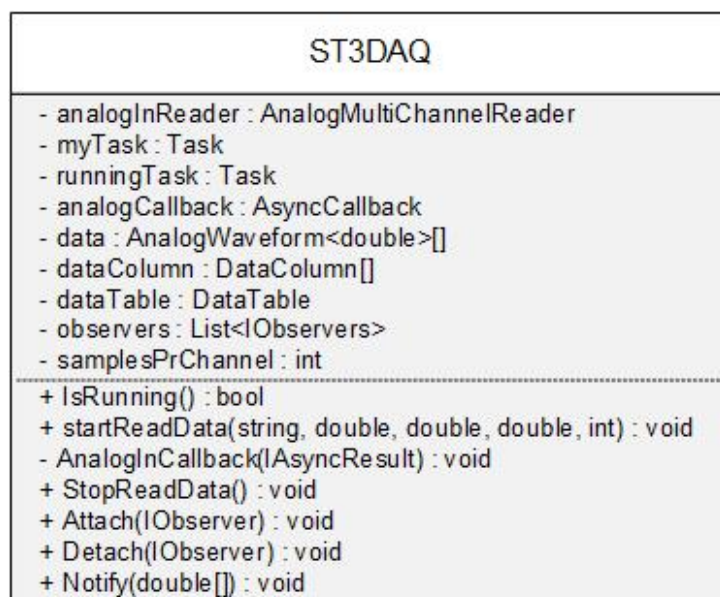
public int getCPR(string)	
Parameter:	string
Returværdi:	int
Beskrivelse:	Metoden kalder en metode der åbner Patient databasen, og oprettet en SQL kommando, hvor CPR hentes fra Patient tabellen hvor Navn er lig med patient parameteren. En metode til at udføre kommandoen kaldes. Metoden henter CPR for patienten, lukker databasen og returnerer CPR.

public int getKode(string)	
Parameter:	string
Returværdi:	int
Beskrivelse:	Metoden kalder en metode der åbner Patient databasen, og oprettet en SQL kommando, hvor Adgangskode hentes fra Personale tabellen hvor Brugernavn er lig med brugernavn parameteren. En metode til at udføre kommandoen kaldes. Metoden henter koden for den sundhedsfaglige, lukker databasen og returnerer koden.

public void GemBTData(long, double, double, double[])	
Parameter:	long, double, double, double[]
Returværdi:	Ingen
Beskrivelse:	Metoden kalder en metode der åbner Patient databasen, og oprettet en SQL kommando, hvor CPR, Dato, KalibreringsTalog NulpunktsjusteringsTal indsættes i Målinger tabellen. En metode til at udføre kommandoen kaldes. Metoden lukker databasen. Metoden kalder en metode der åbner Patient databasen igen, og oprettet en SQL kommando, hvor BTdata og MålingId indsættes i BT-data tabellen. En metode til at udføre kommandoen kaldes. Metoden lukker databasen.

private byte[] GetBytesAlt(double[])	
Parameter:	double[]
Returværdi:	byte[]
Beskrivelse:	Metoden konverterer double array til byte array. Bruges til at gemme data i SQL-database. Metoden bruges i GemBTData() metoden.

ST3DAQ



Figur 3.40: Klassediagram for ST3DAQ.

public ST3DAQ()	
Parameter:	Ingen
Returværdi:	Ingen
Beskrivelse:	Metoden henter en liste af observers fra IObservers.

public bool IsRunning()	
Parameter:	Ingen
Returværdi:	bool
Beskrivelse:	Metoden tjekker om tasken kører. Hvis den kører returnerer den true ellers returnerer denne false.

public void startReadData(string, double, double, double, int)	
Parameter:	string, double, double, double, int
Returværdi:	Ingen
Beskrivelse:	Denne metode starter for en ny opgave, hvilken henter data fra NI-DAQ. Data hentes ved at bruge metoderne koblet til National Instruments ved at kalde Task().

private void AnalogInCallback(IAsyncResult)	
Parameter:	IAsyncResult
Returværdi:	Ingen
Beskrivelse:	Metoden er en delegate, der tjekker om NI-DAQ henter data og laver et asynkront kald. Tilføjer alle data fra NI-DAQ i et array. Dette array bliver notificeret med metoden Notify(). Efter stoppes opgaven startet i startReadData().

public void Attach(IObserver)	
Parameter:	IObserver
Returværdi:	Ingen
Beskrivelse:	Metoden tilføjer en observer til IObserver.

public void Detach(IObserver)	
Parameter:	IObserver
Returværdi:	Ingen
Beskrivelse:	Metoden fjerner en observer fra IObserver.

public void Notify(double[])	
Parameter:	double[]
Returværdi:	Ingen
Beskrivelse:	Metoden kigger på hver observer i IObserver og kalder her metoden Update().

Implementering

Kalibrering

Kalibreringen er blevet implementeret sådan at der indtastes en værdi for trykket som vandsøjlen leverer i mmHg og en værdi for spændingen i volt, hvilken kan aflæses fra waveforms eller måles vha. et multimeter. Disse to værdier bliver herefter sendt ned i software klassen Kalibrering, hvor en kalibreringsværdi/omsætningsværdi kan udregnes ved formlen:

$$X \left[\frac{mmHg}{V} \right] = \frac{pressure [mmHg]}{voltage [V]} \quad (3.12)$$

Den værdi der her udregnes benyttes herefter til at omregne den spænding i volt, der indsendes i systemet, til tryk i mmHg.

$$voltage [V] \cdot X \left[\frac{mmHg}{V} \right] = pressure [mmHg] \quad (3.13)$$

Hermed ganges kalibreringsværdien på samtlige værdier i den liste af data der kommer fra DAQ'en (Blodtryksværdiliste).

Nulpunktsjustering

Nulpunktsjusteringen foregår ved at transduceren er tilsluttet og måler det atmosfæriske tryk, denne værdi skal herefter trækkes fra samtlige værdier for blodtrykket fra DAQ'en. Denne værdi skal derfor trækkes fra inden kalibreringsværdien ganges på. Måden hvormed nulpunktsjusteringen foregår på, er ved at transduceren er tilsluttet og vha. waveforms kan det atmosfæriske tryk aflæses, denne værdi aflæses som en spænding. Denne værdi indtastes på hovedskærmen, hvorefter nulpunktsjusteringen kan startes. Systemet går herefter ind i klassen Nulpunkts_justering, hvor følgende formel bruges til at beregne den værdi der skal benyttes videre:

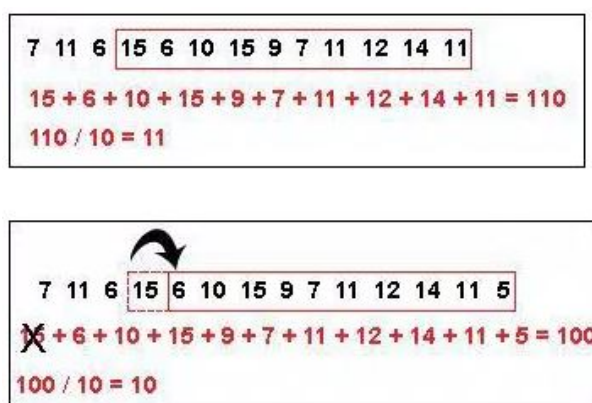
$$V_{zero} [V] = V_{read} [V] - pressure [V] \quad (3.14)$$

Det er denne værdi der herefter bliver kalibreret, for derefter at kunne blive vist som en graf for blodtrykket i mmHg.

Digitalt filter

Det digitale filter der er blevet implementeret i projektet er af typen et glidende middelværdi filter. I et glidende middelværdi filter, lægges der et vindue oven på ens talværdi, i dette vindue tager man gennemsnittet af alle værdien. [1] Vinduet er her betegnelsen for et bestemt tidsinterval.

I projektet er der valgt længden ti på vinduet, det betyder at for hver tiende tal i vores blodtryksliste, bliver der udregnet gennemsnittet af. Herefter rykker vinduet sig til højre og udregner for de næste ti tal i blodtrykslisten. Blodtrykslisten er listen af blodtryksværdier, der bliver indlæst af DAQ'en.



Figur 3.41: Digital filter ramme. Denne viser hvordan rammen bestemmer gennemsnittet og rykker én hver gang.

Længden af vinduet på ti er valgt, da der bliver vist ti tal fra blodtrykslisten på en graf på brugergrænsefladen. Glidende middelværdifilter kan beskrives med formlen:

$$y[n] = \frac{1}{M} \cdot \sum_{k=0}^M x[n-k], \quad (3.15)$$

$$b_k = \frac{1}{M} \quad (3.16)$$

M er de forgående samples for signalet, n er de nuværende værdier af blodtrykket og k er koefficienterne. Filteret udglatte signalets høje peak værdier, da filteret er et lavpasfilter.

Beregning af systolisk, diastolisk og middeltryks værdi

Algoritmerne for udregning af systole, diastole og middeltryk er alle implementeret i klassen Blodtryk.

Systolen er toppunkterne i blodtryksdataene. Dette kræver dog at man skrive en kort algoritme, så man kan holde styr på hvor langt man er kommet i blodtrykslisten og hvor mange systole værdier der er fundet.

Der er valgt at lave en baseline, som er den grænseværdi der skal overskrides, inden man tjekker efter for toppunkter. Grænseværdien baseline er sat til 100, da den normale værdi for systolen ligger omkring 120 mmHg.

Algoritmen BeregnSystole starter med en for lykke, hvor man har en tæller "i", der starter

fra nul og tæller op til længden af blodtrykslisten. Inden i for lykken er der en if lykke, som tjekker om indeks "i" i blodtrykslisten er større end den baseline der er sat. Hvis indeks "i" i blodtrykslisten er større end baseline, vides der at grafen er på vej opad. Herved sættes der en ny baseline, som er lig med værdien for indeks "i" i blodtrykslisten. Der bliver lagt én til værdien baselineoverskredet.

Hvis indeks "i" i blodtrykslisten er mindre end baseline og baselineoverskredet er større end 1, vides der at grafen for blodtrykslisten falder. Når blodtrykslisten falder vides der at den forgående indeks i blodtrykslisten er toppunktet. Der bliver gemt det forgående indeks i blodtrykslisten i en liste der hedder toppunkter. Listen af toppunkterlisten med værdier, er derfor listen af systole værdierne. Baselineoverskredet sættes lig nul, samtidig med at baselinen sættes lig default værdien, der var 100 mmHg. Algoritmen returnerer en liste af toppunkter.

Diastolen er minimumspunkterne i blodtrykslisten. For at finde minimumspunkterne i listen, er der implementeret en algoritme kaldet BeregnDiastole, som finder minimumspunkterne og gemmer dem i en liste.

Algoritmen er opbygget på samme måde som med BeregnSystole, med en lille ændring af at algoritmen skal finde minimumspunkter i stedet for toppunkter.

Der sættes en baseline værdi, der er den værdi der skal tjekkes for om indekset i blodtrykslisten er mindre ind. Default værdien for baseline er sat 90 mmHg, for at være sikker på at indekset i blodtrykslisten finder sig neden for denne værdi.

Algoritmen BeregnDiastole starter med en for lykke, hvor man har en tæller "i", der starter fra nul og tæller op til længden af blodtrykslisten. Inden i for lykken er der en if lykke, som tjekker om indeks "i" i blodtrykslisten er mindre end den baseline der sat. Hvis indeks "i" i blodtrykslisten er mindre end baseline, vides der at grafen er på vej opad. Herved sættes der en ny baseline, som er lig med værdien for indeks "i" i blodtrykslisten. Der bliver lagt én til værdien baselineoverskredet.

Hvis indeks "i" i blodtrykslisten er større end baseline og baselineoverskredet er større end 1, vides der at grafen for blodtrykslisten stiger. Når blodtrykslisten stiger vides der at den forgående indeks i blodtrykslisten er minimumspunkter. Der bliver gemt det forgående indeks i blodtrykslisten i en liste der hedder minimumspunkter. Listen af minimumspunkter med værdier, er derfor listen af diastole værdierne. Baselineoverskredet sættes lig nul, samtidig med at baselinen sættes lig default værdien, der var 90 mmHg.

Algoritmen til udregning af middeltrykket, udregner middeltrykket med [2] formelen:

$$Middeltryk = \frac{2}{3} \cdot Diastolisketryk + \frac{1}{3} \cdot Systolisketryk \quad (3.17)$$

Algoritmen middeltryk starter med en for lykke med en tæller "i", der starter fra nul og tæller op til og med længden af blodtrykslisten. Inde i for lykken udregnes ligningen for middeltrykket for indeks "i", med systolelisten for indeks "i" og diastolelisten for indeks "i". Det udregnede middeltryk gemmes i en liste kaldet middeltrykliste.

EKG-signal

Implementeringen af EKG-signalet er i dette projekt ikke sket. Dette er ikke implementeret, da der ikke har været tid til dette. Hvis der dog havde været tid til dette, ville Semesterprojekt

2: EKG diagnostik, været blevet benyttet til at få grundstrukturen til at se hvordan denne implementering skulle foregå.

Iltmætning

Desuden blev ilt saturation ikke implementeret. Dermed bliver iltmætningen ikke vist for patienten. Dette er ikke blevet implementeret på grund af flere faktorer: Der er ikke nok tid, der er ikke den fornødne viden til at få implementeret dette og der haves ikke de nødvendige materialer.

Observer mønster med PUSH

Observer mønsteret benyttes i dette projekt for at sørge for at data bliver sendt igennem alle lagene på en fornuftig og kontinuerlig måde.

Subject kan være en abstrakt klasse eller et interface, der har tre metoder; en metode der tilknytter en observer, som gerne vil modtage opdateringer, en metode til at fjerne observeren og en notify metode. Notify metoden giver informationen til observeren om ændringer.

Observer er et interface, som skal implementeres af alle de klasser, hvilke skal have besked, når der sker en ændring. Der bruges en observer, når der skal ændres i et objekt og det ikke vides hvor mange objekter der skal ændres i. Observer modtager besked om ændringer fra subjectet, dette kan ske enten ved PUSH eller PULL mønsteret.

PULL fungerer ved at subjectet giver besked til observeren om at der er ændringer. Observeren sender besked til subjectet om at modtage ændringerne. Subjectet sender herefter ændringen til observeren. PUSH fungerer sådan at subjectet giver besked til observeren, om at der er sket en ændring, samtidig med at den sender dataene med op. Data bliver altså i PUSH mønster skubbet op, hvor data i PULL trækkes op. I dette projekt er det PUSH mønsteret der er valgt at arbejde med.

I projektet er der valgt at implementere to interfaces; IObserver og ISubject. Disse to interfaces er implementeret i PUSH mønsteret, der sørger for at der hele tiden sendes data op, når der er en ny opdatering.

ISubject består af tre metoder. En metode til Attach en eller flere observer. En anden metode til at fjerne en eller flere observer. Den tredje metode Notify informerer observerne om, at der er sket en opdatering. Notify metoden sender en liste med data op til observeren, det ses heraf at PUSH mønsteret benyttes.

I projektet er der valgt at skubbe data fra datalaget til logiklaget, og derefter fra logiklaget til præsentationslaget.

Der bliver derfor i projektet implementeret to ISubjects og to IObservers. Datalaget fungerer som subject for logiklaget, som er observer til datalaget. Så hver gang der bliver læste tal ind af DAQ'en, skubber datalaget disse data videre til logiklaget.

Logiklaget er subject for præsentationslaget. Præsentationslaget er observer for logiklaget, og får derfor skubbet data op fra logiklaget.

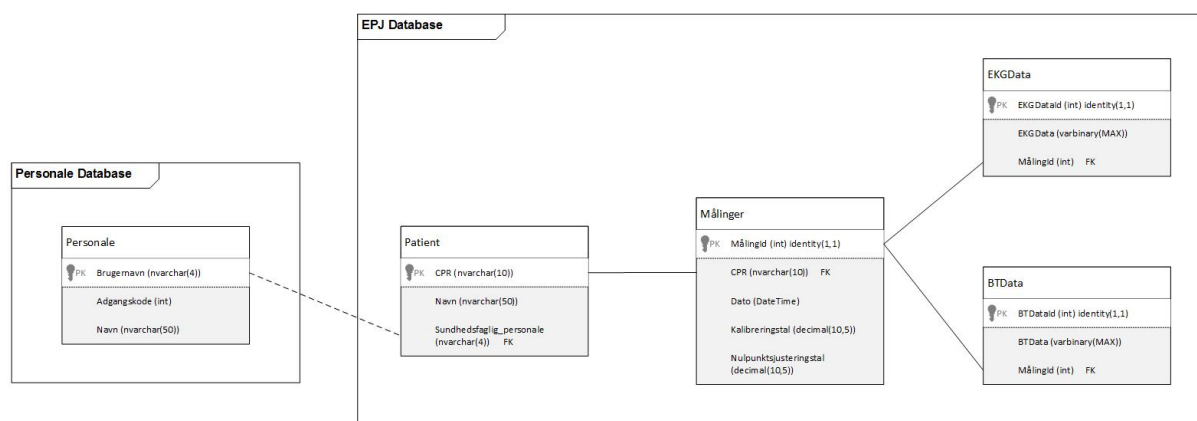
Queue

For at være sikker på at den data, der er blevet nulpunktsjusteret og kalibreret bliver sendt korrekt op til præsentationslaget bliver der benyttet en kø metode: Queue Class [3]. Dette er en klasse der er indbygget i Visual Studio. Denne klasse har to metoder, hvilke benyttes til at sende data ind i den rigtige ende, hvorefter data fjernes fra den anden ende når køen er fuld.

Disse to metoder er Enqueue(object) og Dequeue(). Enqueue(object) er metoden, hvilken tilføjer et object til køen. Dette object er data fra Blodtrykssværdiliste. Metoden ligger i en for-løkke, sådan at data kommer ind sådan som de indlæses fra DAQ'en.

Dequeue() er metoden, som fjerner et object fra den anden ende, og tager den første værdi i rækken og fjerner. Denne metode ligger ligeledes i for-løkken, hvorunder der er if/else metoder. Dette sørger for at køen er fuld inden der bliver fjernet en værdi fra den anden ende og dermed sørger for at de to metoder følges ad.

Databaser



Figur 3.42: Databaserne; Personale database og EPJ database. Her ses sammenhængen mellem tabellerne i databasen, og hvilken data der er under hver tabel.

Dette afsnit vil beskrive opbygningen af databaser og tabellerne. Da der er valgt at tage udgangspunkt i et brugsscenarie, afspejler datamodelleringen i databasen dette scenarie. Det er vigtigt at bemærke at der er en regulær opdeling af databaser og skemaer, realiseret som beskrevet nedenfor.

Personale database

Personale databasen bliver brugt til at tilgå data for personalet. Her er der tale om brugernavn og password. Disse data er lagt i databasen på forhånd, så det er altså en login-database.

Database-adresse: webhotel10.F15ST2ITS2201404669.db__ owner

EPJ database

EPJ databasen bliver brugt til at tilgå data for patienten. Her tænkes på CPR/Navn. Ligeledes gemmes der i denne database målinger, samt metadata for at kvalificere disse data. Der kunne udvides med mange flere metadata, specielt i forhold til afsnittet om fremtidigt arbejde "Datawarehouse". Det bør bemærkes der i databasen er gjort klar til EKG-data, selvom det ikke er implementeret. Patient-tabellen modsvarer vores version af en EPJ-database, hvor BTData-tabellen repræsenterer målingerne foretaget af vores system.

Database-adresse: webhotel10.F15ST2ITS2201405838.db__ owner

Opdeling

Man ville normalt have en "Foreign Key-constraint fra Patient-tabellen til Personal-tabellen.

Men da de befinder sig i to forskellige skemaer, der simulerer to forskellige databaser, så er det ikke en fornuftig løsning. Man ville i stedet benytte sig af en "trigger" der kan håndtere en evt. Fejl. Det er ikke en korrekt løsning og vi har valgt ikke at implementere den, men det er bedre end at "satse" på data stemmer overens. Man ville sandsynligvis løse problemet i software, eller hvis begge tabeller findes i samme skema.

Et eksempel på sådan en trigger vil se sådan her ud;

```

Create Trigger dbo.PatientPersonaleTrigger ON [dbo].[Patient] After Insert
As
Begin
    If NOT Exists(select Brugernavn from webhotel10.F15ST2ITS2201404669.db_owner.Personale
where Brugernavn in (Select Sundhedsfaglig_personale from dbo.Patient))
    BEGIN
        --Her ville vi håndtere fejlen
    END
END

```

Figur 3.43: Eksempel på en trigger.

Unittest

Det er blevet besluttet at lave unittest på softwaren. Ved unit-test forstås at man tester hver enkelt komponent i koden, i objektorienteret kode vil det sige metoderne. Man sikrer sig at metoderne returnerer det man forventer. En unit test bør foregå løbende lige så frit som koden bliver udviklet, så kan man fokusere på de del-elementer som ikke virker efter hensigten.

I dette projekt valgte man at lave unit test efter sidste iteration af koden, da de sidste versioner af softwaren ændrede sig meget, så alle unittests skulle laves forfra. Grundlaget for mange metoder ændrede sig løbende. Disse unit-tests blev lavet i koden ved at sætte breakpoints og følge metoden til ende og evaluere på hvorvidt metoden opførte sig efter hensigten. Unit-tests er blevet foretaget i koden, man har skrevet kommentarer til de metoder der er blevet testet.

Code review

Desuden er der blevet foretaget omfattende code-review, som en er en gennemgang af koden og de metodikker og formatering der er benyttet. Code-review er blevet foretaget af en person fra hardware-holdet. Gennemgangen har resulteret i nogle formateringsændringer i koden, ligeledes er kodesegmenter der ikke bliver brugt kommenteret ud.

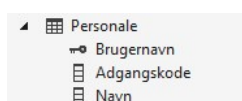
Databaser og test

Databasetest er blevet foretaget i det tilgængelige testmiljø, det vil sige serverne er placeret på IHA og der kræves VPN for at kunne tilgå dem. Der er i projektet benyttet to databaser for at simulere separat personale og EPJ databaser.

Personale database

Bliver brugt til at tilgå data for personalet. Her er der tale om brugernavn og password. Disse data er lagt i databasen på forhånd.

Database-adresse: webhotel10.F15ST2ITS2201404669.db__owner



Personale
Brugernavn
Adgangskode
Navn

Figur 3.44: Personale database med kolonner

EPJ database

Bliver brugt til at tilgå data for patienten. Her tænkes på CPR/Navn. Ligeledes gemmes der i denne database målinger, samt metadata for at kvalificere disse data. Der kunne udvides med mange flere metadata, specielt i forhold til afsnittet om fremtidigt arbejde "Datawarehouse". Det bør bemærkes der i databasen er gjort klar til EKG-data, selvom det ikke er implementeret. Database-adresse: webhotel10.F15ST2ITS2201405838.db__owner



Figur 3.45: EPJ database med kolonner.

Test

Testen af databaser er foregået løbende. Da softwareløsningen gemmer data kontinuert, så kan man altså tjekke om data bliver gemt ved at tjekke tabellernes indhold løbende.

	MålingId	CPR	Dato	KalibreringsTal	Nulpunktsjust...
	29	1234567890	08-12-2015 12:49:54	1,00000	-2,00000
	30	1234567890	08-12-2015 12:58:26	1,00000	-2,00000
	31	1234567890	08-12-2015 13:01:15	1,00000	-2,00000
	32	1234567890	08-12-2015 13:03:12	1,00000	0,00000
	33	1234567890	08-12-2015 13:04:02	1,00000	-2,00000
	34	1234567890	08-12-2015 13:12:11	1,00000	-1,00000
	35	2345678901	08-12-2015 13:13:09	1,00000	-1,00000
	36	1234567890	08-12-2015 13:44:11	1,00000	-1,00000
	37	1234567890	08-12-2015 13:46:42	1,00000	-1,00000
	38	1234567890	08-12-2015 14:03:53	1,00000	-1,00000
	39	1234567890	08-12-2015 14:11:12	1,00000	-1,00000
	40	1234567890	08-12-2015 15:02:27	0,00000	-1,00000
	41	1234567890	08-12-2015 15:05:41	50,00000	-1,00000
	42	1234567890	08-12-2015 15:06:46	50,00000	1,00000
	43	1234567890	09-12-2015 09:31:29	50,00000	-1,00000

Figur 3.46: Udsnit af data gemt i Målinger.

	BTdataId	BTdata	MålingId
	820	<Binary data>	30
	821	<Binary data>	31
	822	<Binary data>	31
	823	<Binary data>	31
	824	<Binary data>	32
	825	<Binary data>	32
	826	<Binary data>	32
	827	<Binary data>	32
	828	<Binary data>	32
	829	<Binary data>	32
	830	<Binary data>	32
	831	<Binary data>	32
	832	<Binary data>	33
	833	<Binary data>	33
	834	<Binary data>	33

Figur 3.47: Udsnit af data gemt i BTData.

3.3 Integrationstest

Kapitel 4

Accepttest

4.1 Indledning

Accepttestene skal vise om kravene der er opstillet for blodtryksmålersystemet i kravspecifikationen lever op til de standarder der er sat op for at produktet aktivt kan indgå i en hverdag på sygehusene.

Accepttestene er en opfølgning af kravspecifikationen, hvilket sikre at alle krav er overholdt og dermed opnået.

Når der i feltet Godkendt er skrevet initialer for den der har udført testen, samt datoen for testens udførelse, betyder det at testen er godkendt.

Igennem Arkitektur og design er systemet blevet identificeret og specificeret, sådan at vides hvilke dele systemet består af og hvordan dette skal præsenteres på user interfaces.

4.2 Accepttest for funktionelle krav

Opstilling

Billede indsættes - findes ikke endnu

Tabel 4.1: Accepttest for Use case 1

Use case 1: Kalibrer apparat	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Normalforløb:	Indtast den aflæste værdi for spændingen fra waveforms, og aflæs trykket fra vandsøjles skema for det af målepunkterne fra vandsøjlen der måles på.	Blodtryksmåler-systemet er tændt, tilsluttet og startet, og tilsluttet kalibreringsudstyret (Analog Discovery og Waveforms).	Service-medarbejder foretager kalibrering. Værdierne indtastes korrekt.	
	Tryk på "Kalibrering"	Blodtryksmåler-systemet er tændt og tilsluttet, og tilsluttet kalibreringsudstyret (Analog Discovery og Waveforms) og værdierne indtastet.	Systemet beregner kalibreringsværdien og kalibrerer systemet, resultatet af beregningen vises på startskærmen	IKKE TESTBAR

Tabel 4.2: Accepttest for Use case 1

Use case 1: Kalibrer apparat	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
[Undtagelse 1: Undlad kalibrering]	Undlad kalibrering og fortsæt til Use case 2.	Blodtryksmåler-systemet er tilsluttet og tændt	Kalibreringsværdien hentes, fra den forrige måling, fra EPJ databasen.	

Det forudsættes for følgende Use cases (Use case 2, 3 og 4) at systemet er tilsluttet korrekt, port til DAQ er bestemt, at der er en VPN-forbindelse til "ASE IHA VPN", desuden skal der være en forbindelse til "webhotel10.F15ST2ITS2201404669.db_owner" (Personaledatabasen) og "webhotel10.F15ST2ITS2201405838.db_owner" (EPJ database).

Tabel 4.3: Accepttest for Use case 2

Use case 2: Foretag måling	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Normalforløb:	Indtast brugernavn "anpe" og kode "1234"	Port valgt. VPN-forbindelse, Personale database og EPJ database er tilsluttet korrekt	Korrekt indtastning fuldendt	
	Tryk "Log ind"	Port valgt. VPN, Personale database og EPJ database er tilsluttet korrekt	Besked: "Logget på" og den sundhedsfaglige er dermed logget på	
	Tryk på patient dropdown på startskærm	En sundhedsfaglig er logget på	Liste med patienter kommer frem	
	Vælg patienten "Arne Jensen"	Den sundhedsfaglige er logget på	Nyt vindue kommer frem: Hovedskærmen	
	Indsend atmosfærisk tryk igennem transducere, aflæs værdien på waveforms og indtast denne korrekt	Patient valgt og blodtryksmåler-systemet (transducere) er tilsluttet	Nulpunktsjusteringsværdien er indtastet korrekt	
	Tryk på "Nulpunktsjustering"	Værdien er indtastet korrekt	Systemet starter nulpunktsjusteringen. Systemet tilpasser grafen.	
	Tryk på "Tænd"	Systemet er nulpunktjusteret	Systemet indhenter data fra transducere og starter timer på hovedskærm. EKG og arteriestryk præsenteres kontinuert på hver sin graf. Puls, systole, diastole og middeltryk vises som talværdier på hovedskærmen. Data gemmes automatisk kontinuert i EPJ database	

Tabel 4.4: Accepttest for Use case 2

Use case 2: Foretag måling	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Undtagelse 1: Brugernavn og/eller kode indtastet forkert	Indtast brugernavn "efgh"og kode "1234"	Port valgt. VPN, Personale database og EPJ database er tilsluttet korrekt	Forkert kombination indtastet	
	Tryk "Log ind"	Port valgt. VPN, Personale database og EPJ database er tilsluttet korrekt.	Besked: "Brugernavn og/eller kode indtastet forkert"	

Tabel 4.5: Accepttest for Use case 2

Use case 2: Foretag måling	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Udvidelse 1: Slå digitalt filter til/fra:	Tryk på "Digitalt filter OFF"radiobutton	Signalet er startet og filteret er sat til pr. default	Systemet slår det digitale filter fra. De to sinus signaler (XX Hz og YY Hz) er indsendt	
	Tryk på "Digitalt filter ON"radiobutton	Signalet er startet og "Digitalt filter OFF"radiobutton er valgt	Systemet slår det digitale filter til. Signalet ses udglattet.	

Tabel 4.6: Accepttest for Use case 2

Use case 2: Foretag måling	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Udvidelse 2: Juster systo- lens/diastolens grænseværdi	Tryk på "Systole op/"Diastole op"	Signalet er startet	Grænseværdien ændres 1 mmHg op og intervallet vises på hovedskærmen	
	Tryk på "Systole ned/"Diastole ned"	Signalet er startet	Grænseværdien ændres 1 mmHg ned og intervallet vises på hovedskærmen	

Tabel 4.7: Accepttest for Use case 3

Use case 3: Alarmér	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Normalforløb:	Grænseværdi overskrides	Signalet er startet	Alarm starter med lyd og "Udsæt alarm"knappen blinker skiftevis mellem rød og hvid	
	Grænseværdien er ikke overskredet mere	Alarmeren igangsat	Alarm lyden stopper og tallet skifter farve tilbage til hvid.	

Tabel 4.8: Accepttest for Use case 3

Use case 3: Alarmér	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Udvidelse 1: Anden grænseværdi overskrides	Endnu en grænseværdi overskrides	Signalet er er startet og en alarm er startet	Lyd fra første alarm fortsætter.	

Tabel 4.9: Accepttest for Use case 3

Use case 3: Alarmér	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Udvidelse 2: Udsæt alarm	Tryk på "Udsæt alarm"knappen	Alarmering er startet	Systemet stopper alarmens lyd i et minut, Udsæt alarm knappen fortsætter med at blinke skiftevis rød og hvid indtil alarmeren ikke længere er overskredet.	

Tabel 4.10: Accepttest for Use case 4

Use case 4: Stop måling	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Normalforløb:	Tryk på "Sluk"	Målingen er foretaget	Målingen, signalet og timer på hovedskærmen stopper	
	Tryk på "Log ud"	Signalet er stoppet	Pop-up vindue kommer op: "Er du sikker?"	
	Tryk "Ja"	Signalet og målingen er stoppet	Startskærmen kommer frem og sundhedsfaglig personale kan logge på igen sådan at ny måling kan foretages	

Tabel 4.11: Accepttest for Use case 4

Use case 4: Stop måling	Test	Prækondition	Forventet resultat	Godkendt/ kommentar
Undtagelse 1: Tryk på "Nej"	Tryk "Nej"	Signalet og målingen er stoppet	Kommer tilbage til hovedskærmen	

4.3 Accepttest for ikke-funktionelle krav

Tabel 4.12: Accepttest for ikke-funktionelle krav

Krav nr.	Krav	Test	Forventet resultat	Resultat	Godkendt/kommentar
1.1	Programmet skal have et digitalt filter til udglatning af blodtrykssignal	Send to frekvenser ind (XX Hz og ZZ Hz)	Den ene (XX Hz) af de indsendte frekvenser er blevet fjernet		
1.2	Programmet skal give alarm når grænseværdier overskrides med lyd og Udsæt alarm knappen skal blinke skiftevis rød og hvid.	Overskrid en grænseværdi og tjek alarmering	Alarmen starter		
1.3	Programmet skal kunne gemme blodtrykssignalet i en database	Indsend signal og gå ind i databasen og se værdier	Der ligger værdier i databasen		
2.1	Programmet skal have to window form: startskærm, der fungerer som EPJ systemet, og hovedskærm, som fungerer som selve blodtryksmåleren	Start program og tjek dette	Der er to window forms		
2.2	Programmet skal have en "Log ind"knapp på startskærmen	Start program og tjek startskærm	Startskærmen har en "Log ind"knapp		
2.3	Programmet skal have en "Kalibre-ring"knapp på startskærmen	Start program og tjek startskærm	Startskærmen har en "Kalibre-ring"knapp		

2.4	Sundhedsfaglig personale skal kunne ændre "device/enhedsnavn" i dropdown på startskærm	Start program og tjek startskærm	Der er en opsætnings dropdown på startskærmen, hvor device/enhedsnavn kan ændres.		
2.5	Programmet skal indeholde en dropdown, hvor patienten kan vælges på startskærmen	Start program, log på og tjek startskærm	Startskærmen har en dropdown med patienter		
2.6	Programmet skal have en "Nulpunkts indstilling" knap på hovedskærmen	Start program og tjek hovedskærm	Der er en "Nulpunkts indstilling" knap på hovedskærmen		
2.7	Programmet skal have en knap, til at slå det digitale filter fra og til, på hovedskærmen	Start program og tjek hovedskærm	Der er en "Digital filter" knap på hovedskærmen		
2.8	Programmet skal have knapper, til at justere systolisk og diastolisk grænseværdiintervaller op og ned, på hovedskærmen	Start program og tjek hovedskærm	Der er ialt fire knapper, som justerer grænseværdierne på hovedskærmen		
2.9	Programmet skal have en "Udsæt alarm" knap på hovedskærmen	Start program og tjek hovedskærm	Der er en "Udsæt alarm" på hovedskærmen		
2.10	Programmet skal have en "Tænd" knap på hovedskærmen	Start program og tjek hovedskærm	Hovedskærmen har en "Tænd" knap		

2.11	Programmet skal have en "Sluk"knapp på hovedskærmen	Tjek hovedskærm	Hovedskærmen har en "Sluk"knapp		
2.12	Programmet skal have en "Log ud"knapp på hovedskærmen	Start program og tjek hovedskærm	Der er en "Log ud"knapp på hovedskærmen		
2.13	Teksten på startskærmen skal kunne aflæses fra 2 meters afstand med en synsstyrke i intervallet ± 1	10 personer med synsstyrke i intervallet ± 1 skal teste startskærmen	Alle 10 personer kan læse teksten tydeligt		
2.14	Teksten og graferne på hovedskærmen skal kunne læses fra 2 meters afstand ved synsstyrke i intervallet ± 1	10 personer med synsstyrke i intervallet ± 1 skal teste hovedskærmen	Alle 10 personer kan læse grafer og teksten på hovedskærmen		
2.15	Programmet skal præsentere arterietryk kontinuert, herudover vise systolisk værdi, diastolisk værdi og middeltryk som talværdier.	Start program og start målingen	Grafen vises kontinuert og talværdierne vises på hovedskærmen.		
2.16	Programmet skal præsentere EKG og puls	Start program og start målingen.	EKG vises kontinuert og pulsen vises som talværdi på hovedskærmen.	IKKE IMPLEMENTERET	
hline 2.17	Programmet skal præsentere iltmætning som graf og talværdi.	Start program og start måling	Iltmætningens grafen vises kontinuert og iltmætning vises som talværdi på hovedskærm.		

2.18	Programmet skal præsentere grafer efter standard	Start program og tjek farver	EKG vises i lysegrøn, arterietryk vises i rød og iltmætning vises i lyseblå		
2.19	Programmet skal præsentere data i tal efter standard	Start program og tjek at talværdiernes farve er efter standard	Pulsværdien vises i lysegrøn, systolisk værdi, diastolisk værdi og middeltryk vises i rød og iltmætning vises i lyseblå.		
3.1	Ingen krav endnu				
4.1	Tiden der går før målingen af data påbegynder/vises i grafer må maksimalt være 2.0 sek.	Stopur igangsættes samtidig med at signalet tændes	Stopuret viser 2 sek. eller mindre		
4.2	Tiden der går fra at data er analyseret til at data er gemt i database må være 2.0 sek. med en tolerance på +/- 15%	-	-		
5.1	Programmet skrives i C# kode.	Tjek programmet	Programmet er skrevet i C#		
5.2	Softwareen skal være opbygget efter trelagsmodellen	Se programopbygningen	Softwareen er opbygget efter trelagsmodellen		
5.3	I softwaren benyttes Observer/Subject mønsteret	Tjek programmet og tjek observer/subject klasser	Observer og subject klasser/mønsteret benyttes		

5.4	I softwaren benyttes PUSH mønsteret	Tjek programmet	Det ses at programmet er opbygget efter PUSH mønsteret.		
6.1	Der skal være adgang til en computer med Windows 7, 8 eller 10 - computeren skal minimum have 4 GB RAM				
6.2	Der skal være adgang til en computer hvor National Instruments er installeret				

4.4 Godkendelses formular

Dato for test	
Godkendes af:	

Ved underskrivelse af dette dokument godkendes den kørte accepttest
 Sted og dato:

 Kundens underskrift

 Leverandørens underskrift

Litteratur

4.5 Referencer

Bøger

Internetsider

- [1] Dotnetwidkunal's Blog, skrevet 4. juni 2010, <https://dotnetwidkunal.wordpress.com/2010/06/04/creating-a-simple-moving-average-function-in-c/>
- [2] Blodtryk, Wikipedia.org, læst 7. december 2015, <https://da.wikipedia.org/wiki/Blodtryk>
- [3] Queue Class, msdn.microsoft.com, læst d. 8. december 2015, [https://msdn.microsoft.com/en-us/library/system.collections.queue\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.collections.queue(v=vs.110).aspx)
- [4] Sallen-Key Low-pass Filter Design Tool, brugt 9. december 2015, <http://sim.okawa-denshi.jp/en/OPseikiLowkeisan.htm>

Dokumenter