

# Softwareimplementering

3. semesterprojekt

Version 1.3

---

**Ingeniørhøjskolen Aarhus**



Vejleder: Peter Johansen

Afleveret: 17/12-2014

Studienummer	Navn	Studieretning
201371026	Camilla Lund Pedersen	ST
201371015	Charlotte Søgaard Kristensen	ST
201370750	Kasper Kronborg Larsen	ST
201370826	Kathrine Duus Kinnerup	ST
201205775	Mads Morgen Kusk	ST
201371061	Thuvaharan Karunakaran	ST

## Indholdsfortegnelse

<b>INDHOLDSFORTEGNELSE .....</b>	<b>1</b>
<b>VERSIONSHISTORIK .....</b>	<b>2</b>
<b>ORDLISTE .....</b>	<b>3</b>
<b>1.0 INDLEDNING .....</b>	<b>4</b>
<b>2.0 KLASSEDIAGRAM .....</b>	<b>4</b>
<b>3.0 KLASSE- OG METODEBESKRIVELSER .....</b>	<b>6</b>
3.1 DATA ACCES LAYER – NAMESPACE PROJEKT.DAL .....	6
3.1.1 Klasse: DAL_Datalagring .....	6
3.1.2 Klasse: DAL_Dataopsamling .....	6
3.1.3 Klasse: DAL_Nulpunktsjustering .....	7
3.2 BUSINESS LAYER – NAMESPACE PROJEKT.BL .....	8
3.2.1 Klasse: BL_Alarmer .....	8
3.2.2 Klasse: BL_Blodtryksudregning .....	9
3.2.3 Klasse: BL_Filtrering .....	10
3.2.4 Klasse: BL_ForetagOgVisMåling .....	11
3.2.5 Klasse: BL_GemCPRMåling .....	12
3.2.6 Klasse: BL_Nulpunktsjustering .....	12
3.2.7 Klasse: BL_Subject .....	13
3.3 PRESENTATION LAYER – NAMESPACE PROJEKT.PL .....	14
3.3.1 Klasse: AlarmInterface .....	14
3.3.2 Klasse: GemInterface .....	15
3.3.3 Klasse: OperatørInterface .....	15
<b>4.0 SEKVENSDIAGRAMMER .....</b>	<b>22</b>
4.1 UC1: NULPUNKTSJUSTÉR .....	22
4.2 UC2: FORETAG OG VIS MÅLING .....	23
4.3 UC3: GEM MÅLING .....	24
4.4 UC6: NULSTIL .....	25
<b>5.0 AKTIVITETSDIAGRAMMER .....</b>	<b>26</b>
5.1 METODEN NULPUNKTSJUSTER I KLASSEN BL_NULPUNKTSJUSTERING .....	26

5.2 METODEN STARTMÅLING I KLASSEN BL_FORETAGOGVISMÅLING.....	26
5.3 METODEN OMREGNSIGNALTILTRYK I KLASSEN BL_FORETAGOGVISMÅLING.....	28
5.4 METODEN FINDMIDDELBLODTRYK I KLASSEN BL_BLODTRYKSUDREGNING.....	29
5.5 METODEN CPR_TJEK I KLASSEN BL_GEMCPRMÅLING.....	29
5.6 METODEN TJEKDIASTOLISKBLODTRYK I KLASSEN BL_ALARMER .....	30
<b>6.0 ENHEDSTEST .....</b>	<b>32</b>
6.1 BL_ALARMERING .....	32
6.2 BL_BLODTRYKSUDREGNING.....	34
6.3 DAL_DATALAGRING .....	35
6.4 BL_NULPUNKTSJUSTERING .....	36
6.5 BL_GEMCPRMÅLING.....	37
6.6 DAL_NULPUNKTSJUSTERING .....	40
6.7 BL_FORETAGOGVISMÅLING.....	42
6.8 DIGITALT FILTER .....	43
<b>6.0 REFERENCELISTE .....</b>	<b>46</b>

## Versionshistorik

Version	Dato	Beskrivelse af ændringer	Initialer
1.0	12/12-14	Oprettelse af dokument. Indsat sekvensdiagrammer, aktivitetsdiagrammer, klassediagram, klasse- og metodebeskrivelser, beskrivelser af enhedstest.	CLP
1.1	13/12-14	Figurtekster, rettelser i unittest.	KDK
1.2	15/12-14	Rettelser.	CLP, KDK
1.3	15/12-14	Rettelser.	MMK, KKL

Tabel 1 Versionshistorik

## Ordliste

Task	En funktion i C# til at lave asynkrone operationer.
PL	Presentation Layer.
BL	Business Layer.
DAL	Data Access Layer.
Cirkulær buffer	Hukommelsescyklus i programmering, der tømmes og fyldes hele tiden.
Buffer	Hukommelse der er tilgængelig for tømning og fyldning.
C#	Programmeringssprog.
Namespace	Bruges i programmering til at give unik identifikation af objekter med forskellig oprindelse.
3-lagsmodel	Måde hvorpå kode inddeles for at sikre overskuelighed og effektivitet.
Visual Studio	Program til udvikling af applikationer i C# programmeringssproget.
True	En returværdi for datatypen Boolean. True indikerer et sandt udsagn.
False	En returværdi for datatypen Boolean. False indikerer et falsk udsagn.
For-løkke	I programmering er en for-løkke en måde hvorpå kodelinjer kan gentages flere gange efter hinanden.
Klasse	En overordnet del af kode indeholdende adskillige objekter samt metoder.
Metode	I programmering indeholder metoder kodelinjer som giver applikationen funktion og mulighed for behandling af data.
Peak	Et udtryk for kraftigt forhøjet amplitude ved en bestemt frekvens. Bruges i dette dokument i forbindelse med støj.

Tabel 2 Ordliste

## 1.0 Indledning

Dette dokument har til formål at dokumentere de tiltag, der er gjort under implementeringen af systemets software. Dokumentet indeholder et klassediagram med tilhørende beskrivelser, sekvensdiagrammer over de implementerede funktioner, aktivitetsdiagrammer over udvalgte metoder og beskrivelse af de enhedstest, der udført undervejs i implementeringen. I dokumentet "Softwaredesign" er der også et klassediagram og sekvensdiagrammer, men da gruppens arbejde under implementering har afvejet fra designet flere steder, er disse diagrammer ændret og opdateret i dette dokument, så det svarer til den implementerede C# kode. Årsagen til denne fremgangsmåde er beskrevet i afsnittet "Resultater og diskussion" i rapporten.

## 2.0 Klassediagram

Figur 1 viser et klassediagram over hele softwaren. I hver klasse er der opskrevet attributter og metoder, og de er alle tildelt et namespace. I programmet er der tre namespaces, Projekt.PL, Projekt.BL og Projekt.DAL, der afspejler 3-lagsmodellen. Tre klasser er ikke blevet fuldt ud implementeret, og disse vises med rødt i klassediagrammet. Årsagen til dette beskrives og diskuteres i rapporten (jf. rapporten s.45).



Figur 1 Klassediagram

## 3.0 Klasse- og metodebeskrivelser

Klasse- og metodebeskrivelserne beskriver alle klasser og metoder fra klassesdiagrammet. Beskrivelserne er inddelt efter de tre lag (med tilsvarende namespace).

### 3.1 Data Acces Layer – namespace Projekt.DAL

#### 3.1.1 Klasse: DAL\_Datalagring

**Ansvar:** Klassen har som ansvar at oprette en CSV-fil og gemme blodtryksmålingerne i denne.

**Metoder:** `public string OpretCSVFil(string CPR)`

Parametre: `string CPR`.

Returværdi: `string filePath`.

Beskrivelse: opretter en CSV-fil med stien `filePath`. Der tjekkes om filen allerede eksisterer.

`public void GemCSVFil(List<double>blodtryksmåling, string CPR)`

Parametre: `List<double>blodtryksmåling, string CPR`.

Returværdi: ingen returværdi.

Beskrivelse: udskriver blodtryksmålinger, dato og CPR ud i CSV-fil.

#### 3.1.2 Klasse: DAL\_Dataopsamling

**Ansvar:** Klassen har som ansvar at håndtere opsamlingen af data fra A/D-converteren, og lægge dem over i lister, som kan videre bearbejdes i programmet.

**Metoder:** `public List<double>Blodtryksmålinger()`

Parametre: ingen parametre.

Returværdi: `List<double>blodtryk`.

Beskrivelse: henter en liste af blodtryksværdier.

`public void OpsamlData(CancellationTokens ct)`

Parametre: `CancellationTokens ct`.

Returværdi: ingen returværdi.

Beskrivelse: opretter en task, der vha. metoden `dataToDataTable` læser data fra A/D-converter og lægger dem over i en cirkulær buffer.

```
private List<double>dataToDataTable(National.Analogwaveform<double>[] sourceArray)
```

Parametre: National.Analogwaveform<double>[] sourceArray.

Returværdi: List<double> lbm.

Beskrivelse: tæller sourceArray fra A/D-converter igennem, lægger sampleværdierne over i listen lbm og returnerer listen.

```
private void OpretDAQForbindelse()
```

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: opretter forbindelse til A/D-converter på de korrekte kanaler og spændingsområder.

```
public void StopMåling(CancellationTokenSource ct)
```

Parametre: CancellationTokenSource ct.

Returværdi: ingen returværdi.

Beskrivelse: stopper målingen ved at kalde dispose på tasken TrådHentDAQData.

### 3.1.3 Klasse: DAL\_Nulpunktsjustering

**Ansvar:** Klassen har som ansvar at udføre en dataopsamling på 100 målinger. Disse skal bruges til at finde et gennemsnitligt tryk, som der kan nulpunktjusteres ud fra.

**Metoder:** public void getATMTryk()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: opretter forbindelse til A/D-converter og opsamler 100 målinger.

```
private void dataToDataTable(NationalInstruments.AnalogWaveform<double>[] sourceArray)
```

Parametre: NationalInstruments.AnalogWaveform<double>[] sourceArray).

Returværdi: ingen returværdi.

Beskrivelse: tæller sourceArray igennem, lægger sampleværdierne over i listen currentATMtryk.



```
public List<double> getTryk()
```

Parametre: ingen parametre.

Returværdi: List<double> currentATMtryk.

Beskrivelse: returnerer listen currentATMtryk.

## 3.2 Business Layer – namespace Projekt.BL

### 3.2.1 Klasse: BL\_Alarmer

**Ansvar:** BL\_Alarmer står for alarmering når de indtastede grænseværdier overskrides. Klassen er ikke implementeret i det endelige program.

**Metoder:** public bool TjekSystoliskBlodtryk(double øvre\_grænse\_systolisk, double systolisk, double nedre\_grænse\_systolisk)

Parametre: double øvre\_grænse\_systolisk, double systolisk, double nedre\_grænse\_systolisk.

Returværdi: bool alarm\_systolisk.

Beskrivelse: tjekker om den systoliske værdi for blodtrykket overskrider den øvre og nedre grænseværdi. Hvis dette er tilfældet sættes alarm\_systolisk til true.

```
public bool TjekDiastoliskBlodtryk(double øvre_grænse_diastolisk, double diastolisk, double nedre_grænse_diastolisk)
```

Parametre: double øvre\_grænse\_diastolisk, double diastolisk, double nedre\_grænse\_diastolisk.

Returværdi: bool alarm\_diastolisk.

Beskrivelse: tjekker om den diastoliske værdi for blodtrykket overskrider den øvre og nedre grænseværdi. Hvis dette er tilfældet sættes alarm\_diastolisk til true.

### 3.2.2 Klasse: BL\_Blodtryksudregning

**Ansvar:** BL\_Blodtryksudregning har ansvaret for, at udregne blodtrykkets systoliske og diastoliske værdier samt middelblodtrykket.

**Metoder:** public double FindSystoliskBlodtryk(List<double> blodtryksmåling)

Parametre: List<double> blodtryksmåling.

Returværdi: double systolisk\_blodtryk.

Beskrivelse: finder max-værdien i listen blodtryksmåling, og returnerer denne værdi. Dette er det systoliske blodtryk.

public double FindDiastoliskBlodtryk(List<double> blodtryksmåling)

Parametre: List<double>blodtryksmåling.

Returværdi: double diastolisk\_blodtryk.

Beskrivelse: finder minimum-værdien i listen blodtryksmåling, og returnerer denne værdi. Dette er det diastoliske blodtryk.

public double FindMiddelBlodtryk(double xsystolisk\_blodtryk, double xdiastolisk\_blodtryk)

Parametre: double xsystolisk\_blodtryk, double xdiastolisk\_blodtryk.

Returværdi: double middel\_blodtryk.

Beskrivelse: udregner middelblodtrykket ud fra det systoliske og diastoliske blodtryk, og returnerer middelblodtrykket. (1)

### 3.2.3 Klasse: BL\_Filtrering

**Ansvar:** BL\_Filtrering har ansvaret for, at filtrere evt. støj på blodtrykssignalet, hvis filtreringsboksen er markeret.

**Metoder:** `public List<double> FiltrerMåling(List<double> input, List<double> output)`

Parametre: `List<double> input, List<double> output.`

Returværdi: `List<double> endelig_output.`

Beskrivelse: parameterlisterne konverteres til arrays vha. metoden `KonverterTilArray()`. Metoden `FilterFunktion()` bliver næst kaldt på disse arrays. Returværdien fra `FilterFunktion()` konverteres til listen, `endelig_ouput` som returneres.

`private void FilterFunktion(double[] x, double[] y)`

Parametre: `double[] x, double[] y.`

Returværdi: ingen returværdi.

Beskrivelse: Stiller en algoritme op, beregnet ud fra et 20.ordens lavpasfilter med en cut-off frekvens på 45Hz, genereret i Matlab, som filtrerer input arrayet og dermed danner et output array.

`private List<double> KonverterTilListe(double[] array)`

Parametre: `double[] array.`

Returværdi: `array.ToList<double>().`

Beskrivelse: Konverterer et array til en liste.

`private List<double> KonverterTilArray(List<double> liste)`

Parametre: `List<double> liste.`

Returværdi: `liste.ToArray().`

Beskrivelse: Konverterer en liste til et array.

### 3.2.4 Klasse: BL\_ForetagOgVisMåling

**Ansvar:** BL\_ForetagOgVisMåling har ansvaret for, at anvende dataen fra dataopsamlingen og styre datastrømmen fra data layer til presentation layer. Herudover fungerer klassen som mellemlid mellem brugerens input, som stop, start og filtrer, og blodtryksdata.

**Metoder:** public void StartMåling(double ATMtryk)

Parametre: double ATMtryk.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden OpsamlData() på et objekt af typen DAL\_Dataopsamling. Derefter igangsættes en task. Denne task ligger data ind i listen data, omregner listen til en trykliste vha. metoden OmregnSignalTilTryk(), sender trykliste til PL via Notify(), som er en del af observer mønsteret, og tilføjer alle målinger til listen gemListe.

public List<double>GetGemListe()

Parametre: ingen parametre.

Returværdi: List<double> gemListe.

Beskrivelse: returnerer gemListe.

public void StopMåling(CancellationTokensource ct)

Parametre: CancellationTokensource ct.

Returværdi: ingen returværdi.

Beskrivelse: stopper dataopsamlingen, og den task som tjekker om dataen har værdier.

public List<double> OmregnSignalTilTryk(List<double>spænding\_liste, double ATM\_tryk)

Parametre: List<double>spænding\_liste, double ATM\_tryk.

Returværdi: List<double>tryk\_liste.

Beskrivelse: henter spændingsdataen fra dataopsamlingen, omregner disse til tryk via omregningsfaktoren (jf. Hardwareimplementering s.23) og returnerer en liste med målinger i mmHg.

public void FiltreMåling()

Parametre: bool status

Returværdi: ingen returværdi.

Beskrivelse: denne metode er ikke implementeret, grundet design af softwaren. Dette er diskuteret i rapporten. (jf. Rapporten s. 45).

### 3.2.5 Klasse: BL\_GemCPRMåling

**Ansvar:** BL\_GemCPRMåling har ansvaret for, at validere det indtastede CPR-nummer.

**Metoder:** public void GemMåling(List<double>blodtryksmåling, string CPR)

Parametre: List<double>blodtryksmåling, string CPR.

Returværdi: ingen returværdi.

Beskrivelse: kalder GemCSVFil() på objektet gemmåling af typen DAL\_Datalagring.

private bool CPR\_gyldigt\_tjek(string patientCPR)

Parametre: string patientCPR.

Returværdi: bool.

Beskrivelse: validerer om CPR-nummeret er korrekt. Hvis det er korrekt returnerer metoden true. Hvis det ikke er et korrekt CPR-nummer, så returneres false.

### 3.2.6 Klasse: BL\_NulpunktsJuster

**Ansvar:** BL\_NulpunktsJuster har ansvaret for, at lave en nulpunktsjustering og omregne fra spænding til tryk.

**Metoder:** public double NulpunktsJuster()

Parametre: ingen parametre.

Returværdi: double gennemsnit\_ATM\_endelig.

Beskrivelse: henter data fra DAL og lægger disse over i en liste. Gennemsnittet af listen findes, og denne værdi omregnes til tryk vha. af omregningsfaktoren (jf. Hardwareimplementering s.23).

### 3.2.7 Klasse: BL\_Subject

**Ansvar:** BL\_Subject har til ansvar, at oprette observer-mønstret (jf. rapporten s.20).

**Metoder:** public void Attach (IObserver observer)

Parametre: IObserver observer.

Returværdi: ingen returværdi.

Beskrivelse: tilføjer observer til listen observers.

public void Detach (IObserver observer)

Parametre: IObserver observer.

Returværdi: ingen returværdi.

Beskrivelse: fjerner observer fra listen observers.

public void Notify(List<double> list)

Parametre: List<double>list

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden Update, som kalder op i PL.

## 3.3 Presentation Layer – namespace Projekt.PL

### 3.3.1 Klasse: AlarmInterface

**Ansvar:** AlarmInterface har til ansvar at alarmere operatøren og at spille alarmtone, hvis grænseværdier er overskredet. Funktionen Alarm er ikke implementeret i programmet. Dette diskuteres i rapporten (jf. Rapporten s. 45).

**Metoder:** `public void AlarmInterface_Load(object sender, EventArgs e)`

Parametre: `object sender, EventArgs e`.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden `AfspilAlarmTone()`. Metoden forekommer, når AlarmInterface åbnes.

`public void KnapAlarm_Click(object sender, EventArgs e)`

Parametre: `object sender, EventArgs e`.

Returværdi: ingen returværdi.

Beskrivelse: sætter boolean `KnapAlarmStopKlik` til true, og lukker interfacet. Metoden forekommer, når operatøren trykker på "Stop Alarm"-knappen.

`public void AfspilAlarmTone()`

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: afspiller alarmlyd.

### 3.3.2 Klasse: GemInterface

**Ansvar:** GemInterface har til ansvar, at registrere brugerens indtastede CPR-nummer.

**Metoder:** public void KnapGemOK\_Click(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden CPR\_tjek() på det indtastede CPR. Hvis CPR-nummeret er gyldigt, kaldes metoden GemMåling(), hvorefter GemInterface lukkes. Metoden forekommer, når operatøren trykker på "Gem"-knappen i GemInterface.

### 3.3.3 Klasse: OperatørInterface

**Ansvar:** OperatørInterface har til ansvar, at vise data som graf, og blodtryksværdier. Derudover har den til ansvar at vælge grænseværdier for systolisk og diastolisk blodtryk, samt vælge om signalet skal filtreres eller ej. Interfacet står også for at kalde nulstil-, stop-, start- og gem-metoderne.

**Metoder:** private void Stop()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden StopMåling() på et objekt af typen BL\_ForetagOgVisMåling.

private void TjekBlodtryk()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: registrerer de valgte grænseværdier, og kalder metoderne TjekSystoliskBlodtryk() og TjekDiastoliskBlodtryk(). Hvis en af disse er true, kaldes metoderne Stop(), og BegyndAlarm(). Metoden er ikke vellykket implementeret i programmet.

private void checkBoxFilter\_CheckedChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.



Beskrivelse: kalder metoden `FiltrerMåling()` på et objekt af typen `BL_ForetagOgVisMåling`. Metoden forekommer, når operatør trykker på "Filtreret"-boksen. Denne er ikke implementeret i programmet.

`private void UpdateVærdier(List<double> list)`

Parametre: `List<double> list`.

Returværdi: ingen returværdi.

Beskrivelse: opdaterer og udskriver værdierne for systolisk, diastolisk og middelblodtryk på `OperatørInterface`.

`private void LavGraf(List<double> list)`

Parametre: `List<double> list`.

Returværdi: ingen returværdi.

Beskrivelse: Ligger data i grafen. Vha. case switch statement tjekker metoden, om der er data i grafen, om der skal lægges data oveni grafen eller grafen skal tegnes forfra.

`public void Update(List<double> list)`

Parametre: `List<double> list`.

Returværdi: ingen returværdi.

Beskrivelse: skifter til GUI-tråden og kalder dernæst metoderne `UpdateVærdier()` og `LavGraf()`.

`private void KnapStart_Click(object sender, EventArgs e)`

Parametre: `object sender, EventArgs e`.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden `DisableAlarmVærdier()` og aktiverer `KnapStop`, mens `KnapStart`, `KnapGem` og `KnapNulstil` deaktiveres. Metoden forekommer, når operatøren trykker på "Start"-knappen.

private void KnapStop\_Click(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden EnableAlarmVærdier(), samt enabler KnapGem, KnapNulstil og KnapStart, mens KnapGem disables. Metoden forekommer, når operatøren trykker på "Stop"-knappen.

private void KnapGem\_Click(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden Show() på et objekt af typen GemInterface og denne bliver vist. Metoden forekommer, når operatøren trykker på "Gem"-knappen.

private void KnapNulstil\_Click(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoderne NulstilGraf(), NulstilBlodtryksværdier() og NulstilAlarm() og aktiverer KnapStart. Metoden forekommer, når operatøren trykker på "Nulstil"-knappen.

private void NulstilGraf()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: nulstiller grafvinduet.

private void NulstilBlodtryksværdier()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: nulstiller blodtryksværdierne i de tilhørende tekstbokse.

private void NulstilAlarm()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: sætter blodtryksværdierne i de tilhørende tekstbokse til standardværdierne.

public void BegyndAlarm()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden Show() på et objekt af typen AlarmInterface og viser denne.

private void TextBoxSystoliskØvre\_TextChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: aflæser trackbaren for den systoliske øvre grænseværdi på GUI og sætter tekstboksen for den systoliske øvre grænseværdi til denne værdi.

private void TextBoxSystoliskNedre\_TextChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: aflæser trackbaren for den systoliske nedre grænseværdi på GUI og sætter tekstboksen for den systoliske nedre grænseværdi til denne værdi.

private void TextBoxDiastoliskØvre\_TextChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: aflæser trackbaren for den diastoliske øvre grænseværdi på GUI og sætter tekstboksen for den diastoliske øvre grænseværdi til denne værdi.

private void TextBoxDiastoliskNedre\_TextChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: aflæser trackbaren for den diastoliske nedre grænseværdi på GUI og sætter tekstboksen for den diastoliske nedre grænseværdi til denne værdi.

private int AlarmGrænserDiastolisk(int i)

Parametre: int i.

Returværdi: int i.

Beskrivelse: kalder metoden TrackBargrænser() og lader grænseværdierne for diastolisk blodtryk være mellem 0 og 150.

private int AlarmGrænserSystolisk(int i)

Parametre: int i.

Returværdi: int i.

Beskrivelse: kalder metoden TrackBargrænser() og lader grænseværdierne for systolisk blodtryk være mellem 60 og 300.

private void EnableAlarmVærdier()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: aktiverer alle trackbarer og tekstbokse, der har at gøre med alarmgrænser.

private void DisableAlarmVærdier()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: deaktiverer alle trackbarer og tekstbokse, der har at gøre med alarmgrænser.

private void TrackBargrænser()

Parametre: ingen parametre.

Returværdi: ingen returværdi.

Beskrivelse: sætter trackbarernes maksimum og minimum. Den systoliske øvre grænseværdi går fra 80 – 300, den systoliske nedre grænseværdi får fra 60 – 280, den diastoliske øvre grænseværdi går fra 20 – 151, mens den diastoliske nedre værdi går fra 0 – 130.

private void trackBarDiastoliskØvre\_ValueChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden TrackBargrænser() og laver en minimumsforskel mellem trackbaren for den diastoliske nedre grænseværdi og trackbar for den diastoliske øvre grænseværdi. Metoden forekommer, når operatøren ændrer i trackbaren tilhørende den diastoliske øvre grænseværdi.

private void trackBarDiastoliskNedre\_ValueChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden TrackBargrænser() og laver en minimumsforskel mellem trackbaren for den diastoliske nedre grænseværdi og trackbar for den diastoliske øvre grænseværdi. Metoden forekommer, når operatøren ændrer i trackbaren tilhørende den diastoliske nedre grænseværdi.

private void trackBarSystoliskØvre\_ValueChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden TrackBargrænser() og laver en minimumsforskel mellem trackbaren for den systoliske nedre grænseværdi og trackbar for den systoliske øvre grænseværdi. Metoden forekommer, når operatøren ændrer i trackbaren tilhørende den systoliske øvre grænseværdi.

private void trackBarSystoliskNedre\_ValueChanged(object sender, EventArgs e)

Parametre: object sender, EventArgs e.

Returværdi: ingen returværdi.

Beskrivelse: kalder metoden TrackBargrænser() og laver en minimumsforskel mellem trackbaren for den systoliske nedre grænseværdi og trackbar for den systoliske øvre grænseværdi. Metoden forekommer, når operatøren ændrer i trackbaren tilhørende den systoliske nedre grænseværdi.

```
private void KnapStartNulstilJustering_Click(object sender, EventArgs e)
```

Parametre:   object sender, EventArgs e.

Returværdi:   ingen returværdi.

Beskrivelse:   kalder metoden NulpunktsJuster() og sætter ATMtryk til denne værdi, samt aktiverer KnapÅbenProgram. Metoden forekommer, når operatøren trykker på "Start nulpunktsjustering"-knappen.

```
private void KnapÅbenProgram_Click(object sender, EventArgs e)
```

Parametre:   object sender, EventArgs e.

Returværdi:   ingen returværdi.

Beskrivelse:   kalder metoden GørSynlig(), samt deaktiverer KnapNulstilJustering og KnapÅbenProgram. Derudover gøres de to labels på GUI usynlige. Metoden forekommer, når operatør trykker på "Åben program"-knappen.

```
private void GørSynlig()
```

Parametre:   ingen parametre.

Returværdi:   ingen returværdi.

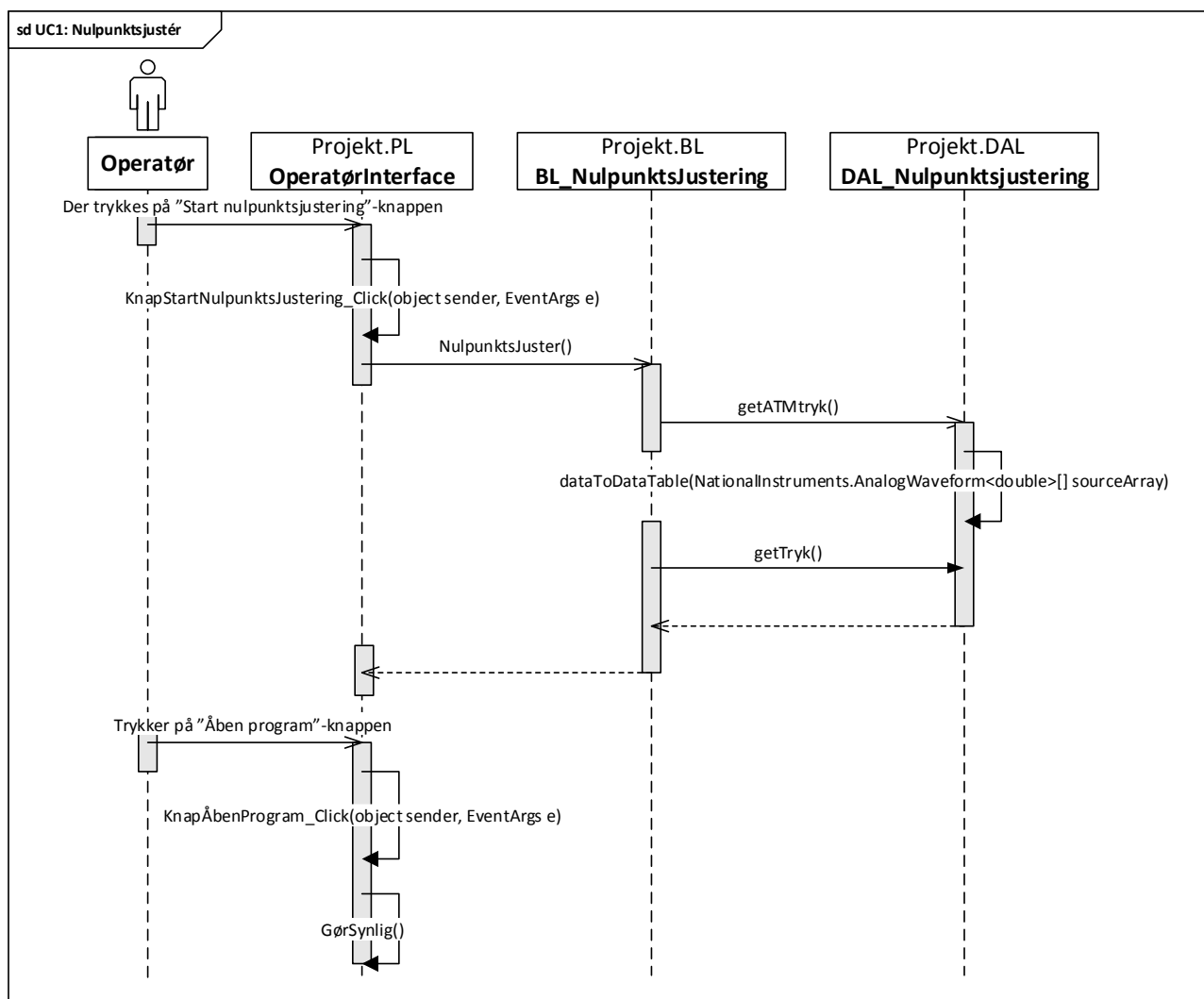
Beskrivelse:   gør alt synligt på GUI af tekstbokse, knapper, labels samt graf.

## 4.0 Sekvensdiagrammer

Sekvensdiagrammerne beskriver en række af metodekald, der forekommer i koden i forskellige situationer. Der er lavet sekvensdiagrammer svarende til systemets Use Cases, undtagen UC4 Filtrer måling og UC5 Alarmer da disse funktioner ikke er vellykket implementeret.

### 4.1 UC1: Nulpunktsjustér

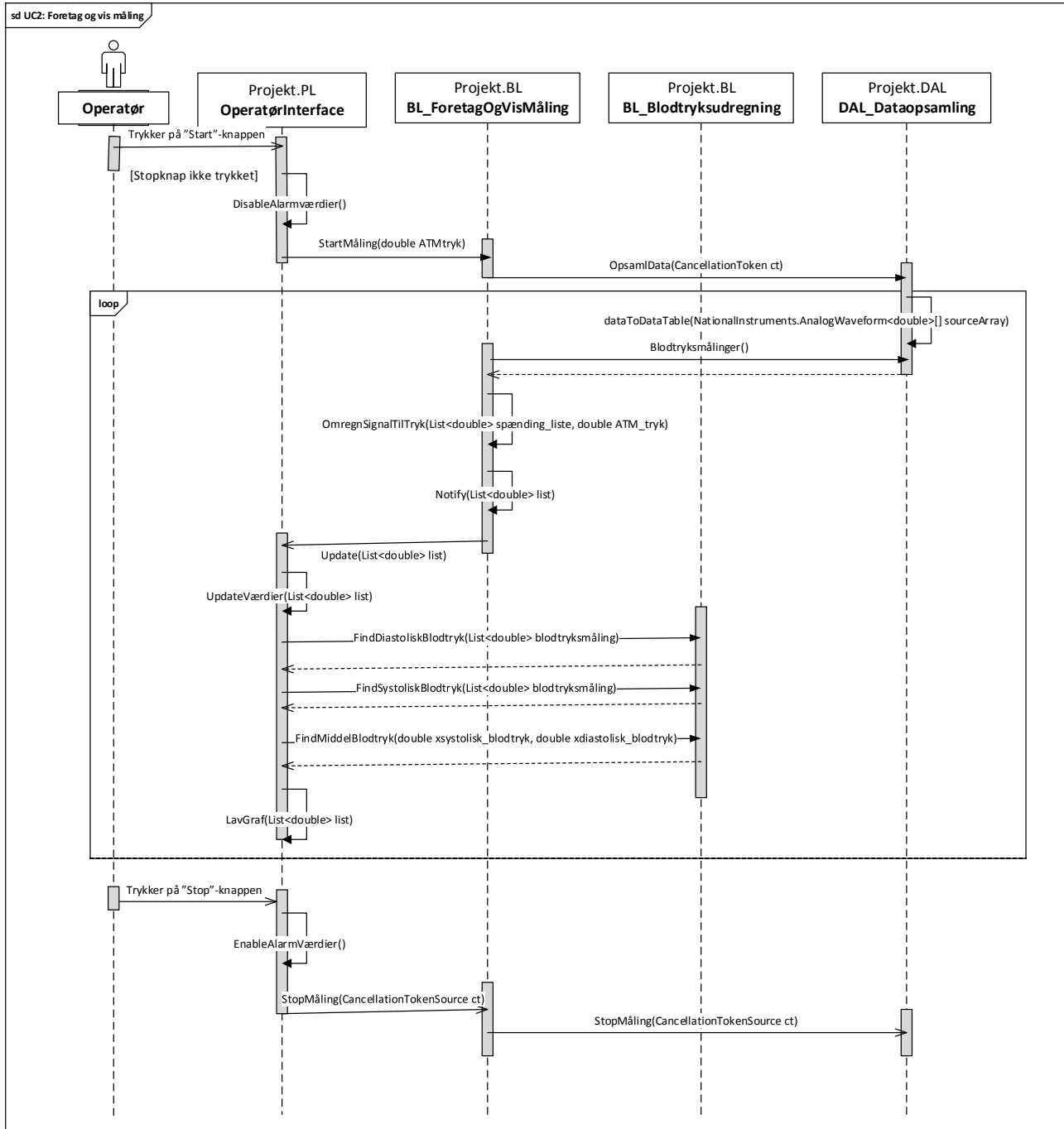
Sekvensdiagrammet på figur 2 beskriver rækken af metodekald, der forekommer, når operatøren trykker på "Start nulpunktsjustering"-knappen. Det knytter sig til UC1 Nulpunktsjustér.



Figur 2 Sekvensdiagram UC1 Nulpunktsjuster

## 4.2 UC2: Foretag og vis måling

Sekvensdiagrammet på figur 3 beskriver rækken af metodekald, der forekommer, når operatøren trykker på "Start"- og "Stop"-knappen. Det knytter sig til UC2 Foretag og vis måling.

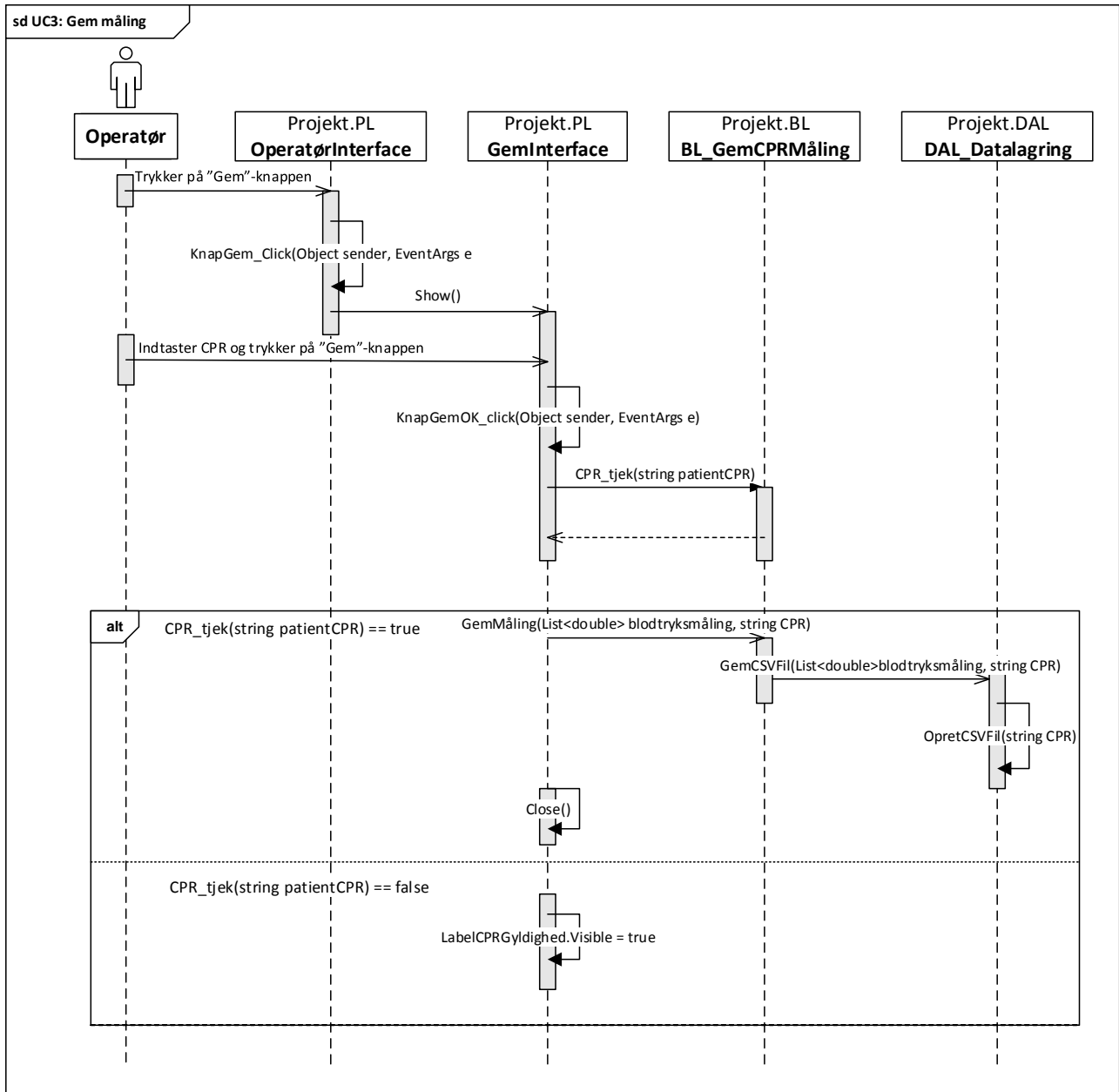


Figur 3 Sekvensdiagram UC2 Foretag og vis måling



### 4.3 UC3: Gem måling

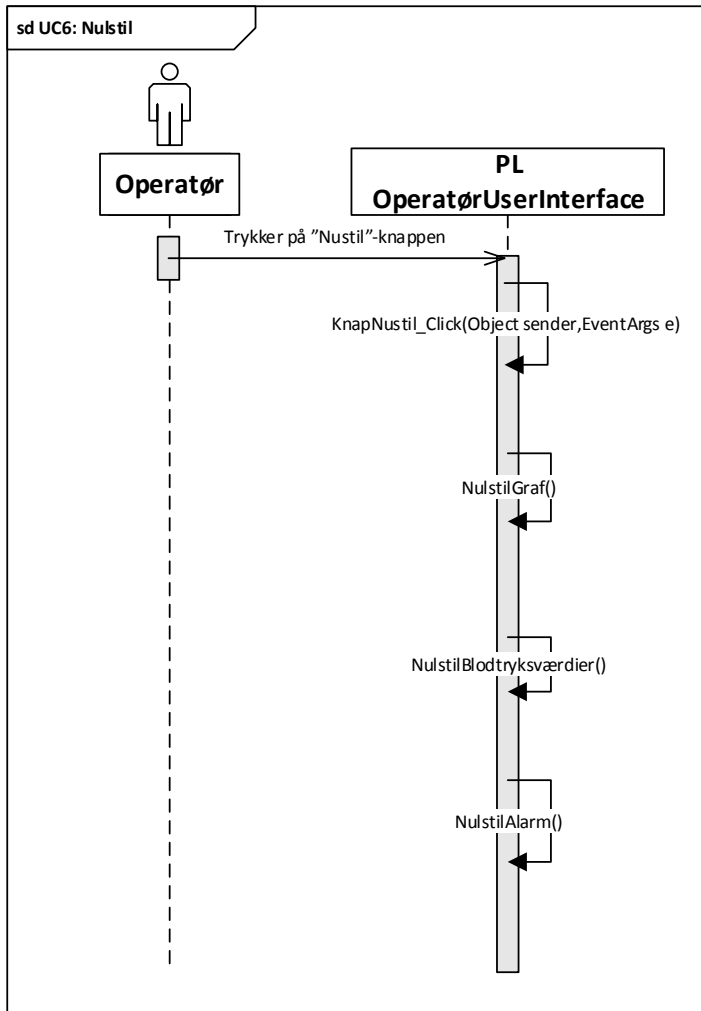
Sekvensdiagrammet på figur 4 beskriver rækken af metodekald, der forekommer, når operatøren trykker på "Gem"-knappen. Det knytter sig til UC3 Gem måling.



Figur 4 Sekvensdiagram UC3 Gem måling

## 4.4 UC6: Nulstil

Sekvensdiagrammet på figur 5 beskriver rækken af metodekald, der forekommer, når operatøren trykker på "Nulstil"-knappen. Det knytter sig til UC6 Nulstil.



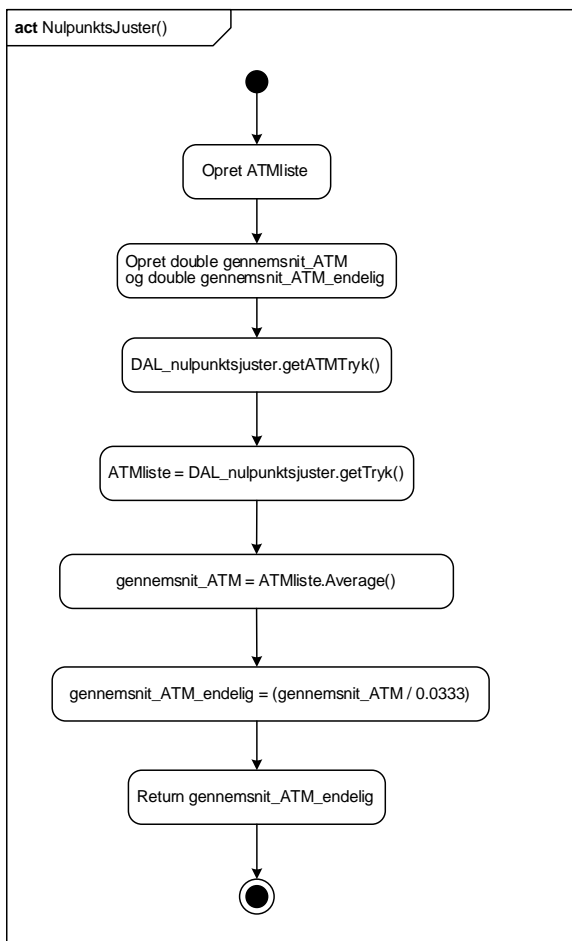
Figur 5 Sekvensdiagram UC6 Nulstil

## 5.0 Aktivitetsdiagrammer

Aktivitetsdiagrammer beskriver aktiviteterne og aktionerne i der forskellige metoder. Der er kun aktivitetsdiagrammer for de metoder, gruppen selv har udarbejdet fra bunden.

### 5.1 Metoden NulpunktsJuster i klassen BL\_Nulpunktsjustering

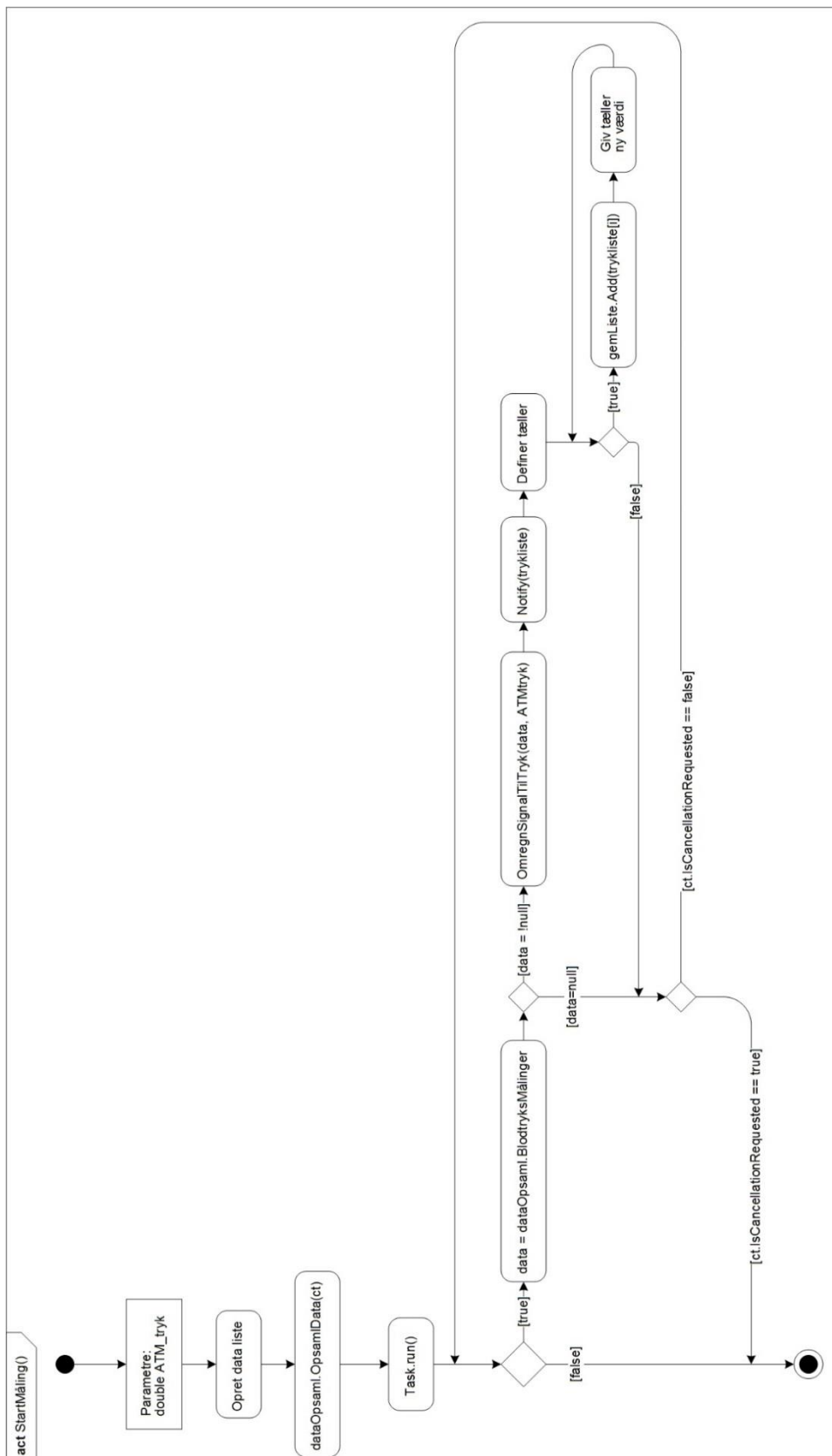
Aktivitetsdiagrammet på figur 6 beskriver metoden Nulpunktsjuster, der optræder i klassen BL\_Nulpunktsjustering.



Figur 6 Aktivitetsdiagram metode NulpunktsJuster()

### 5.2 Metoden StartMåling i klassen BL\_ForetagOgVisMåling

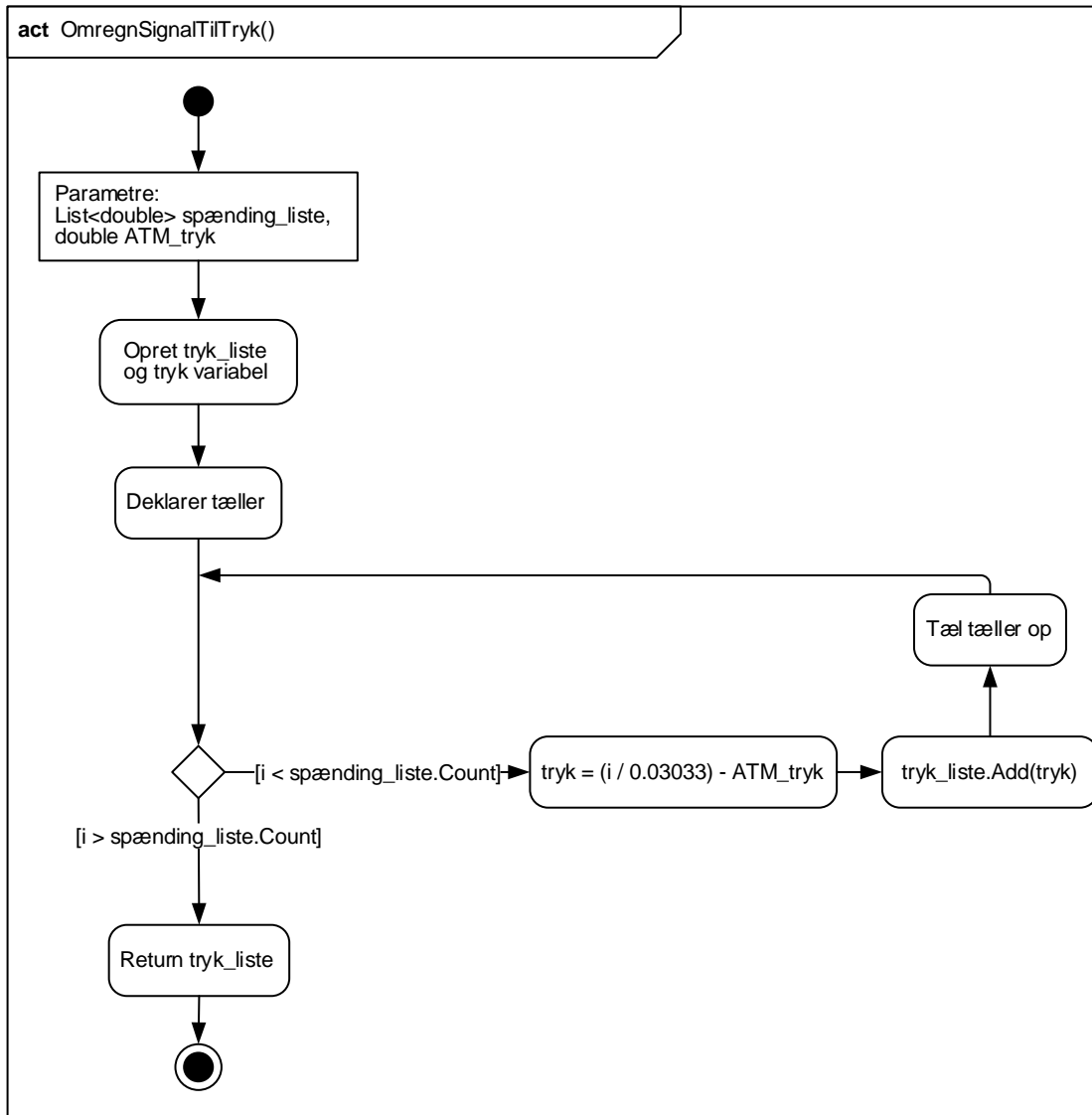
Aktivitetsdiagrammet i figur 7 beskriver metoden StartMåling, der optræder i klassen BL\_ForetagOgVisMåling.



Figur 7 Aktivitetsdiagram metode `StartMåling()`

### 5.3 Metoden OmregnSignalTilTryk i klassen BL\_ForetagOgVisMåling

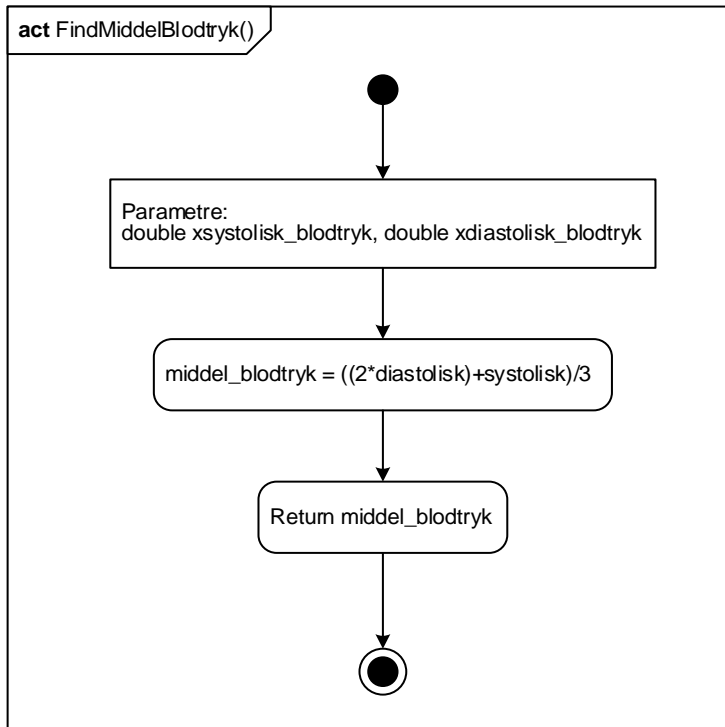
Aktivitetsdiagrammet på figur 8 beskriver metoden OmregnSignalTilTryk, der optræder i klassen BL\_ForetagOgVisMåling.



Figur 8 Aktivitetsdiagram metode OmregnSignalTilTryk()

## 5.4 Metoden FindMiddelBlodtryk i klassen BL\_Blodtryksudregning

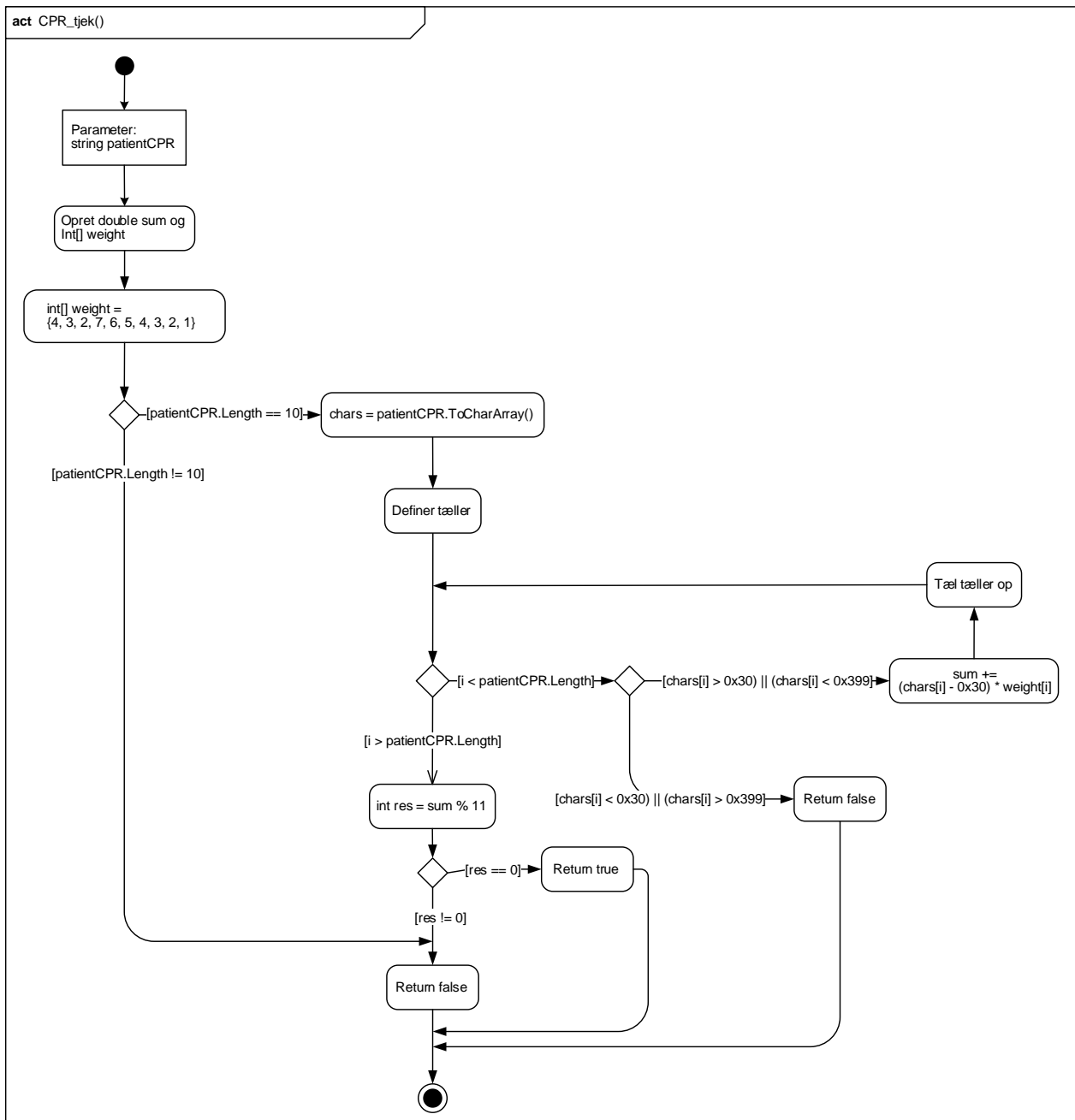
Aktivitetsdiagrammet på figur 9 beskriver metoden FindMiddelBlodtryk, der optræder i klassen BL\_Blodtryksudregning. Der bruges en bestemt formel for middelblodtrykket. (1)



Figur 9 Aktivitetsdiagram metode FindMiddelBlodtryk()

## 5.5 Metoden CPR\_tjek i klassen BL\_GemCPRMåling

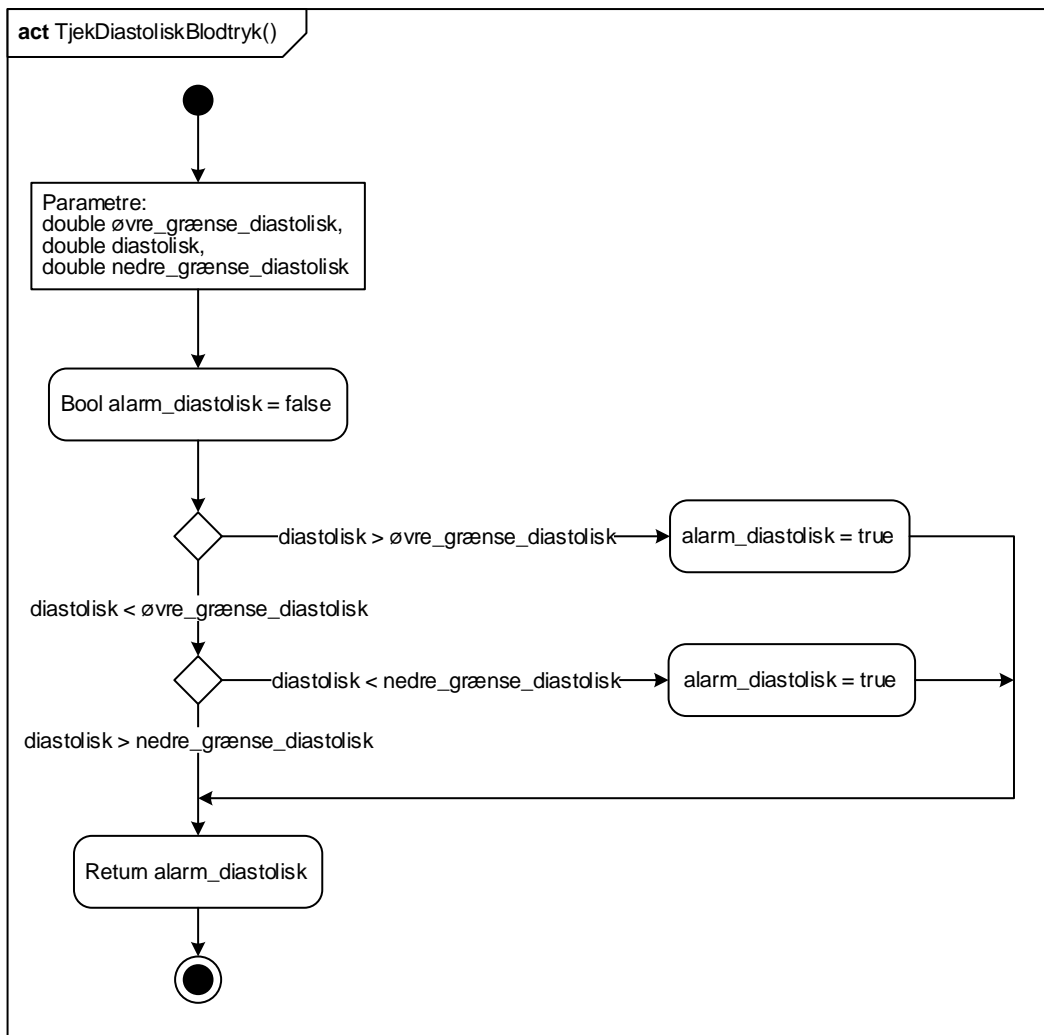
Aktivitetsdiagrammet på figur 10 beskriver metoden CPR\_tjek, der optræder i klassen BL\_GemCPRMåling.



Figur 10 Aktivitetsdiagram metode CPR\_Tjek()

## 5.6 Metoden TjekDiastoliskBlodtryk i klassen BL\_Alarmer

Aktivitetsdiagrammet på figur 11 beskriver metoden TjekDiastoliskBlodtryk, der optræder i klassen BL\_Alarmer. Metoden TjekSystoliskBlodtryk i samme klasse er udarbejdet på samme måde.



Figur 11 Aktivetsdiagram metode TjekDiastoliskBlodtryk()



## 6.0 Enhedstest

Følgende afsnit beskriver de enhedstest, der er foretaget under implementeringen. Koden til de forskellige tests findes under bilag.

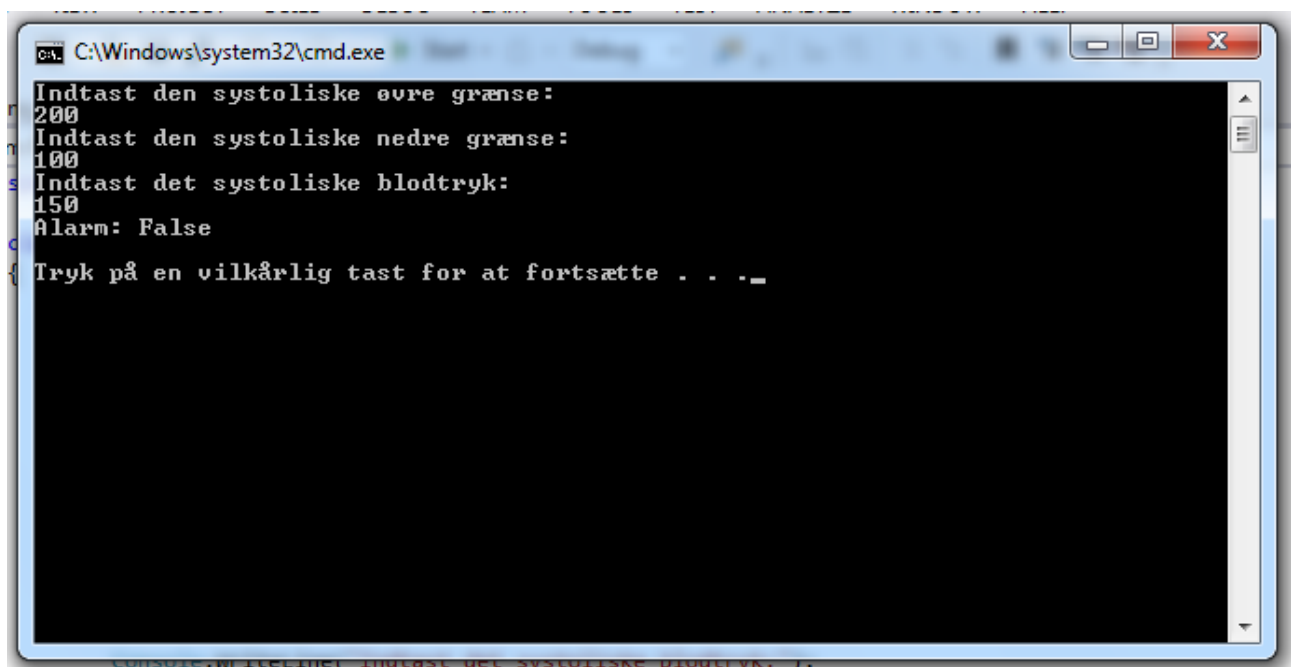
### 6.1 BL\_Alarming

I denne enhedstest testes der på metoden "TjekSystoliskBlodtryk()" i klassen "BL\_Alarming".

Der oprettes et test-program (Console applikation) i Visual Studio med navnet TestAlarm. Her oprettes klassen BL\_Alarming, som den skal implementeres, og klassen kaldes BL\_Alarming\_test.

I program klassen oprettes et objekt af BL\_Alarming\_test klassen, og i console-vinduet bliver man bedt om at indtaste den systoliske øvre grænse, den systoliske nedre grænse og det systoliske blodtryk. Efter dette er indtastet, køres metoden TjekSystoliskBlodtryk() fra BL\_Alarming\_test klassen med de indtastede værdier og den returnerer om alarmen er true (alarmværdierne er overskredet) eller false (alarmværdierne er ikke overskredet). Dette udskrives i console-vinduet.

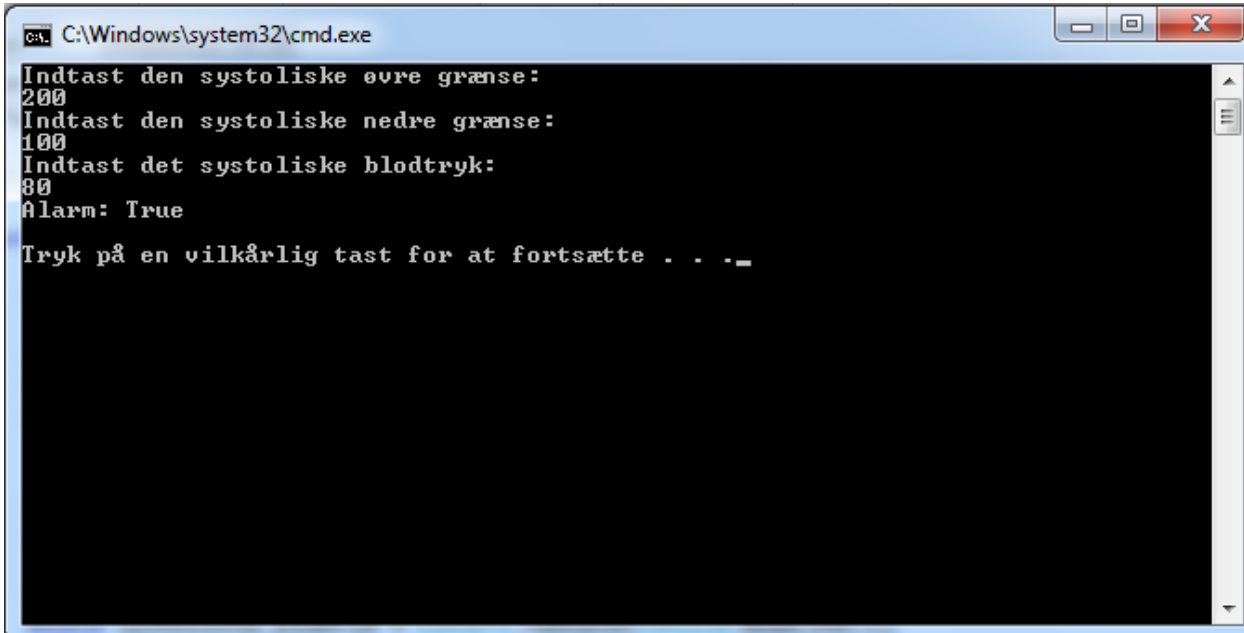
Programmet køres med indtastning af værdierne 200 som øvre grænseværdi og 100 som nedre grænseværdi. Det køres først med en systolisk værdi på 150, som altså ikke overskrider grænseværdierne. Her udskrives det, at alarmen er false. Dette ses på figur 12.



```
C:\Windows\system32\cmd.exe
Indtast den systoliske øvre grænse:
200
Indtast den systoliske nedre grænse:
100
Indtast det systoliske blodtryk:
150
Alarm: False
Tryk på en vilkårlig tast for at fortsætte . . . _
```

Figur 12 Unittest Alarming - grænseværdi ikke overskredet

Dernæst køres programmet med en systolisk værdi på 80, som altså er under den nedre grænseværdi. Her udskrives det, at alarmen er true. Dette ses på figur 13.

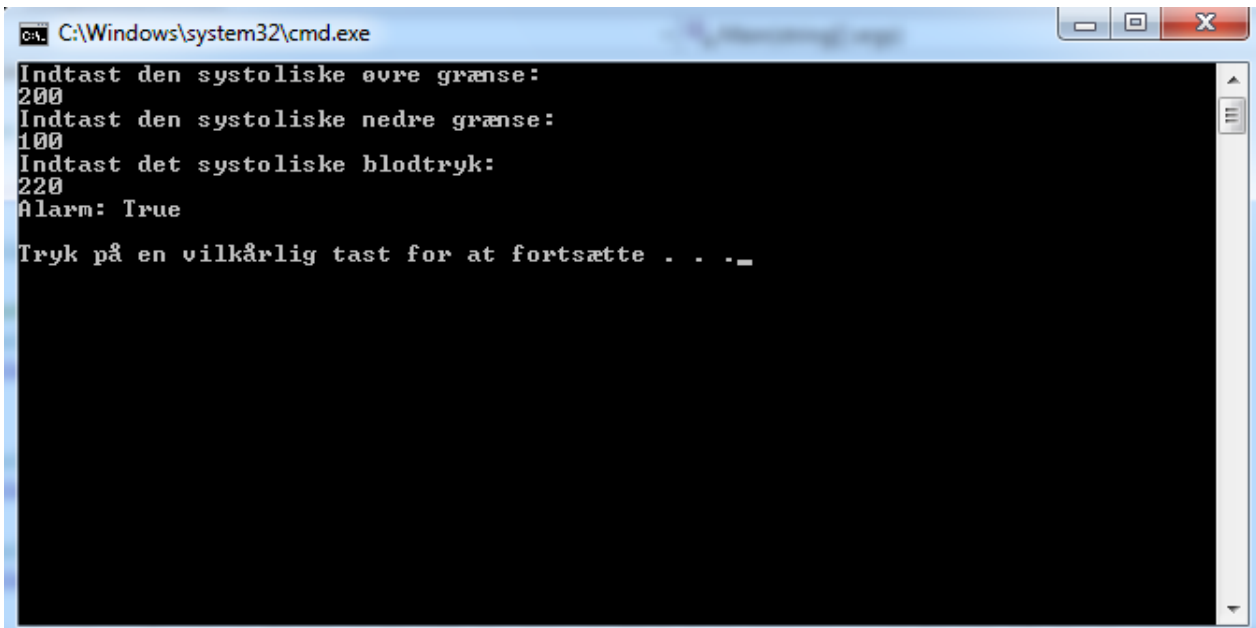


```
C:\Windows\system32\cmd.exe
Indtast den systoliske øvre grænse:
200
Indtast den systoliske nedre grænse:
100
Indtast det systoliske blodtryk:
80
Alarm: True
Tryk på en vilkårlig tast for at fortsætte . . . _
```

Figur 13 Unittest Alarmering – nedre grænseværdi overskredet

Til sidst køres programmet med en systolisk værdi på 220, som altså er over den øvre grænseværdi. Her udskrives det, at alarmen er true. Dette ses på figur 14.

Den tilsvarende metode TjekDiastoliskBlodtryk() er ikke nødvendig at teste, da den har præcis samme opbygning blot med diastoliske værdier i stedet for systoliske.



```
C:\Windows\system32\cmd.exe
Indtast den systoliske øvre grænse:
200
Indtast den systoliske nedre grænse:
100
Indtast det systoliske blodtryk:
220
Alarm: True
Tryk på en vilkårlig tast for at fortsætte . . . _
```

Figur 14 Unittest Alarmering - øvre grænseværdi overskredet

## 6.2 BL\_Blodtryksudregning

I denne enhedstest testes der på metoden "FindMiddelBlodtryk()" i klassen "BL\_Blodtryksudregning".

Der oprettes et test-program (Console applikation) i Visual Studio med navnet Blodtryksudregningstest. Her oprettes klassen BL\_Blodtryksudregning med dens metoder. I program-klassen oprettes en test-liste med værdier svarende til patientens blodtryksværdier. Her kaldes metoderne fra BL\_Blodtryksudregning-klassen. Først udskrives test-listens værdier. Herefter testes metoden FindSystoliskBlodtryk() fra BL\_Blodtryksklassen, som skal finde den maksimale værdi fra listen. Efterfølgende testes metoden FindDiastoliskBlodtryk(), der skal finde den mindste værdi fra listen. Til slut testes metoden FindMiddelBlodtryk(), som skal udregne middelblodtryksværdien, ud fra følgende formel:

$$\text{Middelblodtryk} = \frac{2 * \text{diastolisk blodtryk} + \text{systolisk blodtryk}}{3}$$

Resultatet ses på figur 14.



```

C:\Windows\system32\cmd.exe
Udskriver værdier i test-liste:
2,2
3,2
4,2
5,2
6,2
1,2
systolisk blodtryksværdi: 6,2
diastolisk blodtryksværdi: 1,2
middelblodtryksværdi :2,866666666666667
Press any key to continue . . .
  
```

Figur 15 Unittest Blodtryksudregning

Her ses det, at den maksimale værdi er 6,2, samt den mindste værdi er 1,2. For at eftervise udregningen for middelblodtrykket, gøres følgende:

$$\text{Middelblodtryk} = \frac{2 * 1,2 \text{ mmHg} + 6,2 \text{ mmHg}}{3} = 2,867 \text{ mmHg}$$

Det ses, at metoden FindMiddelBlodtryk()'s resultat stemmer overens med udregningen.

## 6.3 DAL\_Datalagring

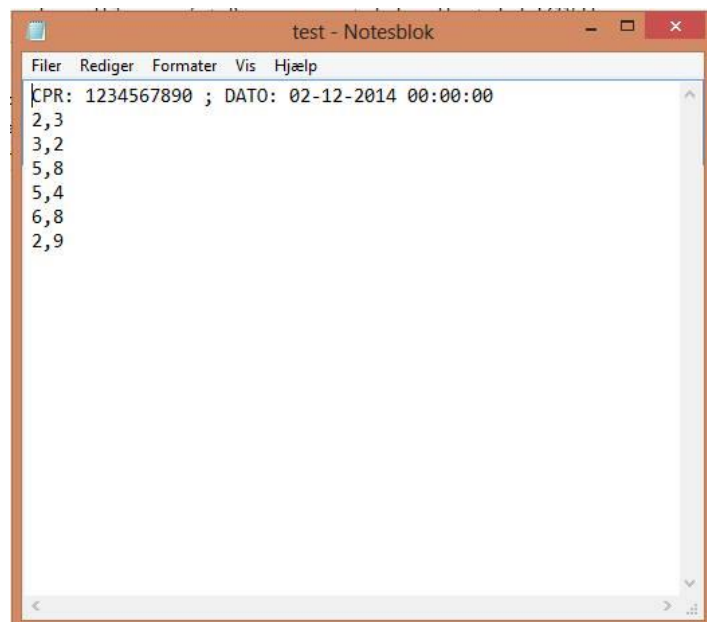
I denne enhedstest testes der på metoderne i klassen "DAL\_Datalagring".

Der oprettes et test-program (Console applikation) i Visual Studio med navnet Test af CSV-fil. Her oprettes klassen DAL\_Datalagring med dens metoder. I program-klassen oprettes en test-liste med værdier af typen *double*: [2.3, 3.2, 5.8, 5.4, 6.8, 2.9]. Her kaldes metoden GemMåling(), hvorefter OpretCSVFil() og GemCSVFil() metoderne bliver kaldt. Der testes OpretCSVFil(), hvorefter en CSV-fil bliver oprettet med navnet "test.csv". Filen er ligeledes oprettet det rigtige sted ift. koden, se figur 16.

Navn	Ændringsdato	Type	Størrelse
Test af CSV-fil	02-12-2014 13:04	Filmappe	
Test af CSV-fil	01-12-2014 13:04	Microsoft Visual S...	1 KB
test	02-12-2014 13:04	CSV-fil	1 KB

Figur 16 Unittest GemMåling - oprettelse af CSV-fil

Næst testes metoden GemCSVFil(). Her bliver test-listens værdier gemt til CSV-filen, samt CPR og dato. Nu åbnes CSV-filen, og det ses, at værdierne er gemt, se figur 17.



Figur 17 Unittest GemMåling - Gem til CSV-fil

Testen er lavet ud fra Projekt v.2.3, der er derfor nogle afvigelser i navngivningen fra det endelige program.

## 6.4 BL\_NulpunktsJustering

I denne enhedstest testes der på metoden "NulpunktsJuster()" i klassen "BL\_NulpunktsJustering".

Der oprettes et nyt program (Console applikation) med navnet "Test af NulpunktsJustering". Heri indsættes koden fra klassen BL\_NulpunktsJustering, og tilpasses, så der kan oprettes en liste med kendte værdier, og teste om metoden gør, som det forventes. Metoden "NulpunktsJuster" ændres til at tage en parametre af typen List<double>. Det er denne liste, som oprettes i program-klassen, der har de kendte værdier.

Fra program-klassen instantieres et objekt af klassen "NulpunktsJustering", og oprettes listen med følgende værdier: [0,2,3,5,6,8,9,7,2,5,9]

Metoden har to funktioner, udover at hente data fra DAL; at finde gennemsnit i listen og omregne til tryk, vha. omregningsfaktoren. Dette testes ved at debugge efter gennemsnitsværdien, og udskrive den omregnede værdi i konsollen. Der regnes efter, om gennemsnitsværdien og den omregnede værdi passer. Gennemsnit af listen:

$$\frac{0 + 2 + 3 + 5 + 6 + 8 + 9 + 7 + 2 + 5 + 9}{11} = 5,09$$

Omregning:

$$\frac{5,09}{0,0301} = 169,133$$

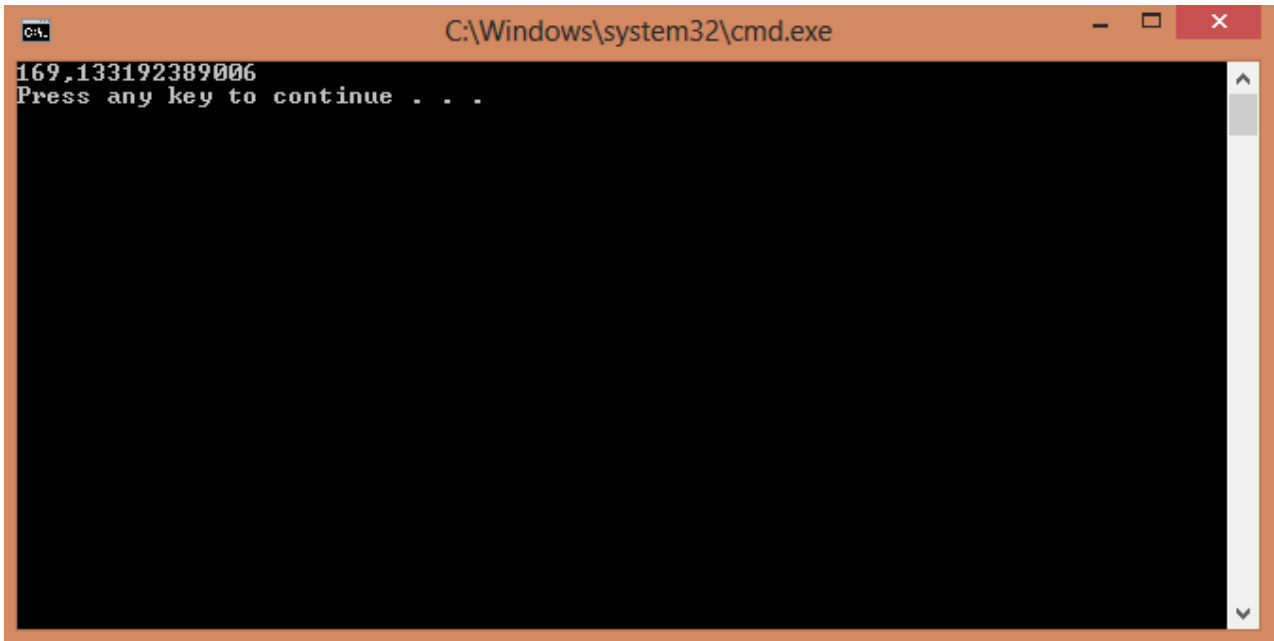
Resultatet af testen ses på nedenstående figurer.

The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays the implementation of the `NulpunktsJuster` method. The method takes a `List<double>` parameter named `liste`. It creates a new `ATMliste`, calculates the average of the list using `ATMliste.Average()`, and then performs a conversion using a constant `5.0909090909090908` (commented as `//omregner til mmHg`). The final result is returned as `gennemsnit_ATM_endelig`.
- Autos Window:** Shows the state of variables during debugging. Key values include:
  - `System.Linq.Enumerable`: 5.0909090909090908 (double)
  - `ATMliste`: Count = 11 (System.Collections.Generic.List<double>)
  - `gennemsnit_ATM`: 5.0909090909090908 (double)
  - `gennemsnit_ATM_endelig`: 0.0 (double)
  - `this`: {Test\_af\_NulpunktsJustering.NulpunktsJuster} (Test\_af\_NulpunktsJustering.NulpunktsJuster)
- Output Window:** Shows the output from the debug session, indicating that the thread has exited with code 0 (0x0).

Figur 18 Unittest NulpunktsJuster - debugging og gennemsnitsværdi

Det ses, at ved debug findes gennemsnitsværdien 5,09 som stemmer overens med den forventede værdi.



```
C:\Windows\system32\cmd.exe
169,133192389006
Press any key to continue . . .
```

Figur 19 Unittest NulpunktsJustering - udskrevet og omregnet værdi

Den udskrevet og omregnede værdi er på 169,133, og dette var også som forventet.

Altså er metoden "NulpunktsJuster" eftervist, og stemmer overens med det ønskede.

Denne tryk værdi bruges så senere i programmet til at trække fra datamålingerne, der hentes fra A/D-converteren.

## 6.5 BL\_GemCPRMåling

I denne enhedstest testes der på metoden "CPRgyldigtjek()" i klassen "BL\_GemCPRMåling".

Der oprettes et test-program (Console applikation) i Visual Studio med navnet CPRvalidering. Her oprettes klassen BL\_GemCPRMåling, som den skal implementeres og kaldes BL\_GemCPRMåling.test.

I program klassen oprettes et objekt af BL\_GemCPRMåling.test –klassen. Herefter oprettes et objekt der tjekker om den gemte CPRnummer "070292\*\*\*\*" er gyldigt. Koden fra projektets CPRgyldigtjek er kopieret ind i klassen "GemCPRMåling". I denne metode bliver der oprettet en variabel, sum, der bliver sat til at være lig med 0. Derefter bliver der oprettet et array, weight. Hvis længden af patientCPR er 10, vil der blive oprettet et char-array, hvor karakterne fra CPR bliver lagt ind i. Herefter køres der en for-løkke, der gennemløber patientCPR. Hvis værdien i char arrayet er mindre end 0 eller større end 9, vil der blive returneret false. Og ellers vil summen blive lagt oveni. Der bliver oprettet en variable, res, der er det

resterende efter sum er blevet divideret med 11. Hvis res = 0 bliver der returneret true. Hvis længden af patient CPR ikke er 10, vil der blive returneret false.

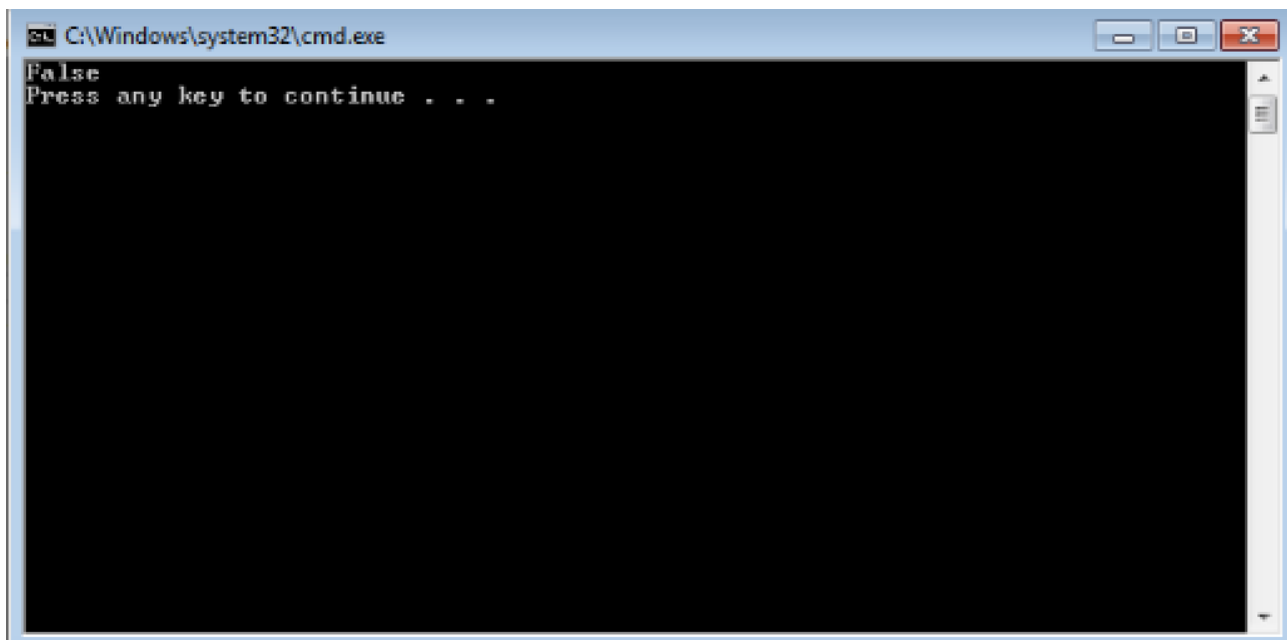
Dette kan demonstreres ved hjælp af dette konsol test program.

Forneden ses, at der er indtastet et CPR-nummer der har mere end 10 karakter.

```
namespace CPRVALIDERING
{
    class Program
    {
        static void Main(string[] args)
        {
            GemCPRMåling test = new GemCPRMåling();
            bool test2 = test.CPR_tjek("07029230333");
            Console.WriteLine(Convert.ToString(test2));
        }
    }
}
```

Figur 20 Unittest GemCPRMåling - forkert CPR nummer

Resultatet ses nedenfor, hvor der bliver returneret false. CPR -nummeret er altså ugyldigt.



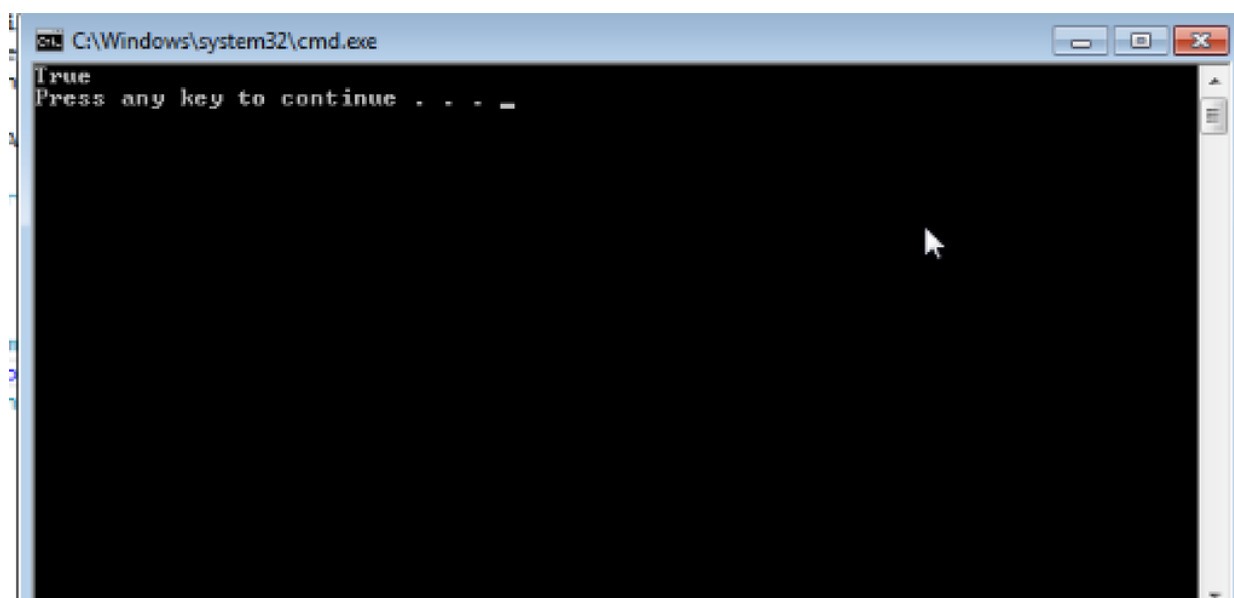
Figur 21 Unittest GemCPRMåling - forkert CPR-nummer

På nedenstående ses at det indtastede CPR-nummer indeholder 10 karakterer.

Resultatet er følgende, hvor der bliver indtastet et korrekt CPR-nummer, og der returneres true.

```
namespace CPRVALIDERING
{
    class Program
    {
        static void Main(string[] args)
        {
            GemCPRMåling test = new GemCPRMåling();
            bool test2 = test.CPR_tjek("0702923085");
            Console.WriteLine(Convert.ToString(test2));
        }
    }
}
```

Figur 22 Unittest GemCPRMåling - rigtigt CPR nummer



Figur 23 GemCPRMåling - rigtigt CPR nummer

Testen er lavet ud fra Projekt v.2.3, der er derfor nogle afvigelser i navngivningen fra det endelige program.

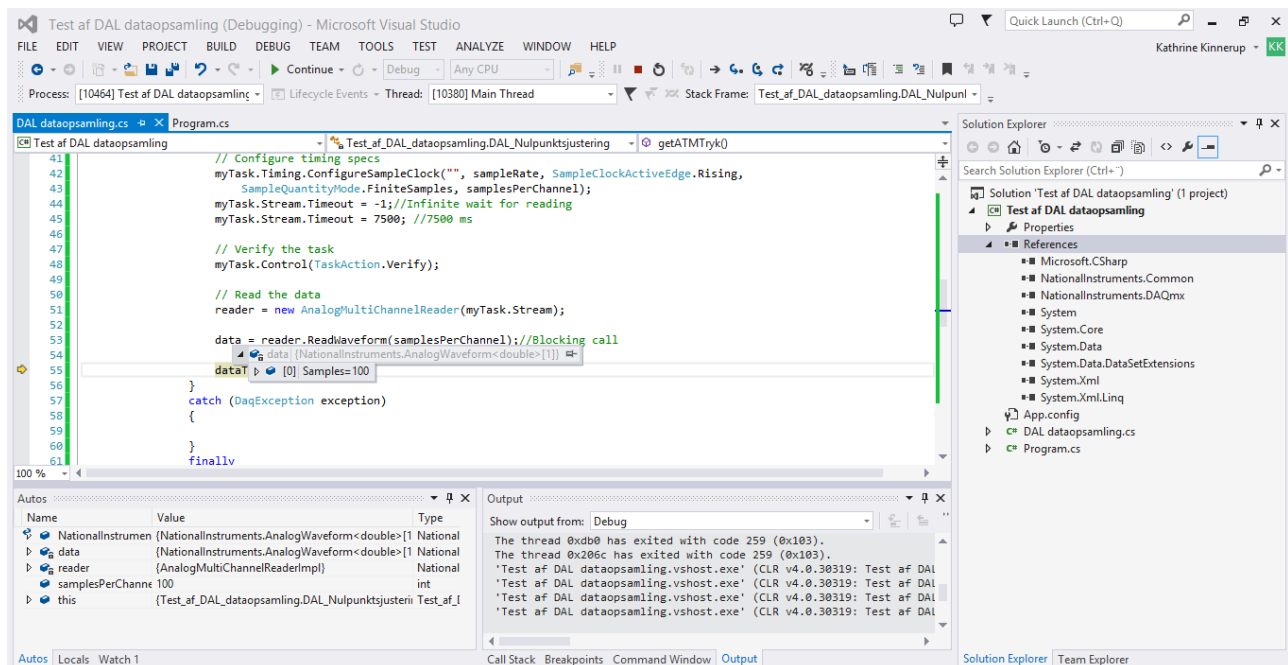


## 6.6 DAL\_Nulpunktsjustering

I denne enhedstest testes der på metoderne i klassen "DAL\_Nulpunktsjustering".

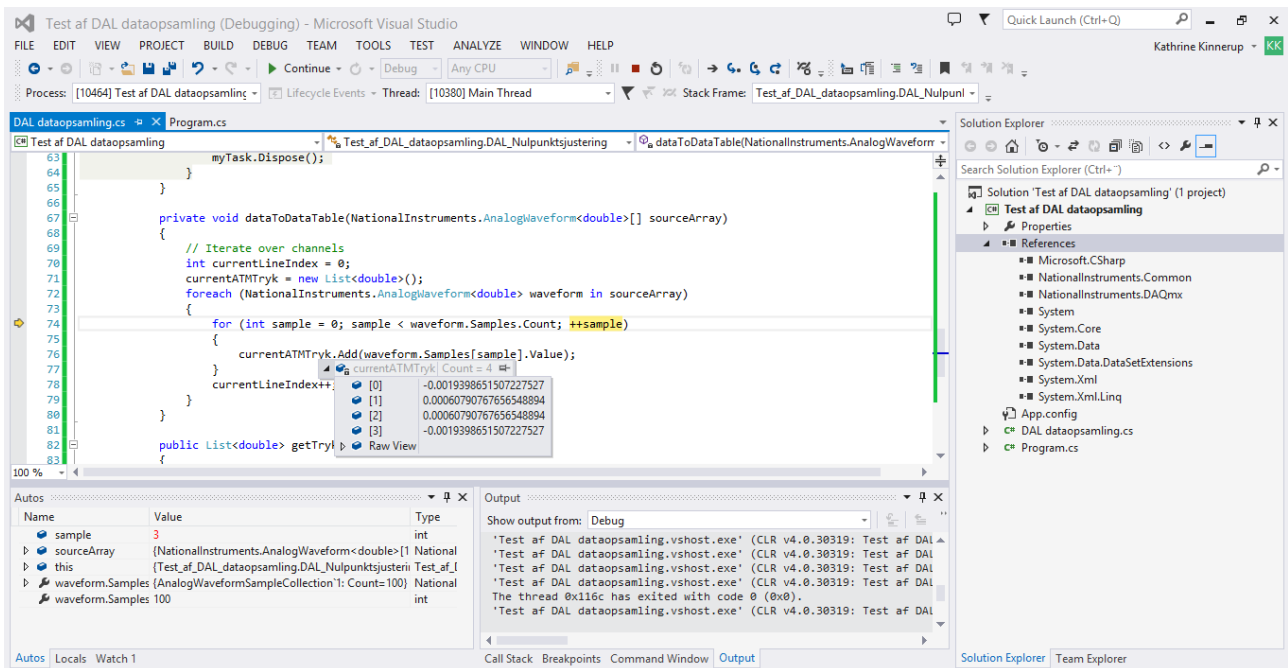
Der oprettes et test-program (Console applikation) i Visual Studio med navnet "Test Af DAL\_Nulpunktsjustering". Her oprettes klassen DAL\_Nulpunktsjuster, hvor programmets metoder bliver kopieret ind i. Herefter bliver et testobjekt oprettet i program klassen af typen DAL\_Nulpunktsjuster, hvor på metoden fra DAL\_Nulpunktsjuster bliver kaldt. Der anvendes nu debugging til at teste, om metoderne virker.

Det ses på figur 24, at data-array'et bliver fyldt med 100 værdier, hvilket svarer til, at der er samlet 100 værdier fra DAQ'en. Der er altså oprettet forbindelse til DAQ'en og indsamlet data herfra.



Figur 24 Nulpunktsjuster - dataarray

Herefter bliver array'ets værdier tilføjet til tryklisten, kaldet currentATMTryk.



Figur 25 Nulpunktsjuster - fra array til liste

Det er denne liste, der bliver brugt i BL\_NulpunktsJustering.

## 6.7 BL\_ForetagOgVisMåling

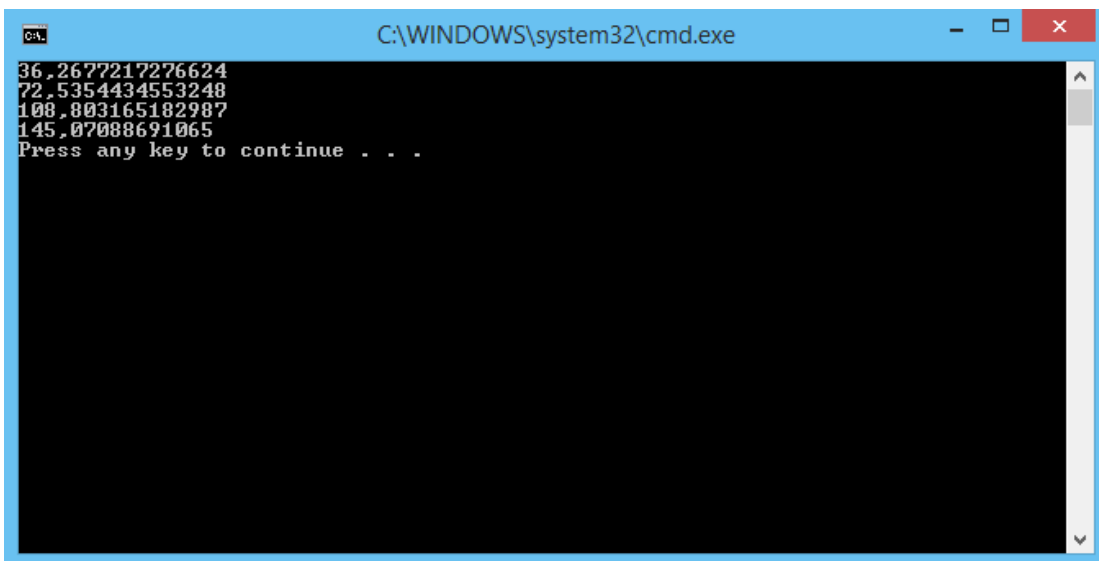
I denne enhedstest testes der på metoden "OmregnSignalTilTryk()" i klassen "BL\_ForetagOgVisMåling".

Der oprettes et test-program (Console applikation) i Visual Studio med navnet Testprogram\_Omregn\_signal\_til\_tryk".

Klassen Omregn\_signal\_til\_tryk oprettes, hvori metoden "OmregnSignalTilTryk()" fra klassen BL\_ForetagOgVisMåling tilpasses. Metoden "OmregnSignalTilTryk()" ændres til at tage én parameter af typen List<double>. I program-klassen oprettes en test-liste med værdier tilsvarende et reelt input til systemet. Værdier er som følger [1.1,2.2,3.3,4.4]. Der instantieres ydermere et objekt af klassen "Omregn\_signal\_til\_tryk".

$$tryk = \frac{spænding}{0,03033}$$

Metoden "OmregnSignalTilTryk()" kaldes på test-listen, og denne udskrives. Resultat vises i et console-vindue.



```

C:\WINDOWS\system32\cmd.exe
36.2677217276624
72.5354434553248
108.803165182987
145.07088691065
Press any key to continue . . .
  
```

Her ses det, at blodtrykket for en spænding på 1,1 er lige med 36,27 mmHg. For at eftervise udregningen for omregningen, gøres følgende:

$$tryk = \frac{1,1}{0.03033} = 36,267721727 \text{ mmHg}$$

Det ses, at metoden "OmregnSignalTilTryk()"s resultat stemmer overens med ovenstående udregning.

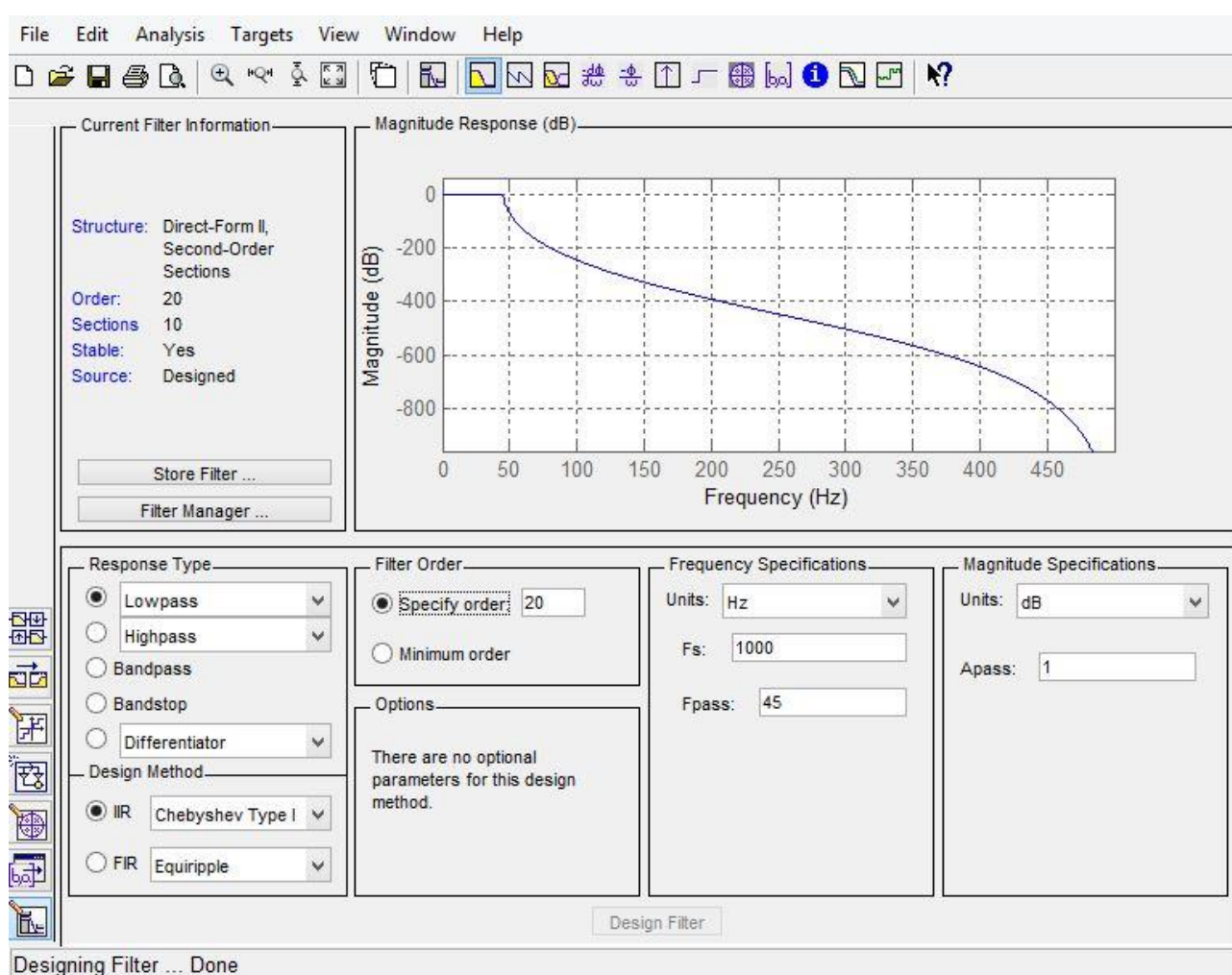
## 6.8 Digitalt filter

Denne test laves for at vise effekten af et digitalt filter. Dette vises vha. et bodeplot og frekvenskarakteristik for et givent signal før og efter filtrering.

Der designes et filter med MatLab's filter designer. Filtret gruppen har tiltænkt at benytte sig af et 20. ordens Chebyshev type 1 lavpasfilter med cut-off frekvens på 45 Hz. Dette bruges for at fjerne så meget 50 Hz støj som muligt.

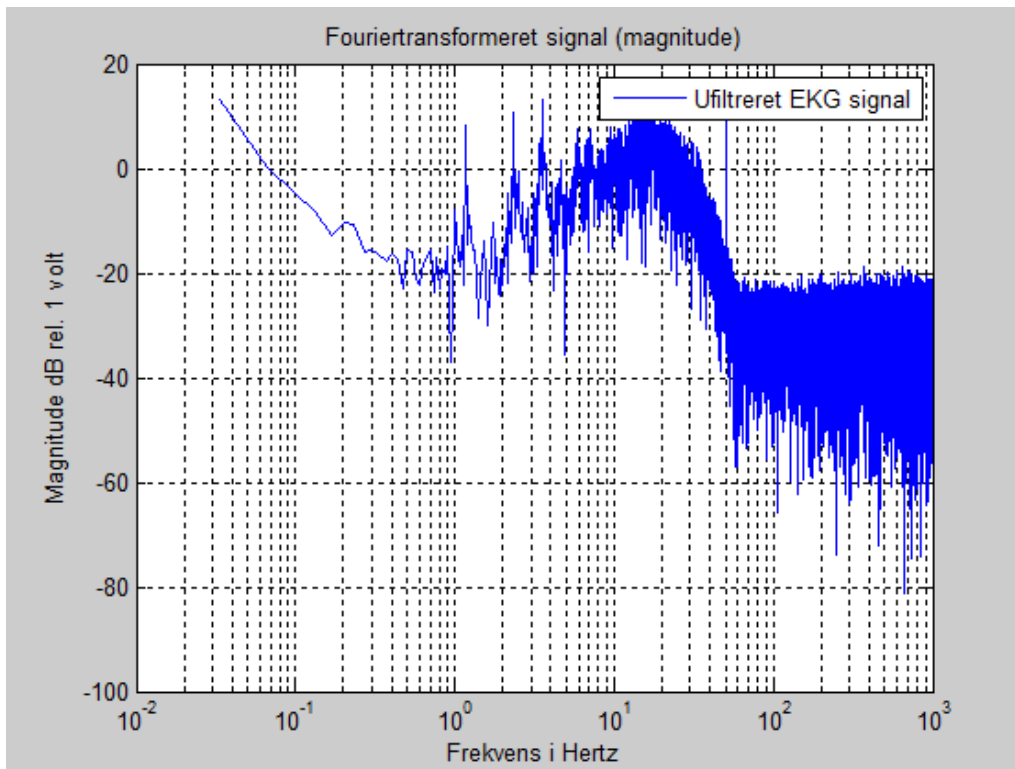
Der indlæses et EKG-signal med tydelig 50 Hz støj.

Filtret designes som vist på billedet nedenfor.



Figur 26 Digitalt filter- MatLab filter designer

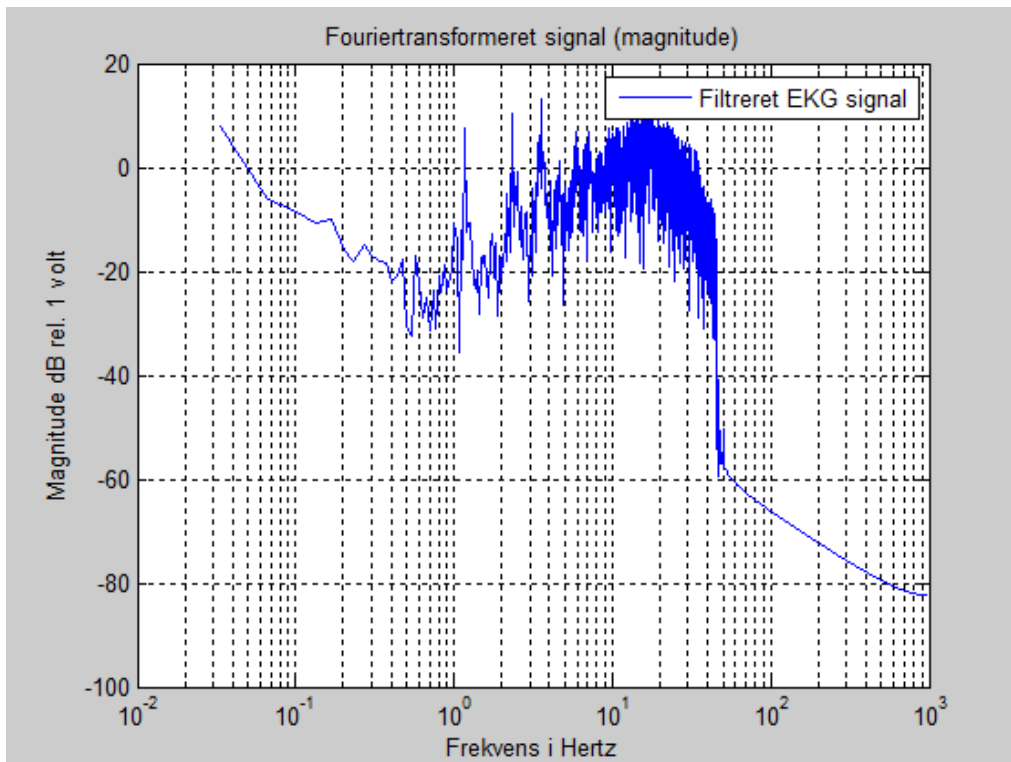
Nedenfor vises frekvenskarakteristikken for det anvendte signal før filtrering.



Figur 27 Frekvenskarakteristik før filtrering

Det ses af figur 27 at signalet indeholder et stort peak ved 50 Hz samt meget støj efter de 50 Hz. Dette ønskes filtreret væk, det før omtalte filter påtrykkes signalet ovenfor vha. MatLab.

Vi ser på frekvenskarakteristikken for det filtrerede signal.



Figur 28 Frekvenskarakteristik efter filtrering

Som det ses af figur 28, er al 50 Hz støj fjernet fra signalet og filteret dæmper alt efter cut-off frekvensen. Denne egenskab er præcis det der søges, og vi ved nu at dette filter burde opfylde gruppens behov i praksis.

## 6.0 Referenceliste

(1) McAuley D. Mean arterial pressure (MAP) Calculator. 2012; Available at:  
<http://www.globalrph.com/map.htm>. Accessed 16/12, 2014.

