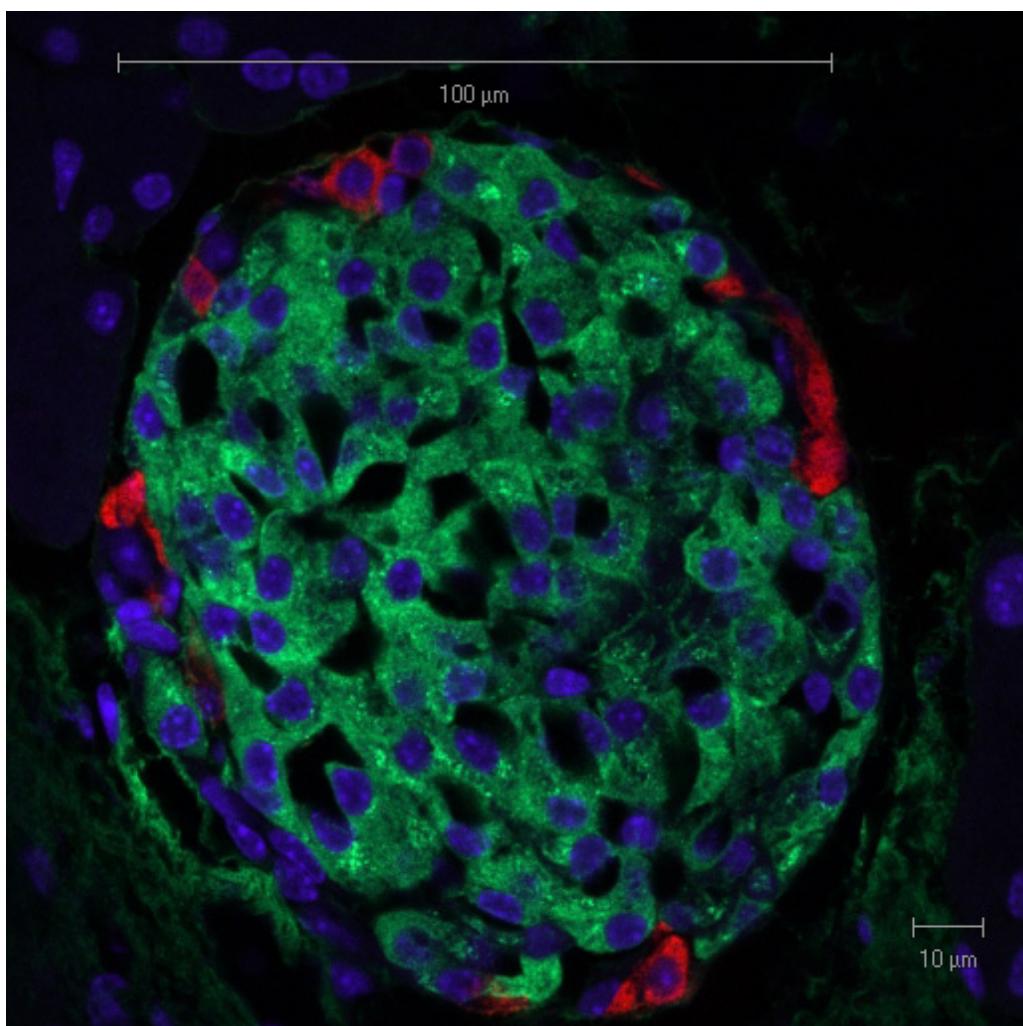


# *Langerhanske Øer Projektdokumentation*

---



BACHELORPROJEKT  
PROJEKTNR: 15137  
INGENIØRHØJSKOLEN, AARHUS UNIVERSITET  
DEN 16. DECEMBER 2015

11424 ANDERS TOFT ANDERSEN  
201270874 ANDERS ESAGER  
PROJEKTVEJLEDER: SAMUEL ALBERG THRYSSØE



# Forord

---

Dette dokument indeholder projektdokumentationen for projektet *Cell sorter for isolation of insulin producing cells*. Dokumentet indeholder kravspecifikation og accepttest for systemet, samt dokumentation af projektets design og implementeringsfase.

Kravspecifikationen er udarbejdet i samarbejde med Søren Gregersen, overlæge på Medicinsk Endokrinologisk Afdeling, Aarhus Universitetshospital, der agerer som projektets kunde.

## Læsevejledning

Alle underdokumenter i denne rapport indeholder en indledning, hvor det enkelte dokuments formål er beskrevet. Hvert dokument indeholder en separat læsevejledning.

Alle dokumenterne og referencer er vedlagt på den afleverede USB.

---

Anders Toft Andersen

---

Anders Esager

## Ordliste

Forkortelser	Betydninger
BDD:	<b>B</b> lock <b>D</b> efinition <b>D</b> igram
IBD:	<b>I</b> nternal <b>B</b> lock <b>D</b> igram
SYML:	<b>S</b> YStems <b>M</b> odeling <b>L</b> anguage
PWM:	<b>P</b> ulse <b>W</b> idth <b>M</b> odulation
ADC:	<b>A</b> nalog <b>D</b> igital <b>C</b> onverter
KVL:	<b>K</b> irckhoff <b>V</b> oltage <b>L</b> aw
KCL:	<b>K</b> irckhoff <b>C</b> urrent <b>L</b> aw
CMRR:	<b>C</b> ommon <b>M</b> ode <b>R</b> ejection <b>R</b> atio
GUI:	<b>G</b> raphical <b>U</b> ser <b>I</b> nterface
ID:	<b>I</b> nner <b>D</b> iameter
OD:	<b>O</b> uter <b>D</b> iameter

# Indholdsfortegnelse

---

<b>Kapitel 1 Kravspecifikation</b>	<b>1</b>
1.1 Indledning . . . . .	1
1.1.1 Formål . . . . .	1
1.1.2 Læsevejledning . . . . .	1
1.1.3 Versionshistorik . . . . .	1
1.2 Systembeskrivelse . . . . .	2
1.2.1 Aktør beskrivelse . . . . .	2
1.3 Funktionelle krav . . . . .	3
1.3.1 Use case diagram . . . . .	3
1.3.2 Use case 1 - Start sorteringscyklus . . . . .	4
1.3.3 Use case 2 - Sortering af Langerhanske Øer . . . . .	5
1.3.4 Use case 3 - Stop sorteringscyklus . . . . .	6
1.3.5 Use case 4 - Indstillinger . . . . .	7
1.3.6 Use case 5 - Data logning . . . . .	8
1.4 Ikke funktionelle krav . . . . .	9
1.4.1 Kvalitetskrav . . . . .	9
1.4.2 Hardware . . . . .	10
1.4.2.1 Microcontroller . . . . .	10
1.4.2.2 Pumpe . . . . .	10
1.4.2.3 Slanger . . . . .	10
1.4.2.4 Beholdere . . . . .	10
1.4.2.5 Ventil . . . . .	10
1.4.2.6 Kamera . . . . .	10
1.4.3 Software . . . . .	10
1.4.3.1 Dataformat og struktur . . . . .	10
1.4.4 GUI - Mockup . . . . .	11
1.5 Projektafgrænsning . . . . .	12
1.6 Samarbejdspartnere . . . . .	12
<b>Kapitel 2 Accepttest</b>	<b>13</b>
2.1 Indledning . . . . .	13
2.1.1 Formål . . . . .	13
2.1.2 Læsevejledning . . . . .	13
2.1.3 Versionshistorik . . . . .	13
2.2 Accepttest af funktionelle krav . . . . .	14
2.2.1 Use case 1: Start sorteringscyklus . . . . .	14
2.2.2 Use case 2: Sortering af langerhanske øer . . . . .	19
2.2.3 Use case 3: Stop sorteringscyklus . . . . .	20
2.2.4 Use case 4: Indstillinger . . . . .	23
2.2.5 Use case 5: Data logning . . . . .	25

2.3 Acceptttest af ikke funktionelle krav . . . . .	26
<b>Kapitel 3 Design</b>	<b>31</b>
3.1 Indledning . . . . .	31
3.1.1 Formål . . . . .	31
3.1.2 Læsevejledning . . . . .	31
3.1.3 Versionshistorik . . . . .	31
3.2 Udviklingsværktøjer . . . . .	32
3.2.1 MATLAB . . . . .	32
3.2.2 GitHub . . . . .	33
3.2.3 Pivotal Tracker . . . . .	34
3.3 Hardware . . . . .	35
3.3.1 Læsevejledning til hardware . . . . .	35
3.3.2 Leverandør af produkter . . . . .	35
3.3.3 Block Definition Diagram . . . . .	36
3.3.4 Internal block Diagram . . . . .	37
3.3.5 Kamera . . . . .	40
3.3.6 Styreenheden . . . . .	42
3.3.7 Motordriver . . . . .	42
3.3.8 Ventil . . . . .	43
3.3.9 Pumpe . . . . .	47
3.3.9.1 Motordriver . . . . .	47
3.3.9.2 Hastighedscontrol af pumpe . . . . .	48
3.3.9.2.1 Pulse width modulation . . . . .	50
3.3.10 Vægtcelle . . . . .	52
3.3.10.1 Forstærkning af signal . . . . .	54
3.3.11 Kamerallys . . . . .	58
3.3.12 Beholdere . . . . .	61
3.3.13 Slanger . . . . .	61
3.3.14 Glasrør . . . . .	62
3.4 Software . . . . .	63
3.4.1 Arduino . . . . .	63
3.4.1.1 LoadCell . . . . .	64
3.4.1.2 Pump Control . . . . .	64
3.4.1.3 Valve Control . . . . .	64
3.4.2 Kamera . . . . .	65
3.4.2.1 CameraFeed . . . . .	65
3.4.2.2 DetectIslets . . . . .	65
3.4.3 Funktioner . . . . .	66
3.4.4 User Interface . . . . .	67
3.4.4.1 Hovedvindue . . . . .	67
3.4.4.2 Indstillinger . . . . .	68
3.4.4.3 Callbacks . . . . .	69
3.4.4.4 Start Callback . . . . .	69
3.4.4.5 Stop Callback . . . . .	70
3.4.4.6 Indstillinger Callback . . . . .	71

---

3.5	Sekvensdiagrammer . . . . .	72
3.5.1	Sekvensdiagram for usecase 1 . . . . .	73
3.5.2	Sekvensdiagram for usecase 2 . . . . .	74
3.5.3	Sekvensdiagram for usecase 3 . . . . .	74
3.5.4	Sekvensdiagram for usecase 4 . . . . .	75
3.5.5	Sekvensdiagram for usecase 5 . . . . .	75
<b>Kapitel 4 Implementering</b>		<b>77</b>
4.1	Indledning . . . . .	77
4.1.1	Formål . . . . .	77
4.1.2	Læsevejledning . . . . .	77
4.1.3	Versionshistorik . . . . .	77
4.2	Hardware . . . . .	78
4.2.1	Vægtcelle . . . . .	78
4.2.1.1	Enhedstest af vægtcelle . . . . .	78
4.2.2	Pumpe . . . . .	80
4.2.2.1	Enhedstest af motor og motordriver . . . . .	80
4.2.3	Ventil . . . . .	82
4.2.3.1	Enhedstest for ventil . . . . .	82
4.2.4	Kameralys . . . . .	83
4.2.4.1	Enhedstest for Kameralysset . . . . .	83
4.2.5	Ikke-elektroniske dele . . . . .	84
4.2.5.1	Stykliste . . . . .	85
4.2.6	PCB design . . . . .	85
4.2.6.1	PCB kredsløbsdiagram . . . . .	86
4.2.6.2	PCB toplayout . . . . .	87
4.2.6.3	PCB bundlayout . . . . .	88
4.2.6.4	Stykliste til PCB: 30-77-1 . . . . .	89
4.2.6.5	Stykliste for stik til print . . . . .	89
4.3	Software . . . . .	90
4.3.1	Kamera . . . . .	90
4.3.1.1	Test . . . . .	90
4.3.1.2	Simulering af kamera . . . . .	92
4.3.2	Program flow . . . . .	96
4.3.3	Data struktur . . . . .	97
4.3.4	Matlab funktioner . . . . .	98
4.3.4.1	initArduino . . . . .	98
4.3.4.2	constants . . . . .	99
4.3.4.3	cameraFeed . . . . .	99
4.3.4.4	detectIslets . . . . .	100
4.3.4.5	loadCell . . . . .	106
4.3.4.6	valveControl . . . . .	108
4.3.4.7	pumpControl . . . . .	110
4.3.4.8	exportData . . . . .	110
4.3.4.9	areaConverter . . . . .	110
4.3.5	GUI . . . . .	111

4.3.5.1	Hovedvindue . . . . .	111
4.3.5.2	Indstillingsvindue . . . . .	111
4.4	Integrationstest . . . . .	112
4.4.1	Software . . . . .	112
4.4.2	Hardware . . . . .	112
<b>Litteratur</b>		<b>113</b>
<b>Appendiks A Bilag</b>		<b>115</b>
A.1	Datablade . . . . .	115
A.1.1	INA114 . . . . .	115
A.1.2	L293D . . . . .	115
A.1.3	L5-W55N-BVW . . . . .	115
A.1.4	161T031 . . . . .	115
A.2	Matlab kode . . . . .	115
A.3	Arduino Testkode . . . . .	115
A.3.1	Kode til enhedstest til vægtcelle.pdf . . . . .	115
A.3.2	Kode til enhedstest til pumpe.pdf . . . . .	115
A.3.3	Kode til enhedstest til ventil.pdf . . . . .	115

# Kravspecifikation

1

## 1.1 Indledning

Dette dokument indeholder kravspecifikationen for *The Cell Collector* (omtales herefter som systemet). Dokumentet er udarbejdet i samarbejde med projektets kunde (Søren Gregersen).

### 1.1.1 Formål

Formålet med dokumentet er at beskrive systemets funktionelle og ikke funktionelle krav. Kravene er specifieret ud fra kundens kvalitetskrav. Der er i forlængelse af kravspecifikationen udarbejdet en accepttest, som har til formål at teste de specificerede krav i kravspecifikation.

### 1.1.2 Læsevejledning

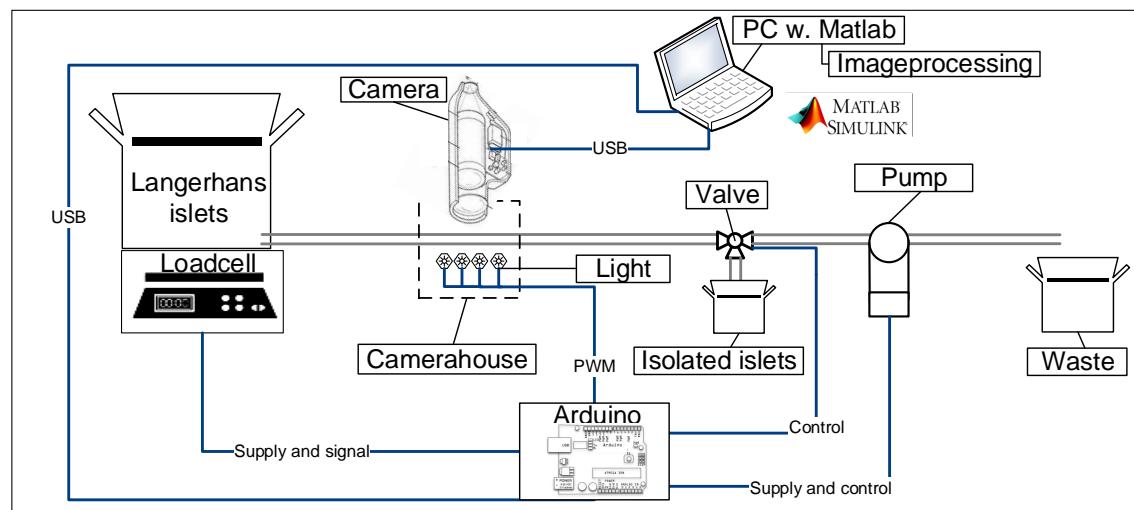
Dokumentet indeholder en systembeskrivelse, som kort beskriver systemets opbygning. De enkelte krav er opdelt i funktionelle og ikke funktionelle krav, hvor de funktionelle krav er beskrevet ved hjælp af fully dressed use case beskrivelser. Dokumentet indeholder herudover en projektafgrænsning i form af MoSCoW modellen, samt et afsnit om projektets samarbejdspartnere.

### 1.1.3 Versionshistorik

Version	Dato	Beskrivelse	Initialer
0.1	19/09 2015	Dokument sendt til review	AE og AT
1.0	19/09 2015	Rettelser fra reviewmøde og Latex layout	AE og AT
1.1	20/10 2015	Kamera krav tilføjet	AE og AT
2.0	6/11 2015	Definitioner og layout ændringer	AE og AT
2.1	30/11 2015	Opdateret kamera krav	AE og AT

## 1.2 Systembeskrivelse

Formålet med projektet er at udvikle et system til isolation af insulin producerende celler (Langerhanske Øer). Mange farmaceutiske virksomheder og forskningsafdelinger udfører forsøg på disse øer fra bl.a. mus og rotter. Processen med isolering af Langerhanske øer startes ved operativt at fjerne pancreas, hvorefter vævet opløses vha. enzymet kollagenase. Når vævet er opløst fortyndes det yderligere inden det hældes i petriskål. Øerne bliver herefter manuelt isoleret vha. mikroskop og diverse præcisions redskaber. Denne proces er både besværlig og tidskraevende. Formålet med projektet er derfor at udvikle en ny metode til isolation af cellerne. Systemet skal indeholde en beholder til opløsningen med langerhanske øer. Denne opløsning skal føres ud gennem en tynd slange( $\varnothing < 0.5$  mm) forbi et kamera, hvor der ved hjælp af Matlab skal udføres billedprocessering. Billedbehandlingen skal genkende, hvornår der er en langerhansk ø. Derefter skal systemet frasortere denne, ved et ventilsystem der åbner på det rigtige tidspunkt. Til at skabe flowet i slangerne anvendes en pumpe. Et krav til pumpen er, at den skal være nænsom ved celleopløsningen, da de langerhanske øer er meget skrøbelige. En automatiseret løsning af sorteringsprocessen kan bl.a. med reducering af omkostningerne, give en mere ensartet sortering samt sikre dokumentation af de sorterede øer. Systemet kan fra et kommersIELT synspunkt bidrage til basal forskning og til screening af nye medicinske præparater.



**Figur 1.1.** Figuren viser den overordnede opbygning af systemet, som beskrevet under systembeskrivelsen

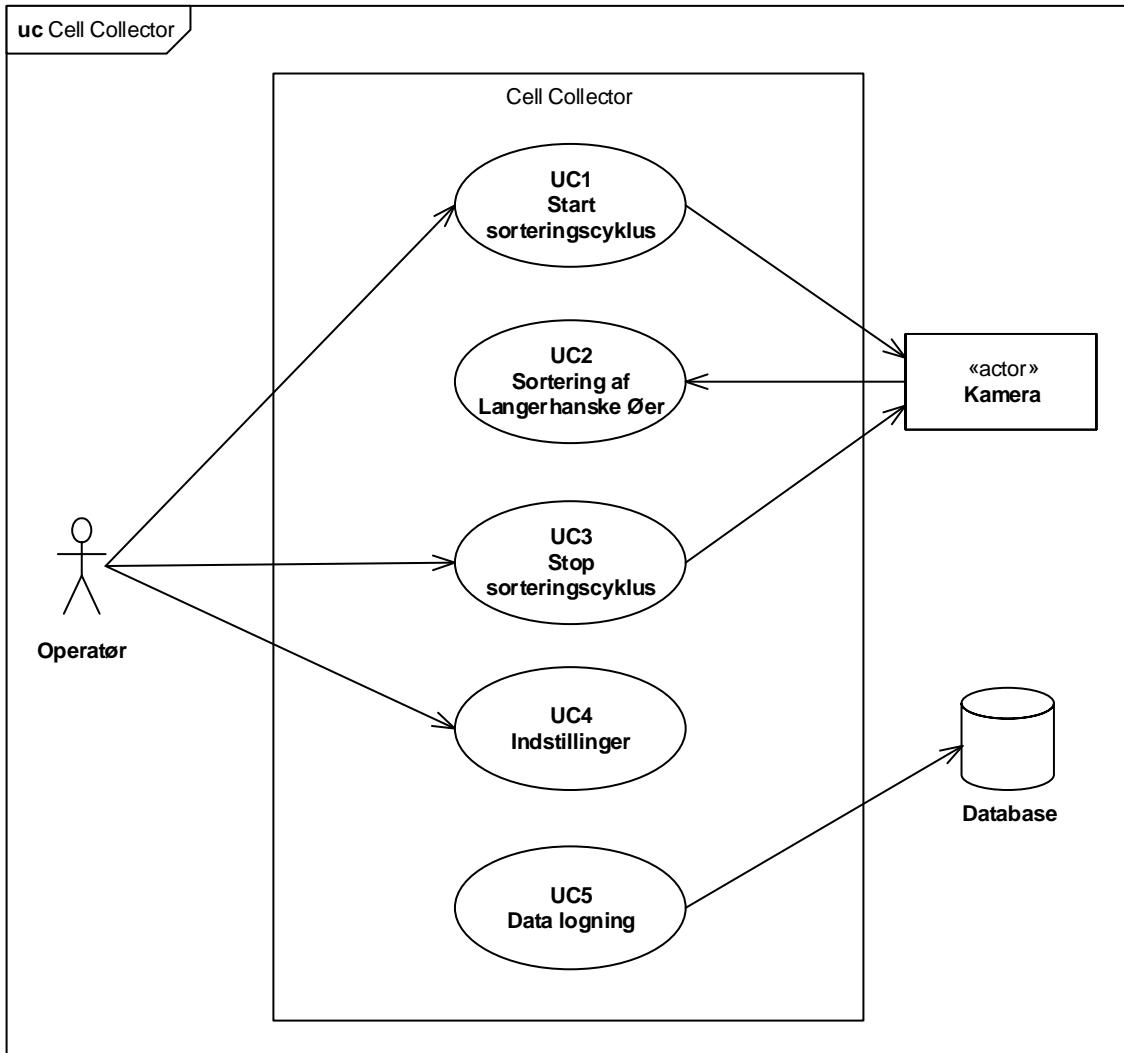
### 1.2.1 Aktør beskrivelse

Systemets primære aktør er operatøren, som står for påfyldning af celler samt start og stop af sorteringsprocessen. Operatøren har mulighed for at interagere med systemet via en grafisk brugergrænseflade. Systemets sekundære aktør er kameraet og PC'ens filsystem. Kameraet er systemets interface til detektion af de Langerhanske øer. Filsystemet er hvor der gemmes en log over sorteringsprocessen.

## 1.3 Funktionelle krav

### 1.3.1 Use case diagram

I Use case diagrammet (figur: 1.2) er der vist, hvilke use cases systemet *The Cell Collector* består af. Yderligere er det vist, hvilke aktører der initierer de enkelte use cases. På venstre side er systemets primære aktør *operatøren* vist, mens systemets sekundære aktører *kamera* og *database* er placeret i højre side.



**Figur 1.2.** Use case diagram for The Cell Collector

### 1.3.2 Use case 1 - Start sorteringscyklus

Mål	Start sorteringscyklus
Initiering	Use casen initieres af operatøren
Aktør	Primær: Operatør Sekundær: Kamera
Startbetingelser	<i>The Cell Collector</i> programmet er startet på computeren
Slutbetingelser ved succes	Systemet starter med sorteringen af Langerhanske øer
Slutbetingelser ved undtagelse	N/A
Normalforløb	<ol style="list-style-type: none"> <li>1. Operatør fylder celleopløsningsbeholderen</li> <li>2. Celleopløsningsbeholderen er fyldt</li> <li>3. Operatør starter sorteringscyklus ved at klikke på [Start]</li> </ol> <p>[Undtagelse 1: Wastebeholder er fyldt]</p> <ol style="list-style-type: none"> <li>4. Systemet initialiserer Arduinoen</li> </ol> <p>[Undtagelse 2: Ingen forbindelse til Arduino]</p> <ol style="list-style-type: none"> <li>5. Systemet kontrollerer celleopløsningsbeholderen ved at konvertere spændingen (V) til ml, og vise beholderens indhold (ml) på GUI</li> <li>6. Systemet initialiserer kameraet</li> </ol> <p>[Undtagelse 3: Kameraet initialiserer ikke]</p> <ol style="list-style-type: none"> <li>7. Systemet tænder for kameralyset</li> <li>8. Systemet tænder for pumpen</li> </ol>
Undtagelser	<p>[Undtagelse 1: Wastebeholder er fyldt]</p> <ol style="list-style-type: none"> <li>1. Systembesked: Tøm venligst Wastebeholder før start</li> <li>2. Operatøren trykker "OK"</li> <li>3. Systemet fortsætter opstartprocessen</li> </ol> <p>[Undtagelse 2: Ingen forbindelse til Arduino]</p> <ol style="list-style-type: none"> <li>1. Systembesked: Ingen forbindelse til Arduino, kontrollér forbindelser.</li> </ol> <p>[Undtagelse 3: Kameraet initialiseres ikke]</p> <ol style="list-style-type: none"> <li>1. System fejlmeddelse: Kameraet er ikke initialiseret</li> </ol>

### 1.3.3 Use case 2 - Sortering af Langerhanske Øer

Mål	Sortere Langerhanske Øer
Initiering	Use casen initieres af [UC 1: Startsorteringscyklus]
Aktør	Kamera
Startbetingelser	Systemet er startet og sorteringscyklussen er i gang
Slutbetingelser ved succes	Systemet har isoleret en Langerhansk ø og ventilen er lukket
Slutbetingelser ved undtagelse	
Normalforløb	<ol style="list-style-type: none"> <li>1. Systemet detekterer en Langerhansk ø</li> <li>2. Arduino sender signal til ventilen om åbning</li> <li>3. Ventilen åbner</li> <li>4. Arduino sender signal til ventilen om lukning</li> <li>5. Ventilen lukker</li> </ol>
Undtagelser	

### 1.3.4 Use case 3 - Stop sorteringscyklus

Mål	Stop sorteringscyklus
Initiering	Use casen initieres af operatøren
Aktør	Primær: Operatør Sekundær: Kamera
Startbetingelser	[UC 2: Sortering af Langerhanske Øer] er startet
Slutbetingelser ved succes	[UC 2: Sortering af Langerhanske Øer] er stoppet
Slutbetingelser ved undtagelse	N/A
Normalforløb	<ol style="list-style-type: none"> <li>1. Operatør stopper sorteringscyklussen ved at trykke på [Stop]            [Udvidelse 1: Tom celleopløsningsbeholder]</li> <li>2. Systemet slukker for pumpen</li> <li>3. Systemet slukker for kameraet</li> <li>4. Systemet slukker for kameralyset</li> <li>5. Systemet slukker for Arduino</li> </ol>
Undtagelser	[Udvidelse 1: Tom celleopløsningsbeholder] <ol style="list-style-type: none"> <li>1. Systemet slukker for pumpen</li> <li>2. Systemet slukker for kameraet</li> <li>3. Systemet slukker for kameralyset</li> <li>4. Systemet slukker for Arduino</li> </ol>

### 1.3.5 Use case 4 - Indstillinger

Mål	Ændre systemets indstillinger
Initiering	Use casen initieres af operatør
Aktør	Operatør
Startbetingelser	[UC 2: Sortering af Langerhanske Øer] er endnu ikke startet
Slutbetingelser ved succes	Systemets indstillinger er ændret
Slutbetingelser ved undtagelse	Systemets indstillinger er uændret
Normalforløb	<ol style="list-style-type: none"> <li>1. Operatøren klikker på [Indstillinger]</li> <li>2. Et nyt vindue åbner med systemets indstillinger.</li> <li>3. Operatøren vælger de ønskede indstillinger, og trykker [Gem indstillinger] [Undtagelse 1: Operatøren klikker [Annuler]]</li> <li>4. Systemets indstillinger gemmes.</li> </ol>
Undtagelser	<p>[Undtagelse 1: Operatøren klikker “Annuler”]</p> <ol style="list-style-type: none"> <li>1. Systemet lukker Indstillingsvinduet og indstillingerne er uændret.</li> </ol>

### 1.3.6 Use case 5 - Data logging

Mål	Logning af data
Initiering	Use casen initieres af systemet ved [UC 3: Stop sorteringscyklus]
Aktør	Database
Startbetingelser	[UC 2: Sortering af Langerhanske Øer] er stoppet
Slutbetingelser ved succes	Systemet har gemt fil med data for sorteringen
Slutbetingelser ved undtagelse	
Normalforløb	<ol style="list-style-type: none"><li>1. Systemet gemmer en fil i formatet .csv af data for den afsluttede sorteringscyklus (Struktur og indhold er beskrevet under ikke funktionelle krav: 1.4.3.1)</li><li>2. Systemet informerer brugeren om at filen er gemt</li></ol>
Undtagelser	

## 1.4 Ikke funktionelle krav

### 1.4.1 Kvalitetskrav

Systemet har følgende krav fra kunden

Nr	Krav	Beskrivelse	Kommentar
1	Hastighed	Hastigheden på systemet skal være højere end 30 øer sorteret pr. minut	
2	Renhed	2.1 mere end 90 % af de isolerede øer skal være faktiske øer (Sandt pos: > 90 %) 2.2 der skal være mindre end 5 % af de isolerede øer, der ikke er øer (Falsk pos: < 5 %) 2.3 der skal være mindre end 5 % af øerne i opløsningen der ikke er blevet isoleret (Falsk neg: < 5 %)	Dokumentation af renhed: 1. Vurdering af erfaren ø-plukker. 2. Opmåling vha. digital billedbehandlingssoftware [?]. 3. Funktionstests i laboratoriet [?] [?].
3	Isoleringsgrad	Over 90 % af det oprindelige antal skal være isoleret	$\frac{\text{Antal isolerede}}{\text{Total antal i opløsning}} * 100$
4	Genkendelsesgrad	Over 90 % af det oprindelige antal skal være isoleret	$\frac{\text{Visionsgenkendte}}{\text{Total antal i opløsning}} * 100$
5	Ø/Cellestørrelse (µm)	Systemet skal minimum kunne sortere øer, der har en størrelse mellem 100 µm og 300 µm	
6	Datalogning	Systemet skal kunne logge informationer omkring opløsningens øer, både størrelse og form	
7	Rensning	Systemet skal kunne lave en automatisk rensning af rør mm.	
8	Køling	Systemet skal kunne køle opløsningensvæsken.	

## 1.4.2 Hardware

### 1.4.2.1 Microcontroller

1. Atmega328p (Arduino)

### 1.4.2.2 Pumpe

1. Pumpe flow: <50ml / min
2. Størrelse på studserne skal kunne tilpasses slangerne

### 1.4.2.3 Slanger

1. Slangerne skal have en indre diameter > 300 $\mu\text{m}$
2. Kameraet skal kunne detektere langerhanske øer igennem slangen, evt. vha. glasrør

### 1.4.2.4 Beholdere

1. Celleopløsningsbeholder skal have størrelse > 250 mL
2. Wastebeholder skal have en størrelse dobbelt så stor som celleopløsningsbeholderen:  
    > 500 mL

### 1.4.2.5 Ventil

1. 3-vejs, dvs. 1 tilgang og kobling mellem 2 udgange
2. Studserne skal kunne tilpasses slangerne
3. Skal være til væske
4. Lukke og åbne tid skal være >50ms

### 1.4.2.6 Kamera

1. Kameraet til kunne detektere langerhanske øer mellem 100 og 300 $\mu\text{m}$
2. Kameraet skal have en zoom funktion, som et mikroskop
3. Kameraet skal have et USB interface

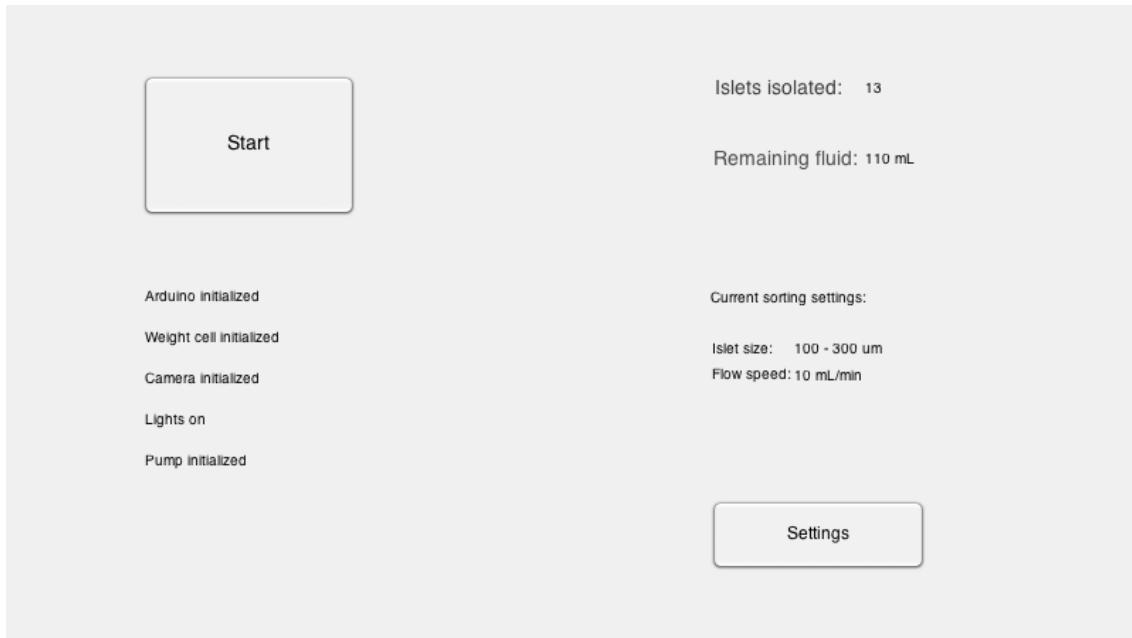
## 1.4.3 Software

### 1.4.3.1 Dataformat og struktur

1. CSV format med kommasepareret delimiter.
2. Filnavn: Dato og starttidspunkt for sorteringscyklus.
3. Header indeholdende opsætningsindstillinger.
4. Filen er opbygget med følgende kolonner:
  - a) Tidsstempel i formatet DD-MM-YYYY-hh:mm:ss
  - b) Ø størrelse
  - c) Radius
  - d) Omkreds
  - e) Middelintensitet

#### 1.4.4 GUI - Mockup

Mockup af GUI



*Figur 1.3.* Mockup af GUI

## 1.5 Projektafgrænsning

Til at afgrænse projektet anvendes MoSCoW modellen, som beskriver hvilke dele projektet skal (**Must**), bør (**Should**), kan (**Could** og ikke (**Won't**) indeholde. MoSCoW modellen (figur: 1.4) viser hvordan de enkelte krav og dele af projektet er prioriteret. Denne metode er brugt for at give en struktureret oversigt over hvilke krav der er vigtigst at få opfyldt inden for tidsrammen og hvilke der evt. kan implementeres senere hvis tiden er til det.



*Figur 1.4.* MoSCoW

## 1.6 Samarbejdspartnere

Gruppens kunde er Søren Gregersen, overlæge på Medicinsk Endokrinologisk Afdeling, Aarhus Universitetshospital. Det er i samarbejde med ham at projektet er blevet specificeret, samt hvilke krav der er til den endelige prototype. Samuel Alberg Thryssøe er gruppens projektvejleder. Der afholdes ugentligt et vejledermøde, hvor gruppen giver status på projektet og hvor der diskutes forskellige problemstillinger. Simon Vammen Grønbæk og Karl Johan Schmidt fungerer som projektets reviewgruppe. Der holdes møde hver anden uge omhandlende aftalt dagsorden. Formålet med reviewgruppen er at få konstruktiv feedback på evt. rettelser, opbygning af rapport og generel forståelse.

# Accepttest 2

---

## 2.1 Indledning

Dette dokument indeholder accepttesten for *The Cell Collector* (omtales herefter som systemet).

### 2.1.1 Formål

Formålet med dokumentet er at sikre, at alle krav til produktet er opfyldt, i henhold til kravspecifikationen.

### 2.1.2 Læsevejledning

Dokumentet indeholder testcases for de enkelte krav, som specifiseret i kravspecifikationen. Accepttesten er opdelt i henholdsvis funktionelle og ikke funktionelle krav. Testcasene er beskrevet i tabeller, hvor der ved hver test er noteret krav nr., handling, forventet resultat og testmetode. Herudover er det faktiske resultat noteret og om testen er godkendt eller ej. Yderligere er initialer og dato noteret, som angiver hvornår den enkelte test er udført.

### 2.1.3 Versionshistorik

Version	Dato	Beskrivelse	Initialer
0.1	19/09 2015	Dokument sendt til review	AE og AT
1.0	19/09 2015	Rettelser fra reviewmøde og Latex layout	AE og AT
1.1	03/10 2015	Tilføjelse af kamera og dataformat krav	AE og AT
2.0	06/11 2015	Definitioner og layout ændringer	AE og AT
2.1	30/11 2015	Opdateret kamera krav	AE og AT
2.2	01/12 2015	Tilretning af testmetoder	AE og AT

## 2.2 Accepttest af funktionelle krav

Accepttesten udføres i samarbejde med vejlederen til projektet, som skal agerer kunde og dermed godkende testen. Testen udføres ved at følge handlingen i nedenstående tabeller, hvor forventede resultat sammenlignes med det givende resultat i testen. Er resultatet som forventet godkendes testen ved at sætte flueben, dato og initialer for personen der udfører testen. Modsat hvis resultatet ikke er som forventet, skal testen markeres med rødt og der laves en fejlrapport. I fejlrapporten beskrives hvad der fejlede ved testen og en handlingsplan for, hvad der skal ændres for at få den godkendt.

### 2.2.1 Use case 1: Start sorteringscyklus

<b>Krav nr.</b>	1.1 & 1.2
<b>Handling</b>	Operatør fylder celleopløsningsbeholderen
<b>Forventet resultat</b>	Celleopløsningsbeholderen er fyldt
<b>Testmetode</b>	Celle opløsningsbeholderen fyldes med væske
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	1.3
<b>Handling</b>	Operatør starter sorteringscyklus ved at klikke på [Start]
<b>Forventet resultat</b>	Opstartsprocessen igangsættes.
<b>Testmetode</b>	Knappen [Start] trykkes, resultatet observeres ved tekstboks på GUI, med teksten <i>systemet starter op</i> .
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	1.4
<b>Handling</b>	Systemet initialiserer Arduinoen
<b>Forventet resultat</b>	Arduino initialiseret signal modtages og gives til GUI
<b>Testmetode</b>	Det observeres på GUI, at Arduinoen er initialiseret, i en tekstboks med teksten <i>Arduino er initialiseret.</i>
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	1.5
<b>Handling</b>	Systemet kontrollerer celleopløsningsbeholderen
<b>Forventet resultat</b>	Antal ml i celleopløsningsbeholderen vises på GUI.
<b>Testmetode</b>	Der hældes 100 ml i celleopløsningsbeholderen. Det observeres på GUI om der vises 100 ml $\pm$ 10 ml
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	1.6
<b>Handling</b>	Systemet initialiserer kameraet
<b>Forventet resultat</b>	Kamera initialiseret signal modtages og gives til UI
<b>Testmetode</b>	Det observeres på GUI, at kameraet er initialiseret, i en tekstboks med teksten <i>Kameraet er initialiseret</i>
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	1.7
<b>Handling</b>	Systemet tænder kameralyset
<b>Forventet resultat</b>	Lyset tænder
<b>Testmetode</b>	Kameralyset observeres
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	1.8
<b>Handling</b>	Systemet tænder pumpen
<b>Forventet resultat</b>	Pumpen starter
<b>Testmetode</b>	Det observeres ved at se flowet i slangerne stiger
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.1
<b>Handling</b>	Waste-beholderen er fyldt
<b>Forventet resultat</b>	System-besked: Tøm waste-beholderen
<b>Testmetode</b>	Der gennemføres 2 sorteringscyklusser. Ved start af 3. sorteringscyklus observeres resultatet på GUI ved en messagebox.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.2 & 1.3
<b>Handling</b>	Operatør trykker [OK]
<b>Forventet resultat</b>	Systemet fortsætter opstartprocessen, observeres ved at næste komponent initialiseres, skrives i tekstboks på GUI
<b>Testmetode</b>	Knappen [OK] trykkes på messagebox.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 2
<b>Handling</b>	Forbindelsen til Arduino frakobles
<b>Forventet resultat</b>	System besked: Arduino kunne ikke initialiseres. Kontrollér forbindelse. Vil du prøve igen?
<b>Testmetode</b>	USB kablet til Arduinoen frakobles før der trykkes [Start]. Resultatet observeres på GUI ved en dialogboks.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 3
<b>Handling</b>	Kameraet initialiseres ikke
<b>Forventet resultat</b>	System-besked: Kameraet kunne ikke initialiseres. Kontrollér forbindelse. Vil du prøve igen?
<b>Testmetode</b>	USB kablet til kameraet frakobles før der trykkes [Start]. Resultatet observeres på GUI ved en dialogboks
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

### 2.2.2 Use case 2: Sortering af langerhanske øer

<b>Krav nr.</b>	2.1
<b>Handling</b>	Systemet detekterer en Langerhansk ø.
<b>Forventet resultat</b>	Tælleren for antal sorterede øer stiger
<b>Testmetode</b>	Sorteringscyklussen er startet. Det observeres på GUI, at tælleren for antal detekterede øer stiger.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	2.2 & 2.3
<b>Handling</b>	Arduino sender signal til ventilen om åbning.
<b>Forventet resultat</b>	Ventilen er åben
<b>Testmetode</b>	Observeres ved at se, at der løber væske ned i beholderen med isolerede øer
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	2.4 & 2.5
<b>Handling</b>	Arduino sender signal til ventilen om lukning.
<b>Forventet resultat</b>	Ventilen er lukket
<b>Testmetode</b>	Observeres ved at se væsken løber ud i wastebeholderen
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

### 2.2.3 Use case 3: Stop sorteringscyklus

<b>Krav nr.</b>	3.1
<b>Handling</b>	Operatør stopper sorteringscyklussen ved at trykke på [Stop]
<b>Forventet resultat</b>	Sorteringscyklussen stopper
<b>Testmetode</b>	En sorteringscyklus er i gang. Knappen [Stop] trykkes. Resultatet observeres på GUI med teksten <i>Systemet stopper</i>
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	3.2
<b>Handling</b>	Systemet slukker for pumpen
<b>Forventet resultat</b>	Flowet i slangene stopper
<b>Testmetode</b>	Observeres ved at se, at flowet i slangene stopper.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	3.3
<b>Handling</b>	Systemet slukker for kameraet
<b>Forventet resultat</b>	Kameraets sluk signal modtages og gives til UI
<b>Testmetode</b>	Det observeres på GUI, at kameraet er slukket
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	3.4
<b>Handling</b>	Systemet slukker for kameraanalyset
<b>Forventet resultat</b>	Lyset slukker
<b>Testmetode</b>	Kameraanalyset observeres
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	3.5
<b>Handling</b>	Systemet slukker for Arduinoen
<b>Forventet resultat</b>	Arduino sluk signal modtages og gives til UI
<b>Testmetode</b>	Det observeres på GUI, at Arduinonen er slukket, ved tekstboks med teksten <i>Arduino slukker</i> .
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.1
<b>Handling</b>	Celleopløsningsbeholderen løber tør for væske
<b>Forventet resultat</b>	Sorteringscyklussen stopper
<b>Testmetode</b>	En sorteringscyklus er i gang. Sorteringscyklussen forsættes indtil celleopløsningsbeholderen løber tør for væske. Resultatet observeres på GUI med teksten <i>Systemet stopper</i> .
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.2
<b>Handling</b>	Systemet slukker for pumpen
<b>Forventet resultat</b>	Flowet i slangens stopper
<b>Testmetode</b>	Observeres ved at se, at flowet i slangens stopper.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.3
<b>Handling</b>	Systemet slukker for kameraet
<b>Forventet resultat</b>	Kameraets sluk signal modtages og gives til UI
<b>Testmetode</b>	Det observeres på GUI, at kameraet er slukket ved, at feedet forsvinder
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.4
<b>Handling</b>	Systemet slukker for kameralyset
<b>Forventet resultat</b>	Lyset slukker
<b>Testmetode</b>	Kameralyset observeres
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1.5
<b>Handling</b>	Systemet slukker for Arduinoen
<b>Forventet resultat</b>	Arduino sluk signal modtages og gives til UI
<b>Testmetode</b>	Det observeres på GUI, at Arduinonen er slukket, ved tekstboks med teksten <i>Arduino slukker</i> .
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

#### 2.2.4 Use case 4: Indstillinger

<b>Krav nr.</b>	4.1 & 4.2
<b>Handling</b>	Operatøren klikker på [Indstilligner].
<b>Forventet resultat</b>	Et nyt vindue åbner med systemets indstillinger.
<b>Testmetode</b>	Knappen [Indstillinger] trykkes og det observeres på GUI, at indstillingsvinduet åbner
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	4.3 & 4.4
<b>Handling</b>	Operatøren vælger de ønskede indstillinger og trykker [Gem indstillinger].
<b>Forventet resultat</b>	Systemets indstillinger gemmes
<b>Testmetode</b>	Knappen [Gem Indstillinger] trykkes. Det verificeres, at indstillingerne er ændret ved at åbne Indstillinger igen.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	Undtagelse 1
<b>Handling</b>	Operatøren klikker [Annuler].
<b>Forventet resultat</b>	Indstillingsvinduet lukkes og systemets indstillinger er uændret.
<b>Testmetode</b>	Knappen [Annuler] trykkes. Det verificeres, at indstillerne er uændret ved at åbne Indstillinger igen.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

### 2.2.5 Use case 5: Data logging

<b>Krav nr.</b>	5.1
<b>Handling</b>	Systemet gemmer en fil i formatet .csv for sorteringsscyklussen
<b>Forventet resultat</b>	Filen er gemt i databasen
<b>Testmetode</b>	En ny sorteringsscyklus startes (UC 1), hvorefter sorteringsscyklussen stoppes (UC 3) ved tryk på [Start] og [Stop]. Det observeres ved styresystemets filsystem, at filen er gemt
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	5.2
<b>Handling</b>	Systemet informerer operatøren om, at filen er gemt.
<b>Forventet resultat</b>	Der vises besked til operatøren.
<b>Testmetode</b>	En ny sorteringsscyklus startes (UC 1), hvorefter sorteringsscyklussen stoppes (UC 3) ved tryk på [Stop]. Observer GUI for messagebox.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

## 2.3 Accepttest af ikke funktionelle krav

<b>Krav nr.</b>	1
<b>Kvalitetskrav</b>	Hastigheden på systemet skal være højere end 30 øer sorteret pr. minut
<b>Testmetode</b>	Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5), hvor der måles med et stopur. Stopuret stoppes efter sorteringsprocessen er færdig. Derefter regnes hastighed ud ved $\frac{\text{Antal øer}}{\text{Minutter}} > 30$
<b>Forventet resultat</b>	Når sorteringscyklussen er færdig er $\frac{\text{Antal øer}}{\text{Minutter}} > 30$
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	2.1
<b>Kvalitetskrav</b>	Mere end 90% af de isolerede øer skal være faktiske øer (Sandt pos: $> 90\%$ )
<b>Testmetode</b>	Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5). Efter endt cyklus skal en kyndig person tælle antallet af faktiske øer. Dette holdes op i mod antallet af isolerede øer.
<b>Forventet resultat</b>	Når sorteringscyklussen er færdig er $\frac{\text{Antal talte øer}}{\text{Antal isoleret}} * 100 \Rightarrow 90$
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	2.2
<b>Kvalitetskrav</b>	Der skal være mindre end 5% af de isolerede øer, der ikke er øer (Falsk pos: < 5%)
<b>Testmetode</b>	Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5). Efter endt cyklus skal en kyndig person tælle antallet af faktiske øer. Dette holdes op i mod antallet af isolerede øer.
<b>Forventet resultat</b>	Når sorteringscyklussen er færdig er $\frac{\text{Antal talte øer}}{\text{Antal isoleret}} * 100 \Rightarrow 95$
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	2.3
<b>Kvalitetskrav</b>	Der skal være mindre end 5% af øerne i opløsningen der ikke er blevet isoleret (Falsk neg: < 5%)
<b>Testmetode</b>	Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5). Efter endt cyklus skal en kyndig person tælle antallet af øer, der er isoleret og antallet af øer i waste beholderen.
<b>Forventet resultat</b>	Når sorteringscyklussen er færdig er $\frac{\text{Antal talte øer i waste}}{\text{Antal talte isoleret øer}} * 100 \Rightarrow 95$
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	3
<b>Kvalitetskrav</b>	over 90% af det oprindelige antal skal være isoleret.
<b>Testmetode</b>	En opløsning med et kendt antal øer benyttes. Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5). Efter endt cyklus skal en kyndig person tælle antallet af øer der er isoleret. Dette antal holdes op i mod antallet af øer i opløsningen fra start.
<b>Forventet resultat</b>	Når sorteringscyklussen er færdig er $\frac{\text{Antal talte øer i opløsningen fra start}}{\text{Antal talte isoleret øer}} * 100 \Rightarrow 90$
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	4
<b>Kvalitetskrav</b>	over 90% af det oprindelige antal, skal være genkendt
<b>Testmetode</b>	En opløsning med et kendt antal øer benyttes. Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5). Efter endt cyklus holdes antal af detekterede øer op imod det oprindelige antal øer fra starten.
<b>Forventet resultat</b>	Når sorteringscyklussen er færdig er $\frac{\text{Antal detekteret øer}}{\text{Antal oprindelige øer}} * 100 \Rightarrow 90$
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	5
<b>Kvalitetskrav</b>	Systemet skal minimum kunne sortere øer, der har en størrelse mellem 100 µm og 300 µm
<b>Testmetode</b>	En oplösning med en ø størrelse på 100 µm og 300 µm benyttes. Normalforløbet ved en sorteringscyklus følges (Use Case 2: Sortering af Langerhanske Øer, s. 5). Efter endt cyklus observeres det om systemet har sorteret de specificerede størrelser.
<b>Forventet resultat</b>	Begge ø størrelser er isoleret
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	

<b>Krav nr.</b>	6
<b>Kvalitetskrav</b>	Systemet skal kunne give informationer omkring oplösningens øer, både størrelse og form.
<b>Testmetode</b>	Normalforløbet ved en sorteringscyklus følges.
<b>Forventet resultat</b>	Efter endt cyklus skal data filen kontrolleres om den har de specificerede værdier.
<b>Resultat</b>	
<b>Angiv godkendelse</b>	
<b>Initialer</b>	
<b>Dato</b>	



# Design 3

---

## 3.1 Indledning

Dette dokument beskriver systemets design og systemarkitektur. Dokumentet indeholder en generel præsentation og beskrivelse af systemet, herunder hvordan *The Cell Collector* er opbygget både hardware og software mæssigt.

### 3.1.1 Formål

Design dokumentets formål er, at beskrive og fastlægge overordnede komponenter for hardwaren og softwaren, samt grænsefladerne mellem dem. Herudover skal dokumentet identificere arbejdsopgaver for projektets implementeringsfase.

### 3.1.2 Læsevejledning

Dokumentet skal ses som en uddybbende beskrivelse af kravene til systemet, der blev fastlagt i kravspecifikationen. Dokumentet er overordnet opdelt i henholdsvis en hardware beskrivelse og en software beskrivelse. De enkelte delelementer er beskrevet vha. SysML diagrammer. Alt SysML udvikles og skrives på engelsk.

### 3.1.3 Versionshistorik

Version	Dato	Beskrivelse	Initialer
1.0	23/10 2015	Første officielle dokument (efter review)	AE og AT
2.0	06/11 2015	Struktur og layout ændringer	AE og AT
2.1	13/11 2015	Ændringer i forhold til motordriver	AT
2.1	14/12 2015	Ventil og pumpe er opdateret(12V)	AT

## 3.2 Udviklingsværktøjer

### 3.2.1 MATLAB

Selve softwaren til systemet udvikles i MATLAB. Versionen der er anvendt er R2015b. Af tilføjelsespakker bruges følgene:

- Arduino Support Package (14.2.2)
- Webcam Support Package (15.2)

Disse pakker anvendes til, at interagere med Arduinoen og USB mikroskopet.

**Fritzing:** *Fritzing* er brugt til at illustrere kredsløbsdiagrammer(schematics), samt enhedstest af hardwaren ved illustrationer af *fumlebræt*. Dette har bidraget til dokumentationen af enhedstestene. Derudover har det givet mulighed for at opstillingerne brugt på *fumlebræt*, let kunne genskabes.

**Eagle:** *Eagle* er et program til udarbejdelse af print layouts. *Eagle* er brugt til at lave et samlet PCB layout for hardwarelementerne. I *Eagle* er der generet Gerber-filer, som er sendt til PCB fabrikanten.

**SolidWorks:** *SolidWorks* er et program til at illustrere mekaniske 3D modeller med. *SolidWorks* er brugt til at lave kamerahuset med for senere at få det 3D printet.

**Arduino IDE:** Programmet som er udbudt af *Arduino* bruges til at programmere *Arduino* udviklingskortet. *Arduino IDE* er i projektet brugt til enhedstest for at teste hardware opstillingerne.

**Microsoft Visio:** *Microsoft Visio* er et tegne program, som bruges til at illustrere forskellige modeller. Programmet er valgt til at lave udviklingsdiagrammer, samt illustrationer med.

**Texmaker:** Dette program er brugt til at skrive alt dokumentationen i projektet. *Texmaker* gør det muligt at arbejde på samme dokument samtidigt, hvilket har været en stor hjælp.

**PivotalTracker:** *PivotalTracker* er et scrumbaseret projektstyringsværktøj, der hjælper med at styre arbejdsressourcerne i projektet. Der er i afsnittet 3.2.3 uddybet hvordan *PivotalTracker* er brugt i projektet.

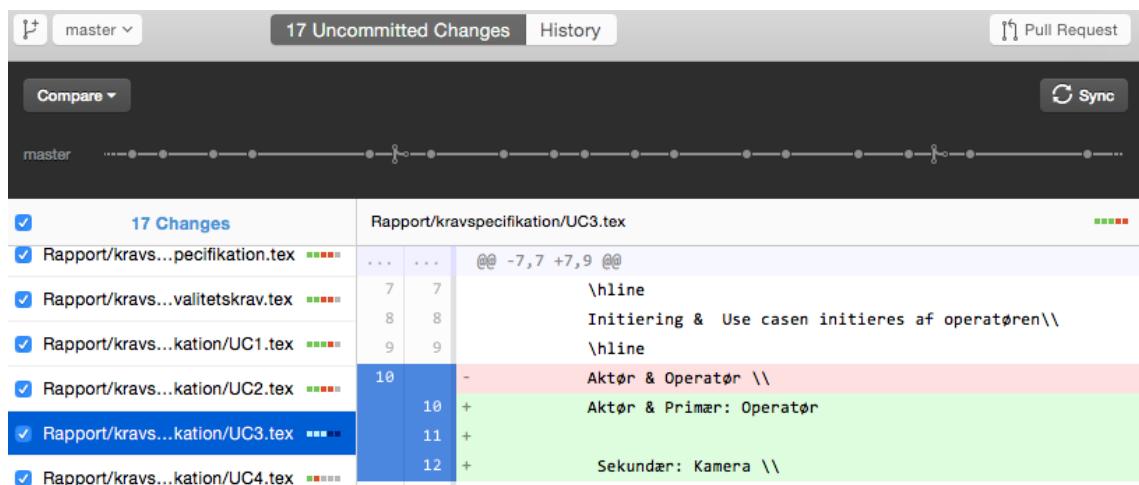
**GitHub:** Er et versionsstyringsprogram, som i projektet er brugt til versionsstyring af dokumenter og *Matlab* koden til projektet. Se afsnit 3.2.2 for uddybende dokumentation omkring hvordan *Github* er brugt.

**Dropbox:** Det cloudbaserede delingsværktøj *Dropbox* er i projektet brugt til at gemme og dele artikler, diagrammer, billeder og generelle noter.

**Microsoft Onenote:** *Onenote* er et program til at lave noter i. I projektet er programmet brugt til at have fælles noter og huskelister.

### 3.2.2 GitHub

Til versionsstyring af projektdokumentationen og source kode anvendes GitHub, som bygger på open source versions styrings systemet Git. Her opdateres der løbende ændringer, så det nyeste dokumentation og source kode altid er tilgængeligt. Som user interface til GitHub anvendes GitHub Desktop (figur: 3.1). I GitHub Desktop vises en tidslinje, for hvornår der er lavet ændringer. Under de enkelte filer kan man se hvad der er ændret i for den gældende version. Programmet giver yderligere indblik i hvilke filer der lokalt er lavet ændringer i, som ikke er tilføjet repositoriet endnu.



Figur 3.1. GitHub Desktop

### 3.2.3 Pivotal Tracker

Til projektstyring anvendes Pivotaltracker, som er et online værktøj baseret på SCRUM. I Pivotaltracker defineres projektets arbejdsopgaver, hvorefter de tildeles point alt efter hvor stor arbejdsbyrden er. De enkelte opgaver prioriteres herefter i projektets backlog, hvor Pivotaltracker automatisk tilføjer opgaver til den igangværende sprint udfra den nuværende “velocity”. En ny sprint påbegyndes automatisk når en ny uge starter.

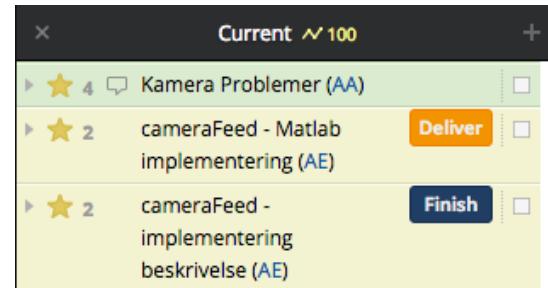
Det betyder, at der er fuldstændig styr på om projektet går for langsomt, eller om udviklingen af projektet er godt med. Dette kan sammenlignes med gatestate arbejds metoden, hvor der er flere deadlines end der vil være i for eksempel vandfaldbaserten metoden.

Herudover giver Pivotaltracker mulighed for en komplet log over projektets udførte opgaver og afsluttede sprints. Her kan man se hvilke opgaver der er udført i hvilken uge. I projektet anvendes dette som logbog over udførte arbejdsopgaver.

En opgave kan have forskellige states, som definerer dens status. Når en opgave er afsluttet kan den afleveres til review, hvor den herefter enten kan godkendes eller afvises. Dette er særligt anvendeligt i projektets udviklingsfase, hvor en feature kan testes og godkendes af et andet projektmedlem. Figur 3.2 viser et overblik over tidligere sprints, hvor figur 3.3 viser en igangværende sprint med opgaver der er godkendte, afsluttet og ikke færdiggjorte endnu.

▼ 8	19 Oct	Pts: 28
▶ ★ 4	Billeder af Langerhanske øer (AA)	[checkbox]
▶ ★ 8	Indkøb af ventil mm. (AA)	[checkbox]
▶ ★ 8	Skriv accepttest over i LaTeX (AA)	[checkbox]
▶ ★ 8	Skriv kravspec over i LaTeX (AA)	[checkbox]
▼ 9	26 Oct	Pts: 14
▶ ★ 2	BDD Software (AE)	[checkbox]
▶ ★ 2	IBD Software (AE)	[checkbox]
▶ ★ 2	Afsnit omkring udviklingsværktøjer (AA)	[checkbox]
▶ ★ 4	Ikke funktionelle krav (AE)	[checkbox]
▶ ★ 4	IBD for hardware (AE)	[checkbox]

*Figur 3.2.* Færdiggjorte sprints



*Figur 3.3.* Igangværende sprint

### 3.3 Hardware

Dette afsnit skal være med til at dokumentere hardwaren i systemet *the cell collector*, derfor indeholder afsnittet beskrivelser af systemets fysiske dele og deres funktionalitet. De fysiske deles specifikationer er uddybet og beskrevet i dette dokument. Der er begrundelser og argumenter for hvorfor de brugte komponenter er valgt. Desuden er der også et teori afsnit til hver komponent, for at dokumentere den forskning der er lavet.

#### 3.3.1 Læsevejledning til hardware

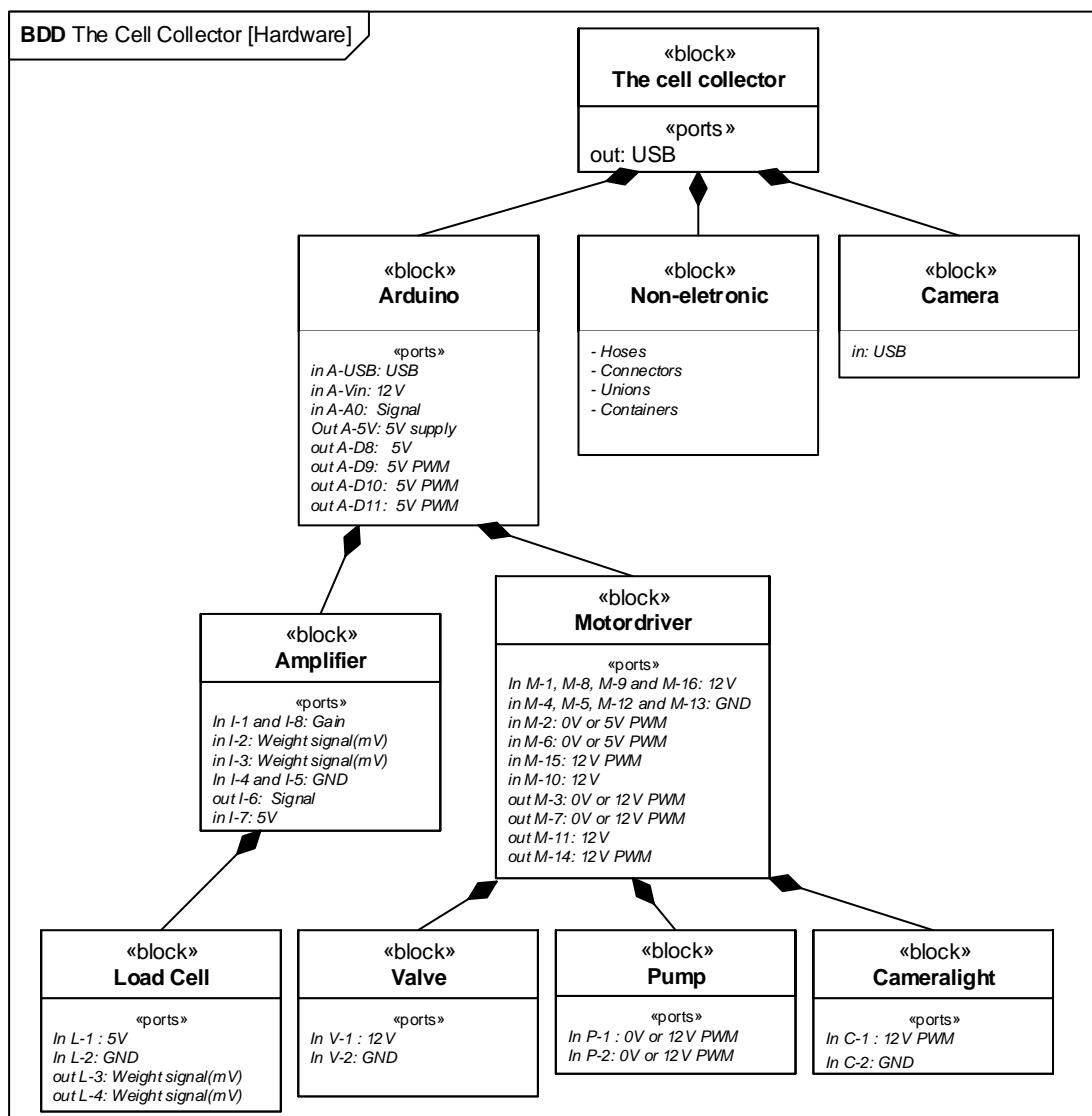
Da dette afsnit er en udspecifcering af hvilke specifikationer hardware komponenterne består af, ligger afsnittet sig tæt op af kravspecifikationen. Da det er kravene i kravspecifikationen, der ligger til grunde for de valgte hardware dele. I afsnittet findes der forskellige diagrammer der er med til at beskrive opbygningen, og kommunikationen af delene. Til hvert diagram vil der være en kort beskrivelse til indholdet. Hvert afsnit indeholder funktionalitet, specifikationer omkring det enkle komponent, samt begrundelser og argumentationer for valgende, i den skrevne rækkefølge. Til sidst er der et teori afsnit til hver komponent, hvilket kan springes over hvis viden her om allerede er etableret. Afsnittene kan godt læses i dele, men for en komplet forståelse bør afsnittene læses i rækkefølge.

#### 3.3.2 Leverandør af produkter

Når der skal vælges leverandør til et produkt er det vigtigt, at de er pålidelige. Da delene skal kunne leveres i hele produktets levetid. Specielt hvis produktet kræver en stor dokumentation og godkendelse, som for eksempel medicinsk udstyr. Når produktet skal sættes i produktion bør der være flere leverandører, som kan leverer det samme produkt. Det er vigtigt for, at dokumentationen ikke skal ændres, fordi at en lille del af produktet er udgået af underleverandørens portefølje. Til systemet *The cell collector*, har budgettet gjort stort præg af hvilke leverandører der er valgt. Primært er de fleste dele fundet på Ebay, for netop at holde budgettet og dermed priserne i bund. Dette medfører selvfølgelig at kvaliteten er nedprioriteret, som et kompromis er kameraet(3.3.5) i projektet købt ved Farnell. Farnell er en mere pålidelig forhandler end Ebay. At produkterne er købt på Ebay, vil medfører at der vil komme ændringer af dokumentation for at hæve kvaliteten på produktet i fremtiden. Ydermere er generelle ikke elektroniske komponenter købt ved Mikrolab, herunder slanger og beholdere.

### 3.3.3 Block Definition Diagram

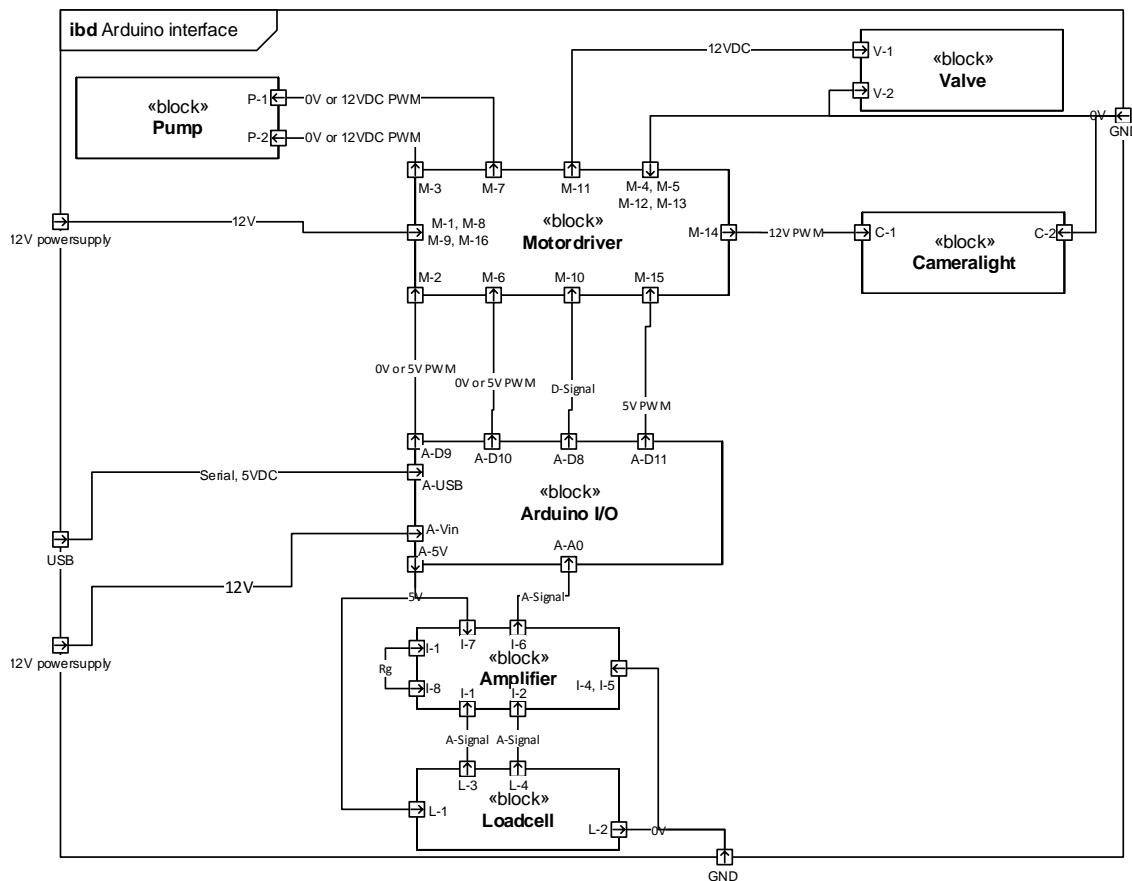
Nedenstående BDD 3.4 giver et overordnet overblik i, hvad *The cell collector* indeholder af hardware elementer. Hierarkiet starter øverst med *The cell collector*, som indeholder tre mindre hardware dele. styreenheden der er defineret som en arduino, der indeholder de elementerne styret af arduinoen bla. motordriveren . Motordriveren indeholder yderligere tre dele, som den styrer. Det vil sige at pumpen, ventilen og kameralyset bliver styret i gennem motordriveren. Udover styreenheden er der ikke elektriske dele, som beholdere og føringsveje til opløsningen med langerhanske øer. Til sidst er kameraet også en af de 3 under dele til *The cell collector*.



*Figur 3.4.* BDD - Cell Collector [Hardware]

### 3.3.4 Internal block Diagram

Nedenstående IBD 3.5 beskriver mere præcist, hvordan de forskellige komponenter interagerer med hinanden. Diagrammet er brugt til, at der tidligt i udviklingsforløbet bliver defineret hvilke spændinger og signaltyper systemet skal indeholde. Systemet skal indeholde bestemte typer for, at kunne kommunikere med de interne dele.



*Figur 3.5. IBD - Cell Collector [Hardware]*

Tabellen nedenfor er brugt til at skabe en oversigt, over hvilke signaler og funktioner den enkle blok indeholder.

Blok navn	Funktionsbeskrivelse	Signaler	Kommentar
Pump	Skabe flow i opløsningen	0V 12VDC PWM	Reference Strømforsyning
Valve	Åbne og lukke for føringsveje	0V 12VDC	Reference Strømforsyning
Motor shield	Forsyne ventil og pumpe	0V 12V 0V or 5V PWM 0V or 5V PWM Signal 12V PWM 0V or 12VDC PWM 0V or 12VDC PWM	Reference Strømforsyning Motor retning Hastighedsregulering Tænd/sluk ventil lysdiode lysintensitet Motor retning Hastighedsregulering
Arduino	Styreenhed for hardware	0V 12V USB A-Signal D-Signal	Reference Strømforsyning Seriel kommunikation og 5V Aanalog indgangssignaler Digital udgangssignaler
Loadcell	Kontroller opløsningsbeholder	0V 5V Signal	Reference Strømforsyning Udgangssignal
Cameralight	Lys til kameraet	0V 12V PWM	Reference Lysstyrke

Tabellen nedenfor beskriver signalerne der findes i systemet, denne tabel anvendes når grænsefladerne skal designes og testes.

Signal navn	Funktionsbeskrivelse	Område	Output	Input	Kommentar
0V	Reference til analoge spændinger	N/A	GND GND GND GND	V-2 C-2 I-4, I-5 M-4, M-5, M-12, M-13	Stel
USB	Seriel kommunikation		USB	A-USB	
5V	Forsyningsspænding	4.9-5.1V	A-5V A-5V	I-7 L-1	5V fra USB
12V	Strømforsyningsspænding	11.9-12.1V	12V PS 12V PS	A-Vin M-1, M-8, M-9, M16	
PWM	Forsyningsspænding	0V eller 5V			Digital
A-Signal(mV)	Registrering af fysiske værdier	0-5mV	L-3 L-4	I1 I-2	Analog fra vægtcelle Analog fra vægtcelle
A-Signal(V)	Registrering af fysiske værdier	0-5mV	I-6	A-A0	Analog fra amplifier
D-Signal(V)	Styring af ventil	4.9-5.1V	A-D8	M-10	
0V or 12V PWM	Forsyning til pumpe	0 eller 12V	M-3 M-7	P-2 P-1	
12VDC	Forsyning til ventil	11.9-12.1V	M-11	V-1	
12V PWM	Forsyning til lysdioder	4.9-5.1V	A-D11	M-15	
5V PWM	Styring af lysdioder	4.9-5.1V	A-D11	M-15	
0V or 5V PWM	Styring af pumpe	0 eller 12V	A-D9 A-D10	M-2 M-6	Hastighed og retning

### 3.3.5 Kamera

Kameraet[Farnell [b]] skal detektere de langerhanske øer i systemet, hvilket gør at det er en elementær del af projektet.

**Specifikationer for kameraet:**

Specifikation	Værdi
<b>Billedes opløsning:</b>	2M pixel
<b>Fokus område:</b>	0-40mm
<b>forstørrelse:</b>	25X-400X (Manuelt)
<b>Frame rate:</b>	30f/s
<b>Interface:</b>	USB 2.0
<b>Forsyning:</b>	DC 5V ( <i>fra USB port</i> )
<b>Dimension:</b>	L:112 mm x B:33 mm

Da Langerhanske øer har en størrelse på 100 µm til 300 µm, skal kameraet have en opløsning så øerne kan detekteres. Opløsningen på kameraet er valg ud fra:

$$\text{Opløsning} = \frac{\text{Afstand}}{\text{Objektstørrelse}} = \frac{10 \text{ cm}}{100 \mu\text{m}} = 1Mp \quad (3.1)$$

[baslerweb,s.5] afstanden fra objektet er vurderet ud fra, hvor langt kameraet er i den absolut maksimale afstand fra øerne. Dertil skal der kunne ses flere detaljer end ned til 100 µm, for at kunne detektere de langerhanske øer. Derfor er der valgt et kamera med en opløsning på 2M pixels. Da de Langerhanske øer er lysere og har et andet fluorescens niveau end resten af vævet vil et kamera uden farver også være anvendeligt. Da systemet skal kunne sortere 30 langerhanske øer i minuttet, er det vurderet at en standard frame rate vil være passende. Til projektet er der taget et valg mellem 3 kamera, det valgte er listet oven over og de 2 fra valgte er i tabellen her under.

<b>Kamera type</b>	USB Microscope Digital-Mikroskop Endoskop[Ebay [c]]
<b>Billedes opløsning</b>	2M pixel
<b>Fokus område:</b>	n/a
<b>forstørrelse:</b>	50X-800X (Manuelt)
<b>Frame rate:</b>	n/a
<b>Interface:</b>	USB (ukendt version)
<b>Forsyning:</b>	DC 5V ( <i>fra USB port</i> )
<b>Dimension:</b>	L:112 mm x B:33 mm
<b>Pris:</b>	148,33kr.

<b>Kamera type</b>	AVEN ZIPSCOPE[Farnell [a]]
<b>Billede opløsning</b>	2M pixel
<b>Fokus område:</b>	10mm-500mm (Manuelt)
<b>forstørrelse:</b>	10x-50x (Optisk), 200X (Digital)
<b>Frame rate:</b>	30f/s
<b>Interface:</b>	USB 2
<b>Forsyning:</b>	DC 5V ( <i>fra USB port</i> )
<b>Dimension:</b>	n/a
<b>Pris:</b>	1734,15kr.

Prisen for det indkøbte Kamera var 264,15kr., hvilket har været på et middel prisniveau ud fra de tre kamera. Valget af kameraet gør præg af prioritering for et ordenligt datablad og et stramt budget.

### 3.3.6 Styreenheden

Microcontrolleren skal være styrerenheden i systemet, det betyder at den skal styre ventil, kameralyset, pumpe og vægtcelle.

**Specifikationer for Microcontroller[Arduino]:**

Specifikation	Værdi
Type:	Arduino UNO (ATmega328P)
Forsyning:	5VDC (USB)
Clock speed:	16 MHz
Digitale I/O pins:	14 stk (6 stk med PWM)
Analoge inputs pins:	6 stk
Interface:	USB
I/O output strøm:	20 mA
Flash hukommelse:	32 KB

Arduino er en open source platform til fremstilling af prototype print, med en ATmega328P microcontroller. Arduino platformen er valgt da Matlab understøtter interaktion via en Support package. Boardet er brugt i et stort omfang omkring i verden. Derfor er det en platform der er nemt tilgængelig og forholdsvis prisvenlig, samt at der findes en stor mængde dokumentation omkring emnet. Da det er en open source platform, kan der købes forskellige versioner som ikke er originaler. Arduino UNO er passende til dette system, da det er en forholdsvis simpel opgave microcontrolleren skal håndtere. Da systemet bruger computerens hukommelse og det meste er koden her på. Antallet af digitale og analoge porte er passende med den nuværende mængde af komponenter der skal styres.

### 3.3.7 Motordriver

Motordriveren skal hjælpe microcontrolleren med, at styre pumpen og ventilen til systemet. Det skal den fordi Arduinoen ikke kan trække pumpen alene, derfor skal der en ekstern forsyningskilde på, som forsynes i gennem motordriveren. Motordriveren skal også forsyne ventilen og kameralyset.

**Specifikationer for Motordriver[bilag A.1.2]:**

Specifikation	Værdi
Type:	L293D
Forsyning:	4,5-36V
Output spænding	4,5-36V
Output strøm	600mA per kanel
Kanaler	4 stk
Interface	Strømforsyning og arduino

Motordriveren kan levere 600mA pr. kanal, hvilket er nok til at forsyne pumpe, ventil og kameralyset. Derudover gives der et PWM-signal til motordriveren, som gives videre til pumpe, ventil og kameralyset. Motordriveren har desuden også den fordel, at den er galvanisk adskilt fra Arduinoen. og derved ikke kan nedlægge Arduinoen mm.

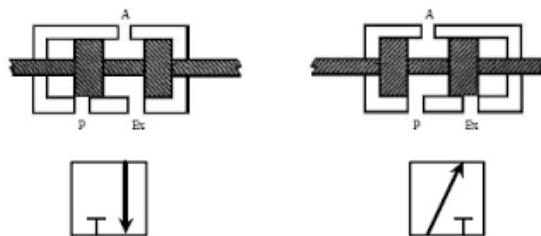
### 3.3.8 Ventil

Ventilens funktion er at sortere de langerhanske øer, fra resten af opløsningen. Det gør den ved at Arduinoen giver den besked om, at åbne og lukke for ventilen.

#### Specifikationer for Ventil[bilag A.1.4]:

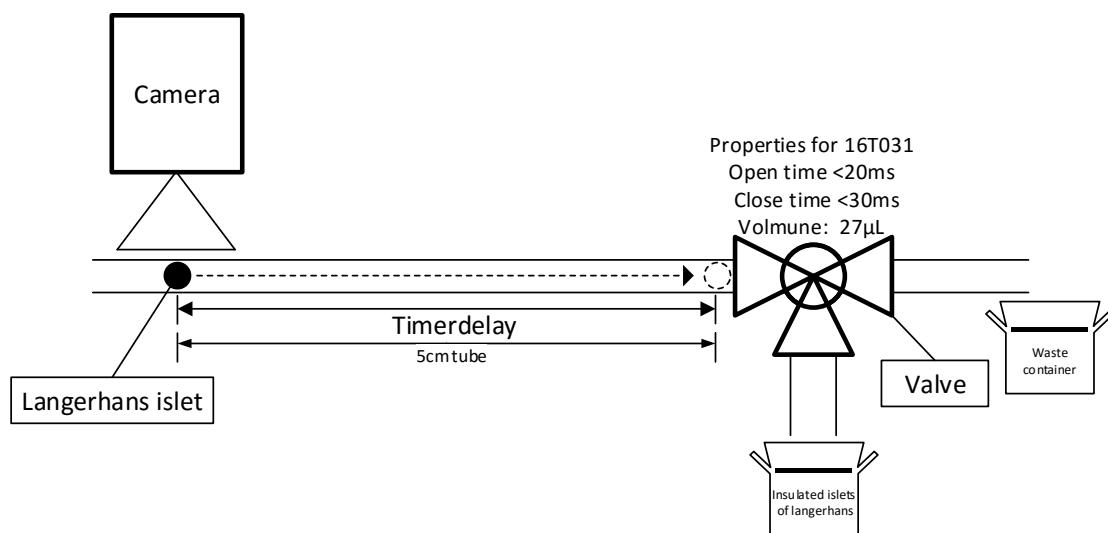
Specifikation	Værdi
Type:	Solenoid Electromagnet Valve
Forsyning:	12VDC, 0.1A
Porte	3 stk
studser	1mm
Åbningstid	<20ms
Lukningstid	<30ms
Interface	open/closed

Ventilen er en vigtig del af systemets hardware, da det er dens ansvar at sortere de detekterede øer fra resten af væsken. Det er svært at finde ventiler med 500 µm studser. Det kan godt lade sig gøre at få adaptere så større ventiler kan bruges, men sporbarheden omkring hvor den enkelte ø befinner sig forsvinder hvis kammerets volumen bliver for stort. Der er utrolig stor variation i kvalitet og pris på magnetventiler. Ventilen er et hardware, hvor der kan bruges meget tid og mange penge i projektet. Kravene til ventilen er, at den skal være 3 vejs med 1 tilgang og 2 udgange, yderligere skal studserne kunne tilpasses slange størrelsen, være til væske og have en lukke/åbne tid under 50 ms. For at kunne bruge ventilen er det vigtigt at forstå ventilens opbygning. En magnetventil virker på den måde, at der er en spole der danner et magnetfelt når der bliver påsat spænding og derved trækker et stempel til sig. Det får ventilen til at åbne ved hjælp af en membran. En trevejs magnetventil, som den brugte i dette projekt virker på samme måde. Forskellen ved en 3-vejs er at en skifter position er designet så der altid er en åben. Dette gør at opløsningen vil løbe ud i wastebeholderen når der ikke er en langerhanske ø, men kommer der en vil den skifte position, hvorefter de bliver frasorteret. Se figur 3.6 for en illustration af en 3 vejs magnetventil.



**Figur 3.6.** Magnetventil position

For at sorteringen af langerhanske øer er succesfuld, kræver det en beregning af hastigheden det tager for en langerhansk ø, at komme fra kameraet til ventilen. En estimation er beregnet igennem nedenstående formler. Kravet fra kunden hedder 30 øer i minuttet, derfor skal systemet som minimum have en flow hastighed på 11,25 ml/min. Se formel 3.6 for udregningen. På figur 3.7 kan der ses en illustration af problemstillingen



**Figur 3.7.** Illustration af ventil timerdelayet

$$\frac{\text{Opløsningsstørrelse}}{\text{Antal øer i opløsning}} = \frac{150 \text{ mL}}{400 \text{ øer}} * 30 \text{ øer/minut} = 11,25 \text{ mL/minut} \quad (3.2)$$

(jf. Søren Gregersen)

Tiden imellem kameraet og ventilen kan estimeres ud fra slangens volumen, imellem de to komponenter 3.3.

$$V = \pi * r^2 * h = \pi * \left(\frac{0,51 \text{ mm}}{2}\right)^2 * 50 \text{ mm} = 0,010\,214 \text{ mL} \quad (3.3)$$

Det vil sige at med et flow på 11,25 mL/minut, som er beregnet ud fra formlen 3.6 kan tiden fra kamera til ventil beregnes ved formel 3.4.

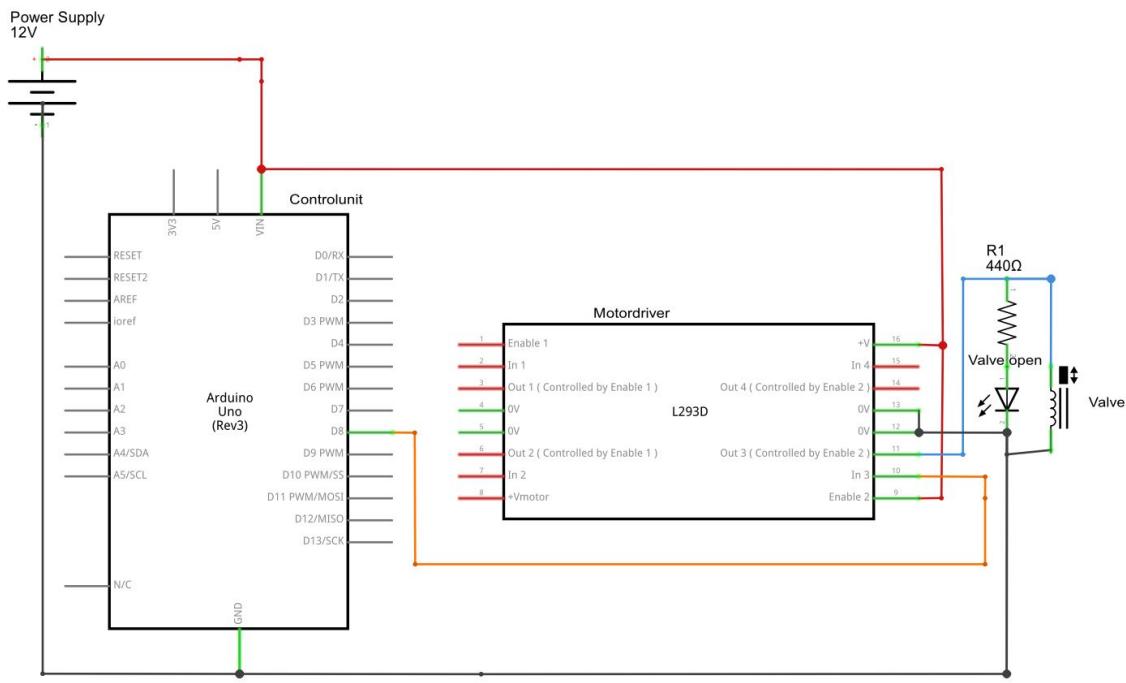
$$\frac{0,010\,214 \text{ mL}}{\frac{11,25 \text{ mL/minut}}{60 \text{ s}}} \rightarrow \frac{0,010\,214 \text{ mL}}{0,1875 \text{ mL/s}} = 0,054\,475 \text{ s} \quad (3.4)$$

Ifølge bilag ?? har ventilen et volumen på  $27 \mu\text{L}$ . For at beregne tiden det tager, at tömme ventilen bruges formel 3.5.

$$\text{Tid for udtømning af ventilen} = \frac{\text{ventil volume}}{\text{ml pr. sekund}} = \frac{27 \mu\text{L}}{0,1875 \text{ mL/s}} = 144 \text{ ms} \quad (3.5)$$

Beregninger til ventilen skal bruges i softwaren til at implementere timer til styring af ventilen. Åbnings- og lukningstid skal beregnes med i tiden til disse, se afsnit 4.3.4.6 for implementeringen af timerne og yderligere udregninger.

På figur 3.8 kan der ses et diagram over arduinoen, motordriveren, ventilen og en diode som indikator for at ventilen er åben. Ligesom ved motoren er det også her nødvendigt, at sætte en modstand foran lysdioden. Modstanden er af samme type som til kameralyset se 3.3.11 for beregninger af dem.



*Figur 3.8.* Kredsløbsdiagram for ventil

### 3.3.9 Pumpe

Pumpen skal skabe det nødvendige flow i væsken fra det ene punkt til det andet. Flowet skal være lavt nok til, at kameraet kan nå at detekterer en ø.

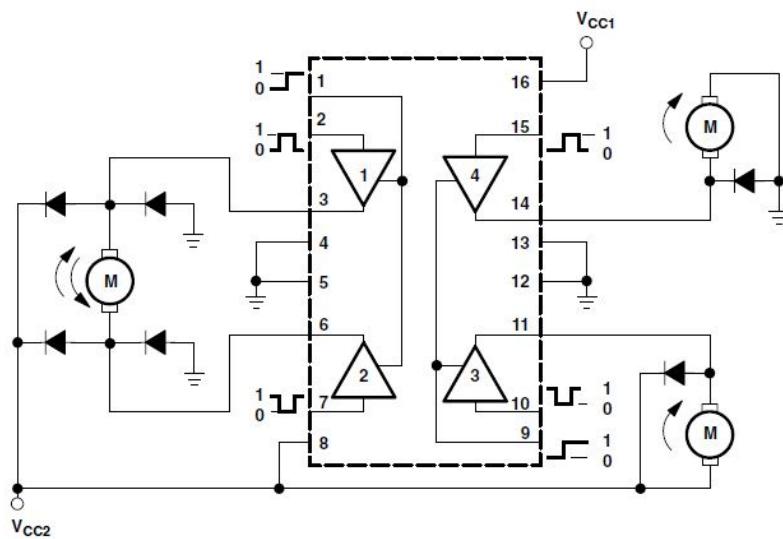
**Specifikationer for Pumpe[Ebay [b]]:**

Specifikation	Værdi
Type:	Peristaltisk pumpe
Forsyning:	12VDC, 0,3A
Hastighed	(2-17mL/min)
studser	ID:1mm OD:3,3mm

Det er et krav at pumpens flow hastighed er variabel, derfor kan flowet justeres. Der findes mange forskellige pumpetyper til formålet, herunder stempel pumper, peristaltiske pumper og vakuum pumper. Der er bestilt en peristaltisk pumpe, som virker ved at klemme på slangen og derved skabe et flow, det skal dog undersøges hvordan de langerhanske øer vil opfører sig ved denne pumpe. Undersøgelserne bør bestå af om de langerhanske øer tager skade ved, at pumpen klemmer på slangen.

#### 3.3.9.1 Motordriver

Til at drive pumpen er det nødvendigt med en motordriver, fordi arduinoen langt fra kan leverer den nødvendige strøm til pumpen. Kravet til motordriveren er, at den kan trække pumpen, samt stadig have mulighed for regulering af pumpens hastighed. Motordriveren består i dette projekt af en L293D, som enten kan drive 4 motorer den ene vej, 2 motorer i begge retninger eller i dette tilfælde en Peristaltisk pumpe, en magnetventil og kameralyset. se figur 3.9 for L293D interne kredsløb diagram.



**Figur 3.9.** Intern kredsløbsdiagram for L293D

Motordriverens opgaver er forsyne motoren, i sin simple form skal L293D tage signalet fra arduinoen, analysere det og give et tilsvarende signal fra strømforsyning til motoren.

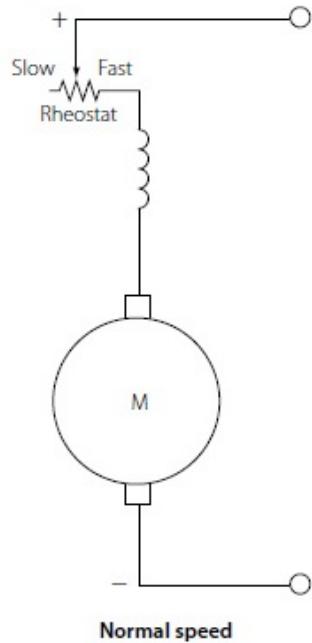
### 3.3.9.2 Hastighedscontrol af pumpe

Da det er svært at finde en pumpe der præcis opfylder kravet til hastigheden i dette system, grundet de 30 øer i minuttet(1.4.1.1)

$$\frac{\text{Opløsningsstørrelse}}{\text{Antal øer i opløsning}} = \frac{150}{400} * 30 = 11,25 mL/min \quad (3.6)$$

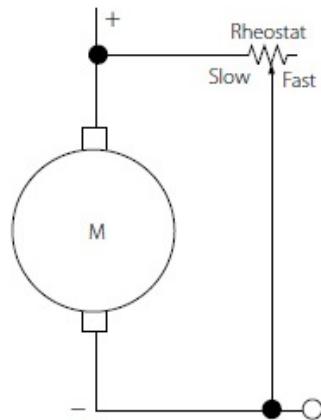
(jf. Søren Gregersen) Der er 3 primært metoder til, at styre hastigheden på en DC-motor [Hambley, 2011]s.810.

1. Indsætte modstand i kredsløbet før motoren er kobles til. Denne metode kan bruges på alle typer af motorer. En simpel måde at gøre det på er, at sætte en variabel modstand i serie med motoren for at styre hastigheden. Ulemperne ved dette kredsløb er, at forbruget vil være ens ved den højeste hastighed og den laveste. Fordi modstanden vil blot omsætte energien til varme, hvilket vil være spild af energi. Se figur 3.10, principippet er baseret på KVL.



**Figur 3.10.** Kredsløbsdiagram for Motor med variablemodstand

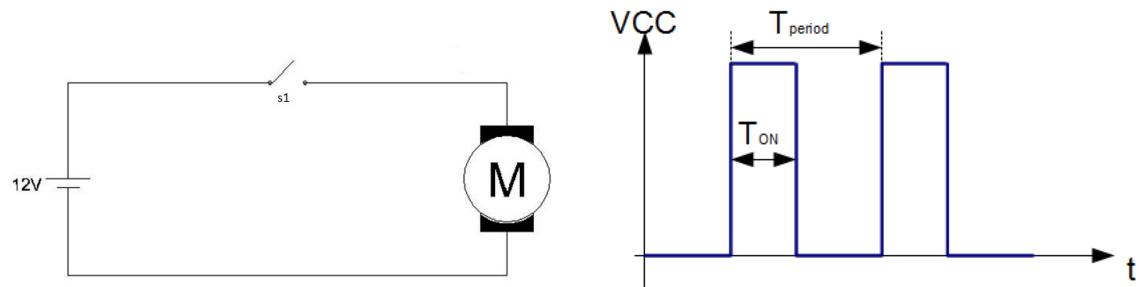
2. Variere strømmen der tilføres motoren, hvor spændingen holdes konstant. Denne metode minder om den overstående, men i stedet for serie kobles den variable modstand parallelt, som det kan ses på figur 3.11. Ulempen ved dette kredsløb er, at når motoren skal køre ved lav hastighed, er modstanden lille og derved tæt på en kortslutning derfor er det vigtigt, at kredsløbet er designet rigtigt. Princippet er baseret på KCL.



**Figur 3.11.** Kredsløbsdiagram for Motor med variable strøm

3. Variere spændingen der tilføres motoren periodevis, hvor strømmen holdes konstant. Konceptet i denne metode er at have en konstant forsyning, som slukkes og tændes for, sagt på en simpel måde. Dvs. at der sidder en kontakt i mellem strømforsyningen og motoren, der tændes og slukkes. Kredsløbet for dette kan stilles op som figur 3.12 og giver et signal som på figur 3.13

Åbningstiden  $T_{on}$  er der hvor kontakten er tændt,  $T_{period}$  er perioden som kontakten åbner



**Figur 3.12.** Kredsløbsdiagram for Motor med spændingskontakt

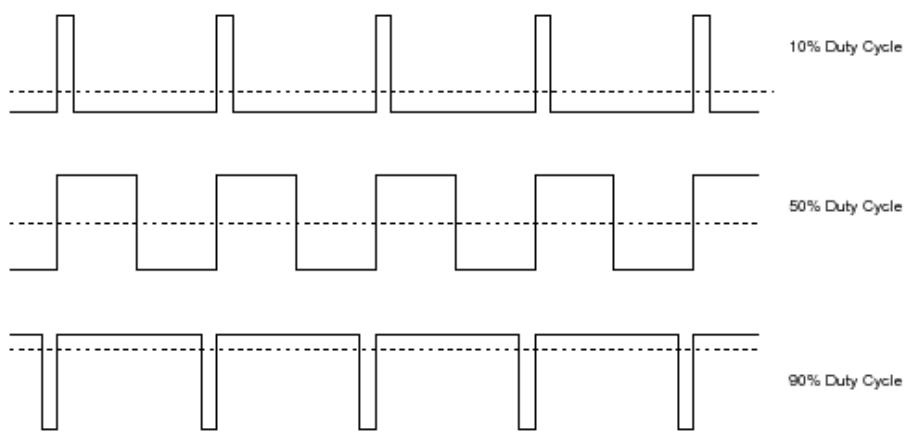
**Figur 3.13.** Motorsignal ved periodevis spænding

og lukker. Den gennemsnitlige spænding kan regnes ud ved 3.7

$$V_T = V_S \frac{T_{on}}{T_{period}} \quad (3.7)$$

Ud fra overstående formel (3.7) ses det, at gennemsnit spændingen og derved styres hastigheden af motoren ved  $T_{on}$ , hvor lang tid kontakten er tændt. Denne form for styring af en motor, kaldes pulse width modulation.

**3.3.9.2.1 Pulse width modulation** Pulse width modulation(PWM) er en meget brugt metode, som kan bruges til at styre hastigheden på en DC motor. Dette gøres ved at tænde og slukke for en digital udgang. Det er duty cyklussen der får hastigheden til at varierer, dvs. at det er tiden, hvor det firkantede signal er "højt" der bestemmer hastigheden. Se figur 3.14 for illustration af PWM duty cyklusser og gennemsnit spændinger. PWM styring kan bruges til blandt andet dæmning af dioder, generering af lydsignaler og styring af motorer.



**Figur 3.14.** PWM duty cyklusser med gennemsnits spændinger

Der er flere metoder, hvor dette kan implementeres vha. arduinoen. Den mest simple metode er `analogWrite(pin, dutyCycle);` hvor `dutyCycle` er en værdi mellem 0 og 255 og

*pin* er en af de digital udgange med PWM. Dette er simpelt, men der er ingen control over frekvens mm.

Manuel PWM implementering er en anden metode, der kan implementeres således:

```

1 void setup()
2 {
3     pinMode(13, OUTPUT);
4 }
5
6 void loop()
7 {
8     digitalWrite(13, HIGH);
9     delayMicroseconds(100); // Approximately 10% duty cycle @ 1KHz
10    digitalWrite(13, LOW);
11    delayMicroseconds(1000 - 100);
12 }
```

Her sættes pin 13 til *høj*(5V ved arduino), hvor efter der ventes 100 mikro sekunder. Derefter sættes pin 13 *lav*(0V) og der ventes 900 mikro sekunder. Ved denne metode kan alle digitale udgange bruges og ikke kun dem med PWM. Dog er processoren brugt hele tiden da den også bruges når der ventes. Det vil sige at den ikke kan bruges til andet samtidigt, hvilket ikke kan bruges i projektet. Microcontrolleren der sidder på arduinoen indeholder 3 timer, som kan bruges til PWM generering. Måden dette virker på er, at timeren går fra 0 til 255 hvis *Timer* = 0 sættes outputtet højt, hvor den tæller til enten 255 og slukker hvilket vil skabe en duty cyklus på 100%. Derfor kan der defineres en værdi hvor der slukkes mellem 0 og 255. Ydermere findes der scalar (1, 8, 64, 256 eller 1024 som divideres med arduinens clock frekvens. Alt dette konfigureres ved bits i registre (TCCRnA og TCCRnB) på microcontrolleren. Et eksempel på dette kunne være:

```

1 void setup()
2 {
3     pinMode(3, OUTPUT);
4     pinMode(11, OUTPUT);
5     TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
6     TCCR2B = _BV(CS22);
7     OCR2A = 180;
8     OCR2B = 50;
```

Ved overstående kode vil der være en duty cyklus på output A da være

$$\frac{180 + 1}{256} * 100 = 70,7\%$$

og en PWM frekvens på

$$\frac{16MHz}{64/256} = 976.6Hz$$

. Output B vil have en duty cyklus på

$$\frac{50 + 1}{256} * 100 = 19.9\%$$

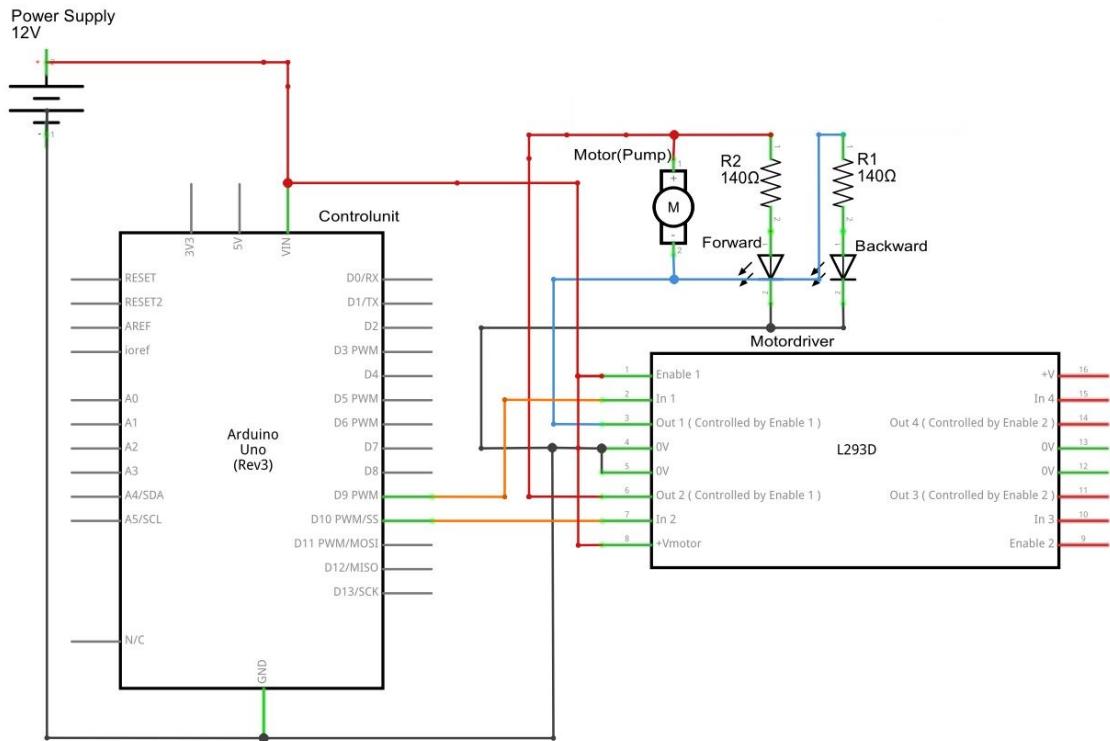
og den samme frekvens. Den sidst nævnte metode, er den metode hvor der er mest kontrol over PWM signalet. Derfor bruges denne i dette projekt.

For at lysdioderne får deres ønskede spænding, er det nødvendigt med modstande foran dem. Modstandene kan beregnes på følgende måde.

$$R = \frac{V_{source} - V_{lysdiode}}{I_{lysdiode}} = \frac{6 - 3.2}{20mA} = 140\Omega \quad (3.8)$$

Dette giver en effekt afsætning i modstandene på ( $V_{source} - V_{lysdiode} * I_{lysdiode} = 2,8 * 20mA = 0,056W$ ) hvilket giver grund for at 0.33W modstande er passende.

På figur 3.15 kan ledningsdiagrammet til motordriver med motor og frem og tilbage lysdioder observeres. Med opsætningen kan retningen på motoren bestemmes ved, hvilken en af udgangene der sættes til lav(0V). Hvis den modsatte udgang konfigureres til PWM. Gøres det modsat kører motoren den modsatte vej.



**Figur 3.15.** Kredsløbsdiagram for motor og motordriver

### 3.3.10 Vægtcelle

Vægtcellen skal bruges til at kontrollere om, der er væske i celleopløsningsbeholderen. Når celleopløsningsbeholderen løber tør skal systemet stoppe.

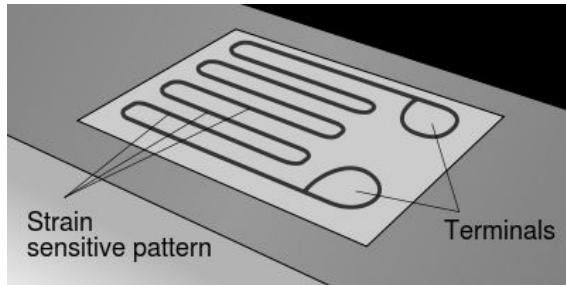
**Specifikationer for Vægtcelle[Ebay [a]]:**

Specifikation	Værdi
<b>Max belastning:</b>	1 kg
<b>Anbefalet arbejdsspænding</b>	3-12V
<b>Output</b>	1.0mV/V±0.15mV/V

Den indkøbte vægtcelle kan veje op til 1 kg, hvilket dækker vægten for celleopløsningsbeholderen på 250ml + beholderens vægt.

For at kunne udnytte funktionen af en vægtcelle, skal opbygningen af denne forstås. En

vægtcelle mäter, hvor meget vægt den udsættes for, vægtcellen der er brugt i projektet består af strain gages koblet i en wheatstone bro. En strain gage bruges til at måle fysisk stræk eller kompression. Den er simpel i sin opbygning ved, at bestå af en meget tynd elektrisk ledende tråd, som er ført frem og tilbage på et elastisk folie, hvilket er illustreret på 3.16<sup>1</sup>.



**Figur 3.16.** illustration af strain gages

En strain gages modstand stiger ved stræk og falder ved kompression, dette kan beskrives ud fra formlen 3.9[Webster, 2010]s.47

$$R = \frac{\rho * L}{A} \quad (3.9)$$

Hvor R=modstand,  $\rho$ =modstand per meter, L=længden og A=tværsnitsarealet. Der er mere til formlen end dette, bla. materiales egenskaber og temperatur. På vægtcellen er der fire strain gages, som er placeret på vægtcellen dette er vist på figur 3.17. Strain gages R1 og R4 bliver strukket, hvor R2 og R3 på undersiden bliver skubbet sammen.

Måden de er forbundet er vist på figur 3.18, denne metode at koble modstande på kaldes en wheatstone bro[Hambley, 2011]s.122. mere specifikt for denne er også kendt som et full-bridge kredsløb[Perlovsky, 2009]s.76. Forholdet mellem input spændingen og output spændingen kan beskrives, som vist ved formlen3.10

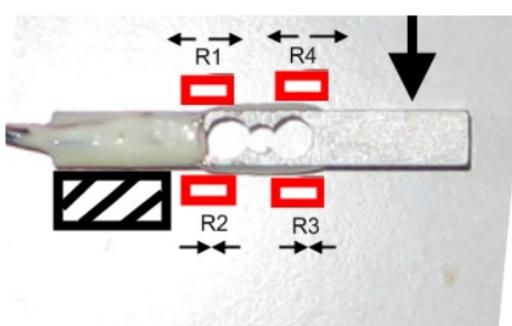
$$V_{out} = GF * V_{in} * \varepsilon \quad (3.10)$$

Hvor  $V_{out}$ =udgangssignalet,  $V_{in}$ =indgangsspændingen , GF=gages factor som er materialelets egenskab og  $\varepsilon$ =strukket der er tilført til vægtcellen.

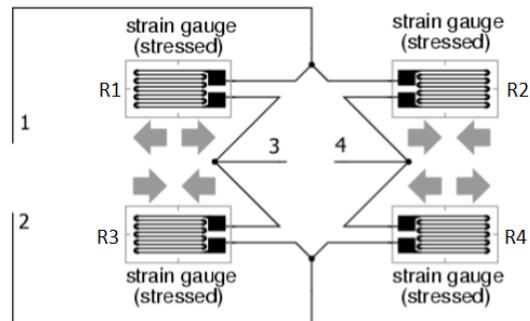
Ud fra databladet 3.3.10 ses det, at output spændingen er 1.0mV pr volt på indgangsspændingen. Dette er pga. det meget lille strain der tilføres til emnet, hvilket også er med til at strain gagesene ikke går i stykker. Da Arduinoens analog til digital konverterer er 10bit, hvilket vil sige  $2^{10} = 1024$  betyder det at konverteren har 1024 trin fra 0 til 1023. Konverterens arbejdsspænding går fra 0 V til 5 V.

$$\text{Spænding per trin} = \frac{\text{Maksimale spænding}}{\text{Antal trin}} = \frac{5V}{1024 \text{ trin}} = 0,0049V = 4,9mV \quad (3.11)$$

<sup>1</sup>kilde: [https://en.wikipedia.org/wiki/Strain\\_gauge](https://en.wikipedia.org/wiki/Strain_gauge)



**Figur 3.17.** Illustration af strain gages i loadcell på virkning



**Figur 3.18.** Illustration af koblingen strain gages i loadcell

I 3.11 viser det sig, at den mindste værdi ADC kan måle er  $4,9\text{mV}$ , dette medfører at arduinoen maksimalt vil måle et step ved  $1\text{kg}$  på vægtcellen, da  $1\text{mV} * 5\text{V} = 5\text{mV}$ . Dette giver to valgmuligheder

1. Anskaffe en bedre ADC, bestående af flere bits
2. Hæve udgangsspændingen fra vægtcellen

I dette tilfælde vælges punkt 2, at hæve udgangsspændingen fra vægtcellen.

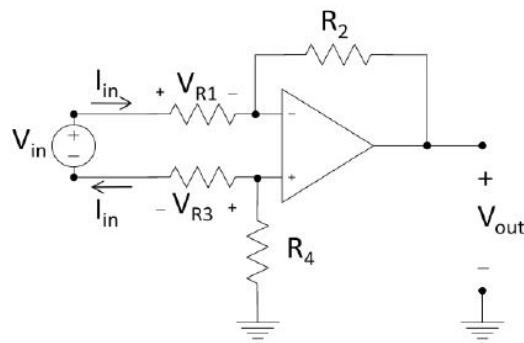
### 3.3.10.1 Forstærkning af signal

Til dette formål bruges en operationsforstærker, mere specifikt en differens operationsforstærker. En operationsforstærker består i sin mest simple funktion at forstærke et signal, men da udgangsspændingen fra vægtcellen er lav ønskes der følgende egenskaber:

1. En høj indgangsimpedans
2. Differentielt input med et single ended output
3. En høj undertrykkelse af støj
4. En simple forstærkning

Punkt 1, en høj indgangsimpedans ( $R_{in} > 10-100\text{M}\Omega$ ) sikrer at forstærkeren ikke belaster måle objektet. Det ønskes ikke at signalet undertrykkes ved, at forstærkeren forbruger strømmen fra signalet for. Derfor ønskes der adgang til hele signalet.

Punkt 2, Differentielt input med et single ended output. Det er et ønske som kan begrundes ved, at vægtcellen leverer et differentielt output og at ADC'en kun kan modtage et single ended input. Derfor er det lettere at arbejde med et single ended output. Indgangsmodstanden er bestemt ved modstanden mellem de to indgangsterminaler, som kan illustreres ved figur 3.12 og beregnes ved formlen 3.19



**Figur 3.19.** illustration af differensforstærkerens indgangsmodstand

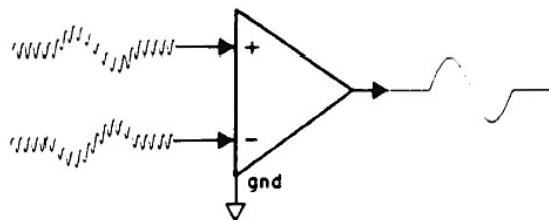
$$R_{in} = V_{in}/I_{in} \quad (3.12)$$

I den ideelle verden antages det, at spændingsforskellen i mellem indgangsterminalerne er nul derfor kan der skrives vha. Kirchoffs spændingslov skrives i 3.13

$$V_{R3} - V_{in} + V_{R1} = 0 \Rightarrow V_{in} = V_{R3} + V_{R1} = I_{in}(R1 + R3) \Rightarrow R_{in} = \frac{V_{in}}{I_{in}} = \frac{R1 + R3}{1} \quad (3.13)$$

I formel 3.13 vises det sig at indgangsmodstanden, er beskrevet ved modstandene i det omkring-liggende kredsløb. Dette giver anledning til at vælge så høje modstande som mulige, men store modstande øger risikoen for støj i kredsløbet. Det betyder at ønsket om en høj indgangsmodstand ikke kan opfyldes, med en simpel differensforstærker.

Punkt 3, en høj undertrykkelse af støj. Støj kan skyldes rigtig mange ting, en typisk støjkomponent er 50 Hz brum. 50 Hz brum fremkommer ofte, da støjen skyldes de omkring liggende EL installationer, hvor der foregår en elektromagnetisk kobling. Når der benyttes en differensforstærker, kan common mode støjen undertrykkes. Common mode støj er indstrålet støj der kommer på begge ledninger til differensforstærkeren. Det er differensforstærkeren god til at frasortere, fordi den trækker de to input fra hinanden. Det vil sige at den vil udligne støjen på de to ledninger, ved at subtrahere den samme værdi fra hinanden, hvilket vil give nul. En illustration af dette kan ses på figur 3.20[Thomas, 2012].

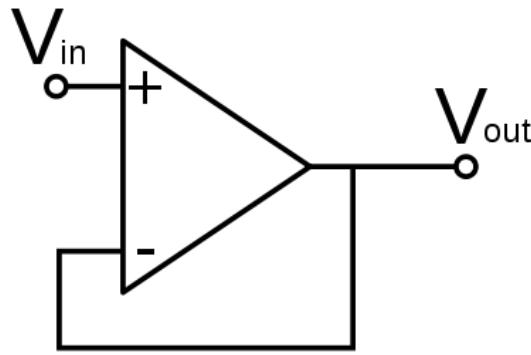


**Figur 3.20.** Illustration af differensforstærkerens common mode støj undertrykkelse

Punkt 4, en simpel forstærkning. Med det overstående kredsløb, kan en simpel forstærkning ikke opnås. Det kan det ikke da det kræver at  $R_2$  divideret med  $R_1$  er lig med  $R_4$  divideret

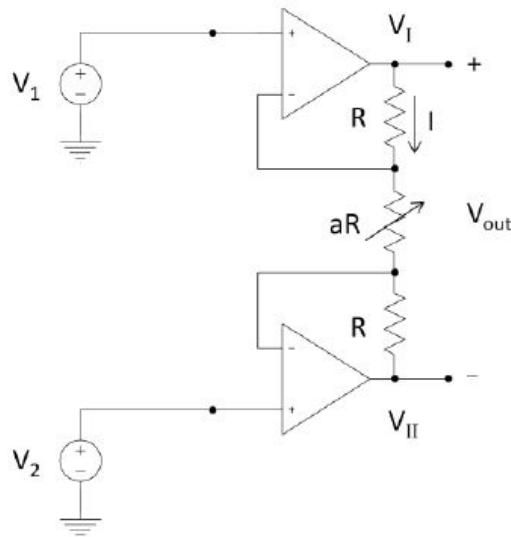
med  $R_3$ . Da modstandes præcision svinger indenfor tolerancer, vil  $\frac{R_2}{R_1}$  aldrig være lig med  $\frac{R_4}{R_3}$

For at opfylde de ønskede egenskaber skal differensforstærkeren modificeres. For at sikre en høj indgangsmodstand kan en spændingsfølger benyttes, som har til formål at forstærke en-til-en. Det vil sige at den i teorien har samme udgangsspænding som indgangsspændingen. se figur 3.21 for en illustration, denne sættes på begge outputs fra vægtcellen.



**Figur 3.21.** illustration af spændingsfølger

Med en spændingsfølger opnås der en høj indgangsmodstand, men signalet er stadig differentielt og uden forstærkning. Derfor indsættes der 3 modstande som vist i 3.22[Thomas, 2012]s.200

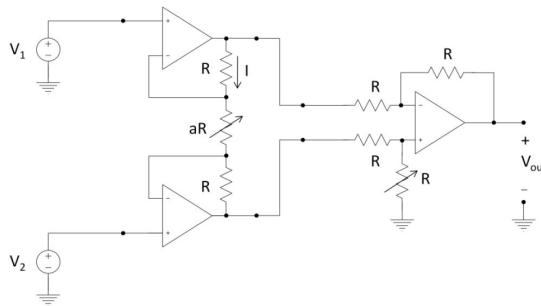


**Figur 3.22.** illustration af spændingsfølger med gain

Ved brug af KVL og KCL kan forstærkningen skrives som i 3.14, hvor  $A_d$  er forstærkningen og  $R_a$  er modstanden i midten. Dette gør forstærkningen simpel fordi der kun skal ændres en modstand.

$$A_d = 1 + \frac{R + R}{R_a} \quad (3.14)$$

Til at opfylde ønskerne om undertrykkelse af støj og et single ended output kan differensforstærkeren sættes efter spændingsfølgerne med forstærker trinet. Dermed kommer kredsløbet til at se ud som vist i figur 3.23

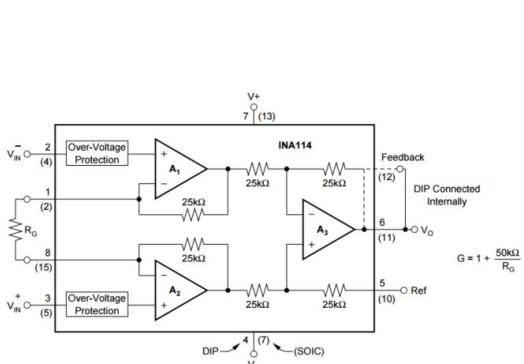


**Figur 3.23.** illustration af spændingsfølger med gain og differensforstærker

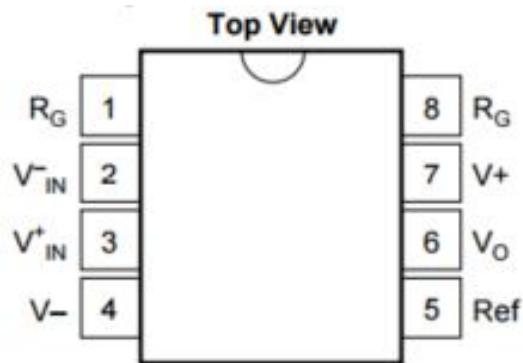
Ligning 3.15 viser overføringsfunktionen for diagrammet i 3.23.

$$V_{out} = (V_2 - V_1) * \left(1 + \frac{R + R}{R_a}\right) \quad (3.15)$$

Kredsløbet på figur 3.23 kaldes en instrumentationsforstærker, hvilket der i projektet er indkøbt da den lever op til de ønskede egenskaber og er mere præcis end at lave kredsløbet selv. Den indkøbte instrumentationsforstærker hedder INA114, som har kredsløbet- og pinkonfiguration vist på figur 3.24 og 3.25A.1.1



**Figur 3.24.** INA114 diagram



**Figur 3.25.** pin konfiguration af INA114 8pin

I databladet A.1.1 til INA114 ses det, at den har en CMRR på 115dB ved et gain på 1000 og en indgangsmodstand på  $10G\Omega$ . Forstærkningen kan regnes ud fra formlen i databladet 3.16

$$G = \left(1 + \frac{50K\Omega}{R_G}\right) \quad (3.16)$$

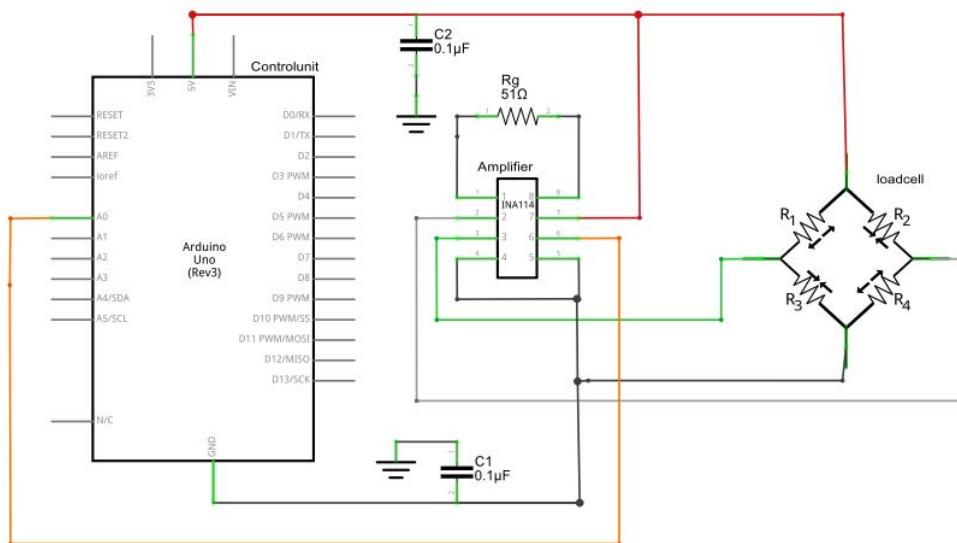
I dette projekt skal der bruges et gain på  $\frac{4,9V}{5mV} = 980$ , 4,9V for ikke at komme i mætning på arduinoens ADC og 5mV da det er den maksimale spænding vægtcellen kan give, ved 5V forsyning.

$$R_G = \frac{50K\Omega}{980 - 1} = 51\Omega \quad (3.17)$$

Med et gain på 980 giver en  $NY_{Maksimalespnding} = 980 * 5mV = 4,9V \pm 0,147V$ , dvs at der nu er en opløsning på

$$\frac{1000g}{trin} = \frac{1000g}{1024} = 0,977g/trin \Rightarrow 0,977 * \frac{1024}{5V} = 200g/V \pm 30g \quad (3.18)$$

Kredsløbet for INA114 og vægtcellen til arduinoen kan ses på figur 3.26



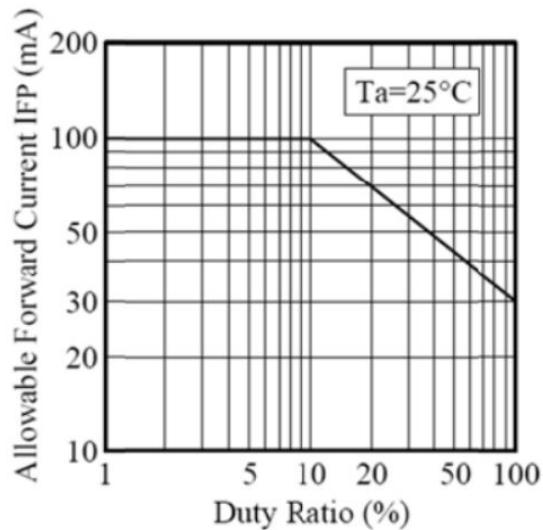
**Figur 3.26.** Diagram for arduino, INA114 og vægtcelle

### 3.3.11 Kameralys

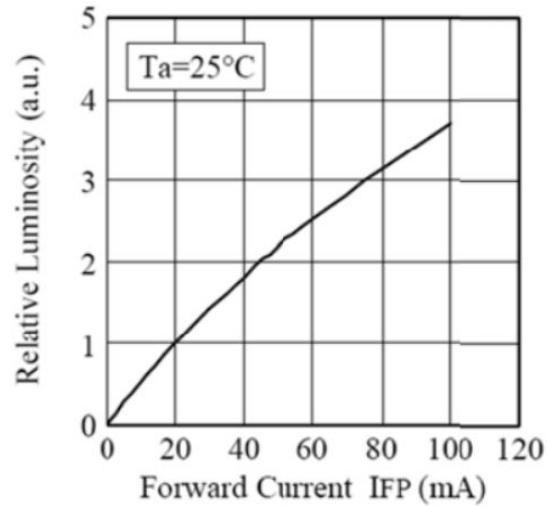
Kameralyset skal styre lyset til kameraet, så operatøren selv kan styre hvor meget lys der skal være i kamerakammeret. Dette gøres for at få den bedste mulighed for, at detekter de langerhanske øer. **Specifikationer for Lysdiode[A.1.3]:**

Specifikation	Værdi
Type:	L5-w55N-BVW
Anbefalet arbejdsspænding	3.2-3.5V
Max strøm	100mA v. 10% duty cyklus
Farve	6500K
Lysstyrke	22000-33000 mcd
Spredningsvinkel	15°

Lysdioden har et tilladt strøm til Duty cyklus forhold, som kan ses på figur 3.27 yderligere har lysdioden også et lysintensitet til strøm forhold, dette kan ses på figur 3.28. Da sandsynligheden for, at kameralyset er tændt i længere periode bruges 20mA. Lysdioden kan tåle 30mA, men pga. den lille kasse lyset skal sidde i, er det vurderet bedre at ligge lidt under.



**Figur 3.27.** Tilladte strøm og duty cyklus forhold



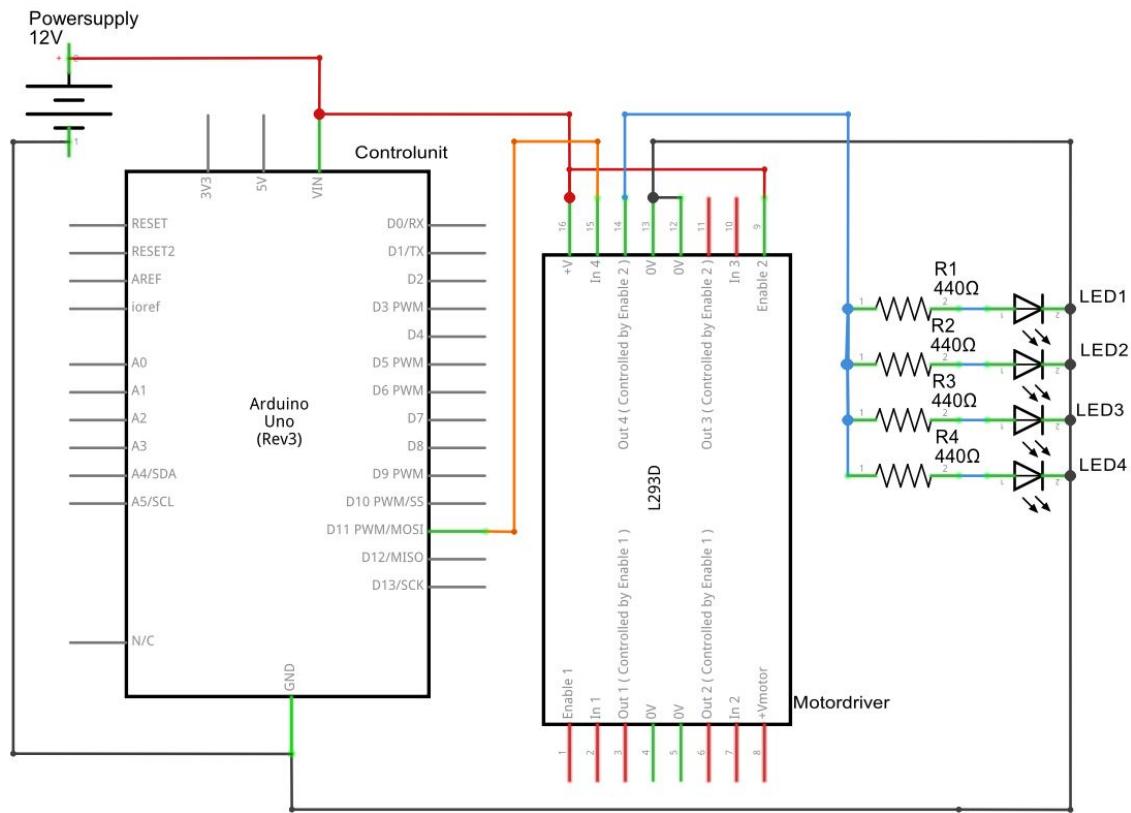
**Figur 3.28.** Lysintensitet og strøm forhold

Kameralyset skal kunne variere, da operatøren skal kunne indstille lyset for den bedst mulige opsætning. Det gøres for at kameraet har de bedste muligheder for at detekterer de langerhanske øer. Der skal sættes 4 lysdioder i en kasse, hvor kun kameralyset lyser kassen op. Kameraet skal detektere de langerhanske øer der løber i slangen inde i kassen. Lyset implementeres for at styre lysniveauet på billederne. Da der ikke kan trækkes strøm nok ud af arduinoen, kræves det en anden forsyning til lysdioderne. Dertil udnyttes den sidste udgang i motordriveren, men da denne forsynings med 12V og lysdioder kun tåler 3.2V skal spændingen nedjusteres. Den lettest måde at gøre dette på er, at sætte lysdioderne i serie. Hvis en lysdiode bliver defekt, skabes der ingen lys til kameraet. Derfor er der valgt en løsning med modstande siddende parallelt foran hver lysdiode. Formodstandenes værdi er beregnet ved 3.19.

$$R = \frac{V_{source} - V_{lysdiode}}{I_{lysdiode}} = \frac{12 - 3.2}{20mA} = 440\Omega \quad (3.19)$$

Dette giver en effekt afsætning i modstandene på ( $V_{source} - V_{lysdiode} * I_{lysdiode} = 8,8 * 20mA = 0,176W$ ) hvilket giver grund for, at 0.33W modstande er passende.

På figur 3.29 vises diagrammet for Kameralyset sammen med motordriveren.



Figur 3.29. Kredsløbsdiagram for kameralys

### 3.3.12 Beholdere

Systemet består af tre beholdere der hver især har sin egen funktion. Den første kaldes *celleopløsningsbeholderen*, som skal indeholde den usorteret opløsning med de langerhanske øer. Beholder nummer to kaldes *beholderen med de langerhanske øer*, som er den beholder hvor de isolerede langerhanske øer samles i. Den tredje beholder er *waste beholderen*, den skal have resten af opløsningen som ikke består af langerhanske øer.

Specifikation	Værdi
Type:	Opløsningsbeholder(Mikrolab 33184)
Størrelse:	250 ml

Størrelsese kravene til beholderne er, at opløsningsbeholderen skal mindst være 250ml, da det er den største mængde opløsning der vil blive brugt(jf. *Søren Gregersen*). Wastebeholderen skal således være dobbelt så stor, så der kan gennemløbes to sorterings cyklusser uden at skulle tømme beholderen i mellem. Den isolerede beholder, skal kunne rumme mængden af de isolerede øer. Da projektet som tidligere nævnt er et ”proof of concept” er den eneste beholder der er hentet informationer om opløsningsbeholderen. Den bør være støvtæt uden, at være lufttæt da der ellers vil dannes et vakuum i beholderen. Derudover vil det være en fordel hvis den er forholdsvis robust, så den kan køles ned osv. på et senere tidspunkt. ML 33184 fås med et skruelåg med forskruninger, der kan tilpasses de indkøbte slanger. Yderligere fås den med et luftfilter, så der ikke dannes vakuum i beholderen.

### 3.3.13 Slanger

Slangerne anvendes til at føre opløsningen med de langerhanske øer, i gennem systemet bl.a. forbi kameraet.

#### Specifikationer for slangerne:(flere)

Specifikation	Værdi
Materiale:	Teflon, silikone og PVC
Ydre diameter:	1-1,5mm
Indre diameter:	500 µm-700 µm

Da de største langerhanske øer, som systemet skal håndtere er 300 µm i diameter er det valgt, at slangerne skal have en indre diameter på 500 µm. Da det er optimalt at kameraet kan se direkte igennem slangen, er der valgt et gennemsigtigt materiale. Yderdiameteren er valgt for at, pumpen ikke skal ødelægge celler og slange, hvilket også har været med til at bestemme materialet af slangerne.

### 3.3.14 Glasrør

For at sikre, at kameraet er i stand til at se de langerhanske øer er der valgt et glasrør.

**Specifikationer for Glasrøret:**

Specifikation	Værdi
Materiale:	glas
Ydre diameter:	1.3mm
Indre diameter:	1mm

Glasrøret er tiltænkt kun til at sidde, hvor kameraet skal detektere de langerhanske øer. For at være sikker på at kameraet kan se dem. Glasrøret der er valgt er rundt. Krumningen i røret kan muligvis ændre billedet fra kameraet. Et rektangel glasrør har været overvejet, men risikoen for at de langerhanske øer kan overhale hinanden inden i glasrøret er tilstede. Derfor er et rundt glasrør valgt.

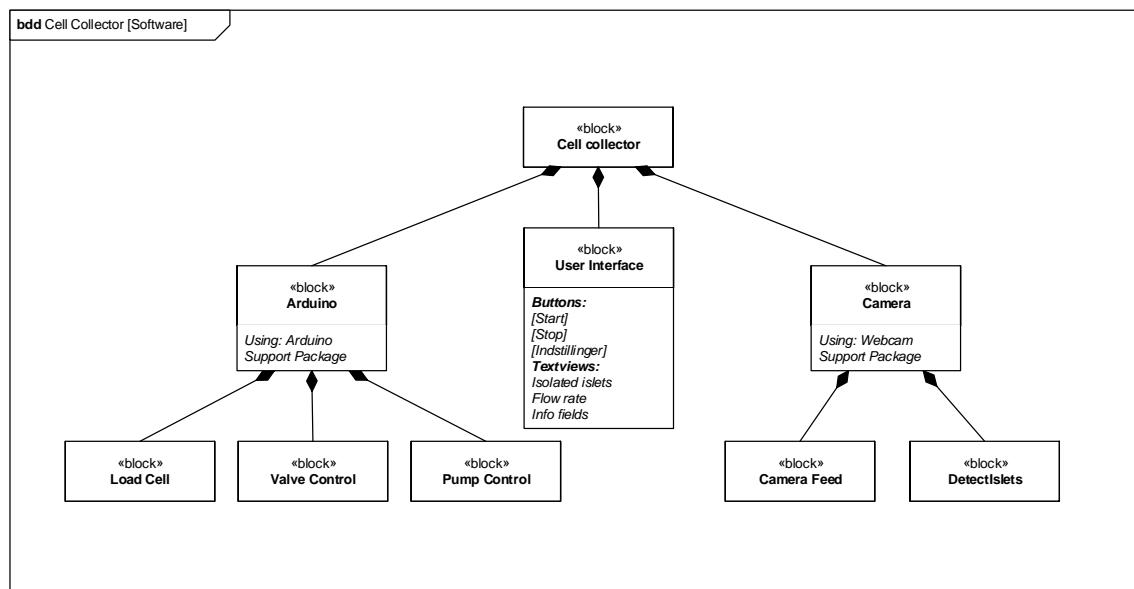
## 3.4 Software

Softwareen til *The Cell Collector* udvikles i Matlab. Programmet udvikles af modulær kode, der afgrænser de enkelte funktionaliteter. Ved objektorienteret programmering beskrives koden typisk vha. klassediagrammer og 3-lags modellen, hvor de enkelte klassers ansvar og grænseflader tydeligt er defineret. Da Matlab kode er opdelt i funktioner fremfor for klasser er modellen ikke velegnet. I stedet beskrives softwaren vha. blokdiagrammer, hvor funktionernes opbygning og relationer med hinanden er vist.

Blokdiagrammet (figur: 3.30) viser hvordan programmet er opdelt i blokke, som afgrænser de enkelte funktionaliteter. Blokkene i det nederste lag skal ses som ækvivalente til funktioner i Matlab. De overordnede blokke i programmet er:

- Arduino
- Kamera
- User Interface

Disse blokke er nærmere beskrevet under hver deres afsnit.

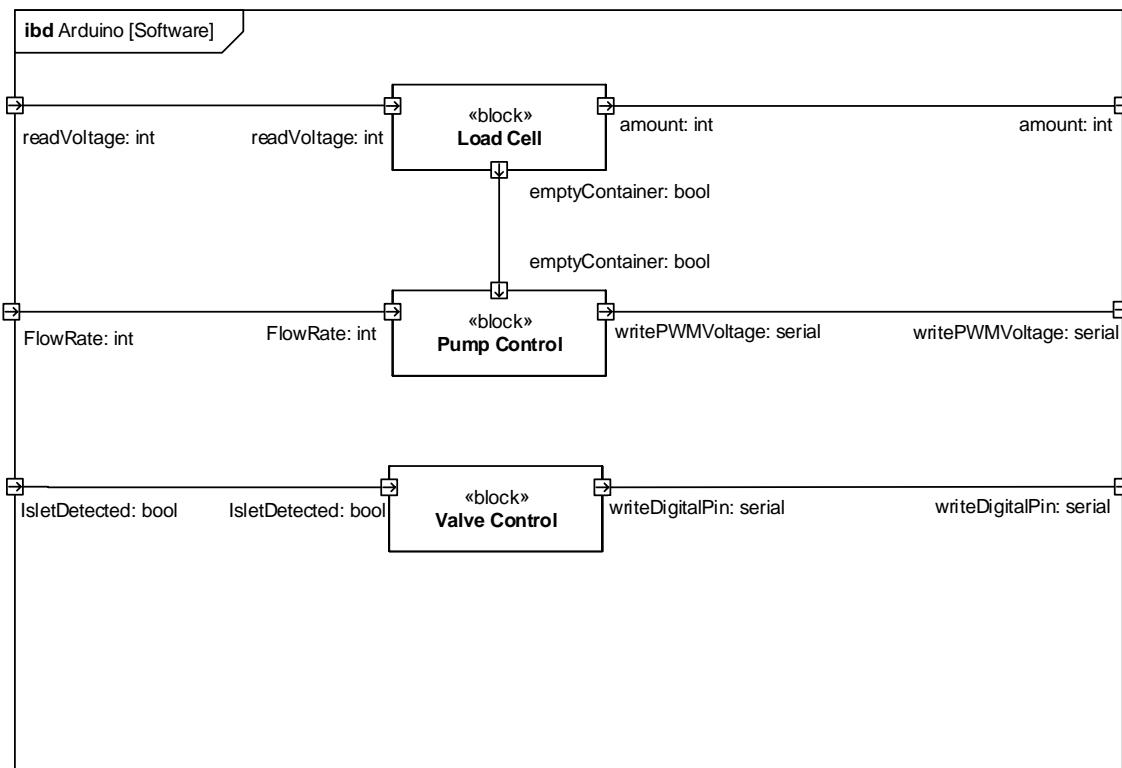


**Figur 3.30.** BDD - Cell Collector [Software]

### 3.4.1 Arduino

Denne bloks formål er, at håndtere al funktionalitet til styring af Arduinoen. Til styring af Arduinoen anvendes Arduino Support Package, som frit kan hentes hos Mathworks. Den indeholder basale funktioner til bl.a. opsamling af analoge signaler, understøttelse af digitalt og PWM output og styring af DC motorer. Support biblioteket indeholder de funktioner der skal til for at styre systemets hardware. For at initialisere Arduinoen og hardwaren implementeres en funktion, som opsætter Arduinoen med de inputs og outputs der er specificeret under hardware afsnittet (figur: 3.5, s. 37). Denne funktion er kaldt initArduino. Når brugeren klikker "Stop" skal systemet lukke ned som specificeret i Use Case 3: Stop sorteringscyklus (s. 6). Til dette implementeres en funktion kaldet stopArduino.

Arduino blokken er yderligere opdelt i 3 underkategorier, som vist i figur 3.30. I det interne blok diagram (figur: 3.31) ses underblokkenes relationer med hinanden. Disse blokke er nærmere beskrevet herunder.



*Figur 3.31.* IBD - Arduino [Software]

#### 3.4.1.1 LoadCell

Denne funktion anvendes til, at få feedback fra loadcellen. Den læser det analoge input fra Arduinoen og sammenligner den med grænseværdien for hvornår celleopløsningsbeholderen er tom. Funktionens output er en boolean, som enten er true eller false alt efter om celleopløsningsbeholderen er tom. Herudover konverteres det analoge input til indholdet af beholderen, udtrykt i mL.

#### 3.4.1.2 Pump Control

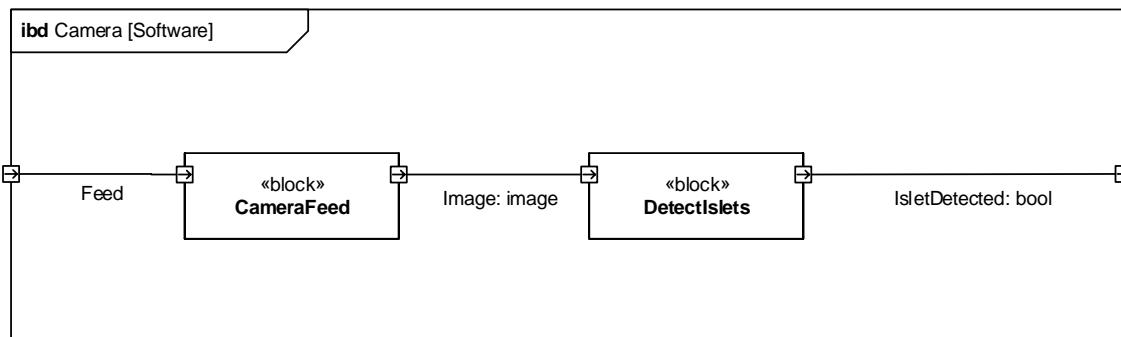
I denne funktion implementeres alt funktionalitet til styring af pumpen. Funktionen har en integer værdi som input, der specificerer flow hastigheden. Output af funktionen er et PWM moduleret signal til, at justere hastigheden på pumpen. Til dette anvendes funktionen writePWMVoltage fra Arduino Support pakken.

#### 3.4.1.3 Valve Control

Funktionen til styring af ventilen har en boolean som input. Denne værdi indikerer om en Langerhansk Ø er detekteret af kameraet. Alt efter værdien af denne sættes forbindelsen til ventilen høj eller lav med funktionen writeDigitalPin.

### 3.4.2 Kamera

Denne bloks formål er, at modtage et feed fra kameraet samt at detektere om en Langerhansk Ø har passeret kameraet. Som det ses på det overordnede blok diagram for softwaren (figur: 3.30) består kamera blokken af 2 underblokke. Nedenstående interne blok diagram (figur: 3.32) viser, hvordan disse blokke er forbundet internt. De enkelte blokkes funktion er yderligere beskrevet herunder.



*Figur 3.32.* IBD - Camera [Software]

#### 3.4.2.1 CameraFeed

Denne bloks funktion er at modtage feedet fra kameraet og gemme billedet i handles. Til dette anvendes funktionen snapshot, der gemmer det nuværende billede som en variabel.

#### 3.4.2.2 DetectIslets

I denne funktion foregår alt billedbehandlingen på det omsamlede billede. Billedet segmenteres for at fjerne støj og andet væv. Alt efter om en Langerhansk Ø er detekteret eller ej returneres true eller false.

### 3.4.3 Funktioner

I nedenstående liste er systemets funktioner opsummeret.

- initArduino
- pumpControl
- valveControl
- loadCell
- cameraFeed
- detectIslets

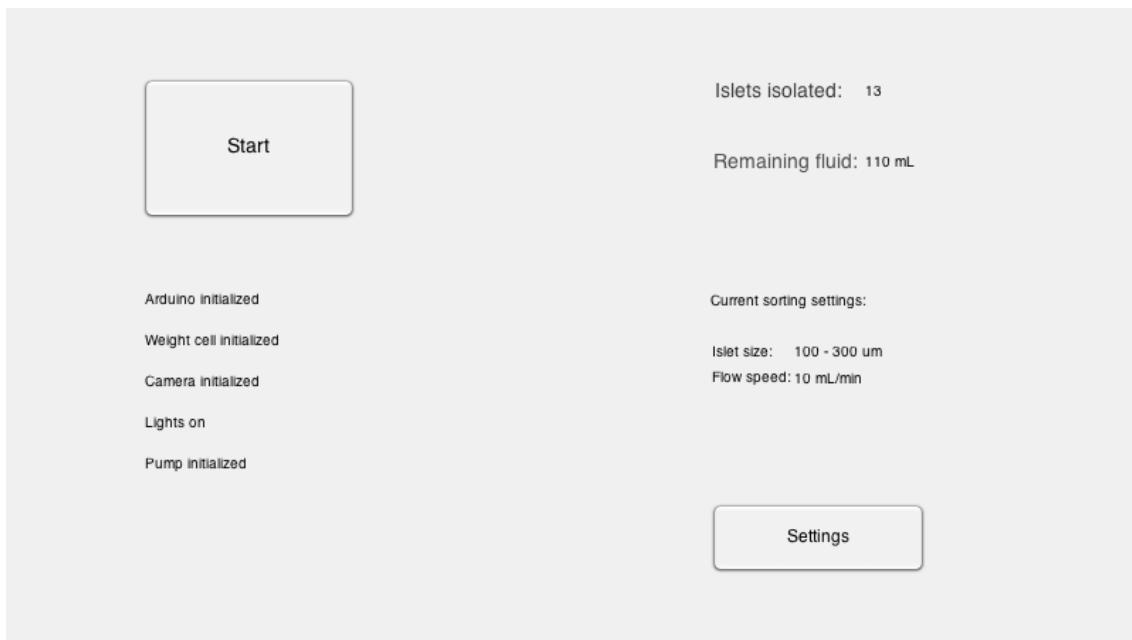
Herudover skal systemets indstillinger kunne ændres, samt data om sorteringscyklussen skal logges. Til dette implementeres funktionerne settings og exportData.

- settings
- exportData

### 3.4.4 User Interface

#### 3.4.4.1 Hovedvindue

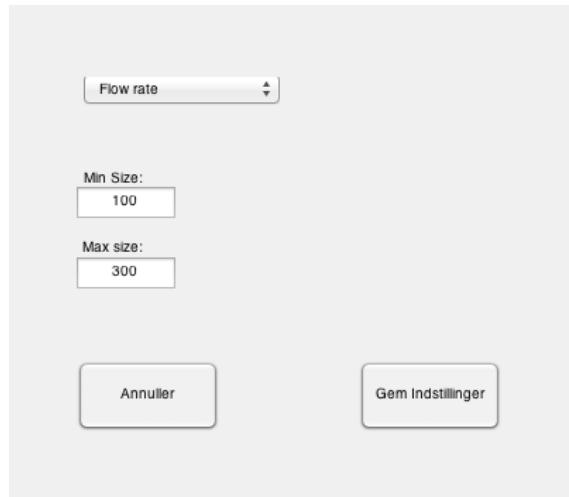
I figur 3.33 er et Mockup af GUI'en vist. I venstre side er der placeret en *Start knap*, som skifter stadie til en *Stop knap* når den er klikket. Under denne knap er en række indikatorer placeret til, at give operatøren feedback om status omkring initialiseringen af Hardwaren. I højre side af GUI'en er der placeret tekstfelter til, at give brugeren feedback omkring den nuværende sorteringscyklus, samt de anvendte indstillinger. Under disse felter er en knap til *Indstillinger*. Når denne klikkes åbnes et nyt vindue, hvor operatøren kan ændre i indstillingerne.



**Figur 3.33.** Mockup af GUI

### 3.4.4.2 Indstillinger

I figur 3.34 er et Mockup af *Indstillingsvinduet* vist. Via 2 tekstfelter har operatøren mulighed for, at ændre størrelsen for de celler som systemet skal sorterer. Herudover har operatøren via en dropdown menu mulighed for at ændre flowhastigheden for pumpen. I Indstillingsvinduet er der yderligere placeret en "Annuler" knap og en "Gem Indstillinger" knap.



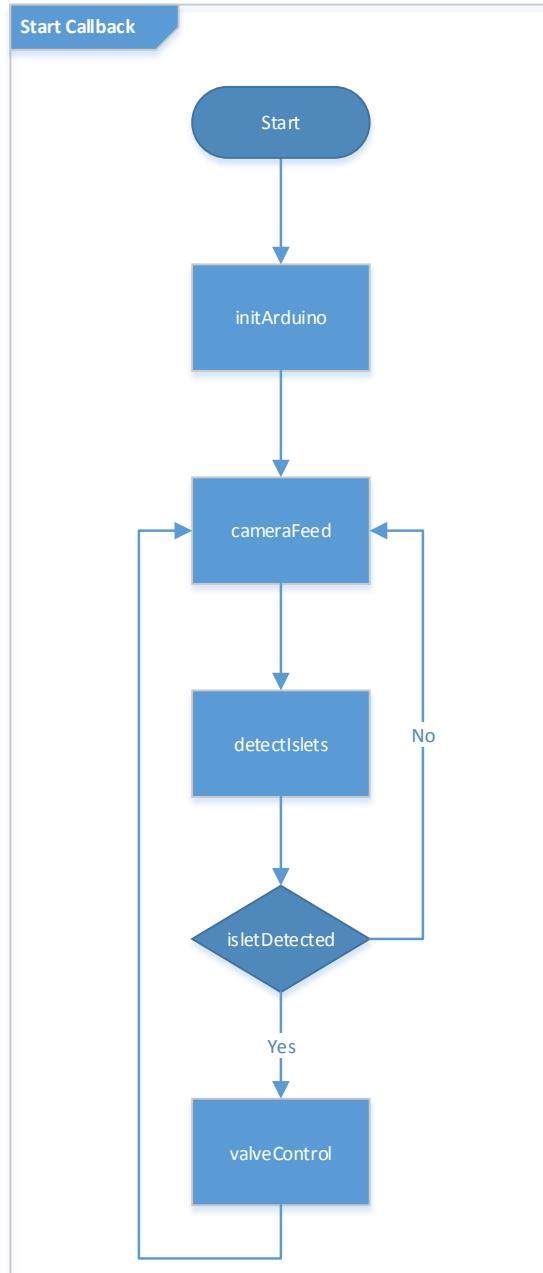
**Figur 3.34.** Mockup af Indstillinger

### 3.4.4.3 Callbacks

For de 3 knapper i GUI'en oprettes der 3 callback funktioner, hvor forskellig kode eksekveres når knapperne klikkes. Disse 3 callback funktioner er nærmere beskrevet herunder bl.a. vha. flow chart diagrammer.

### 3.4.4.4 Start Callback

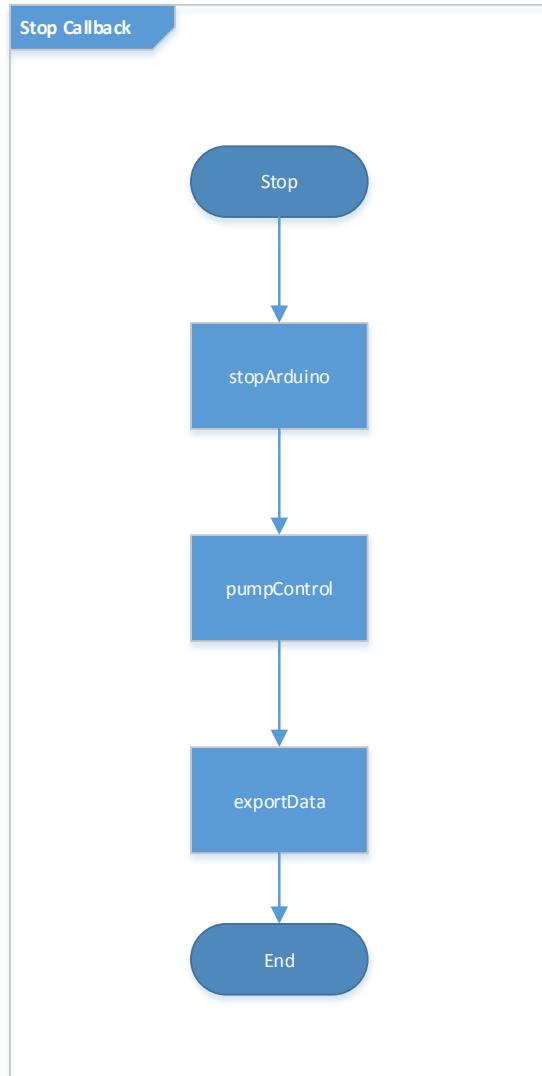
Denne callback funktion kaldes når operatøren klikker på Start knappen på GUI'en. Flowet i callbacket er vist i figur 3.35.



**Figur 3.35.** Flowchart diagram for Start Callback

#### 3.4.4.5 Stop Callback

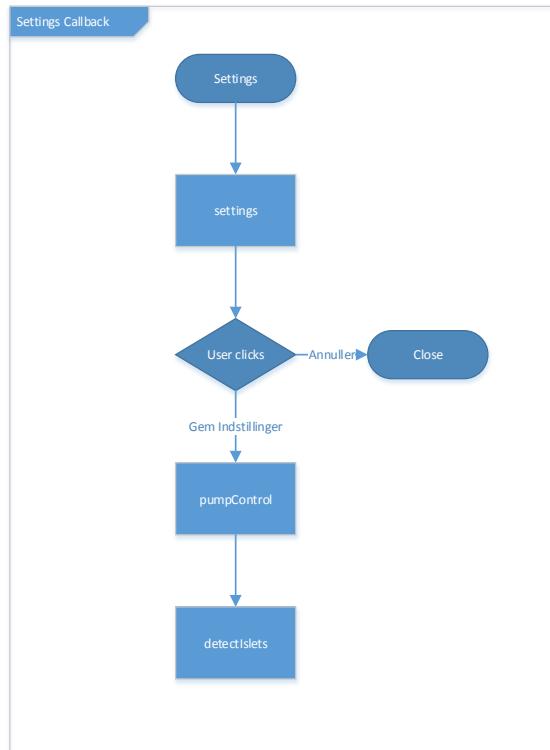
Denne callback funktion kaldes når operatøren klikker på Stop knappen på GUI'en. Flowet i callbacket er vist i figur 3.36.



**Figur 3.36.** Flowchart diagram for Stop Callback

### 3.4.4.6 Indstillinger Callback

Denne callback funktion kaldes når operatøren klikker på Settings knappen på GUI'en. Flowet i callbacket er vist i figur 3.37. Når knappen klikkes åbnes et nyt vindue, hvor systemets indstillinger kan ændres. De ændrede indstillinger anvendes i funktionerne detectIslets og pumpControl.

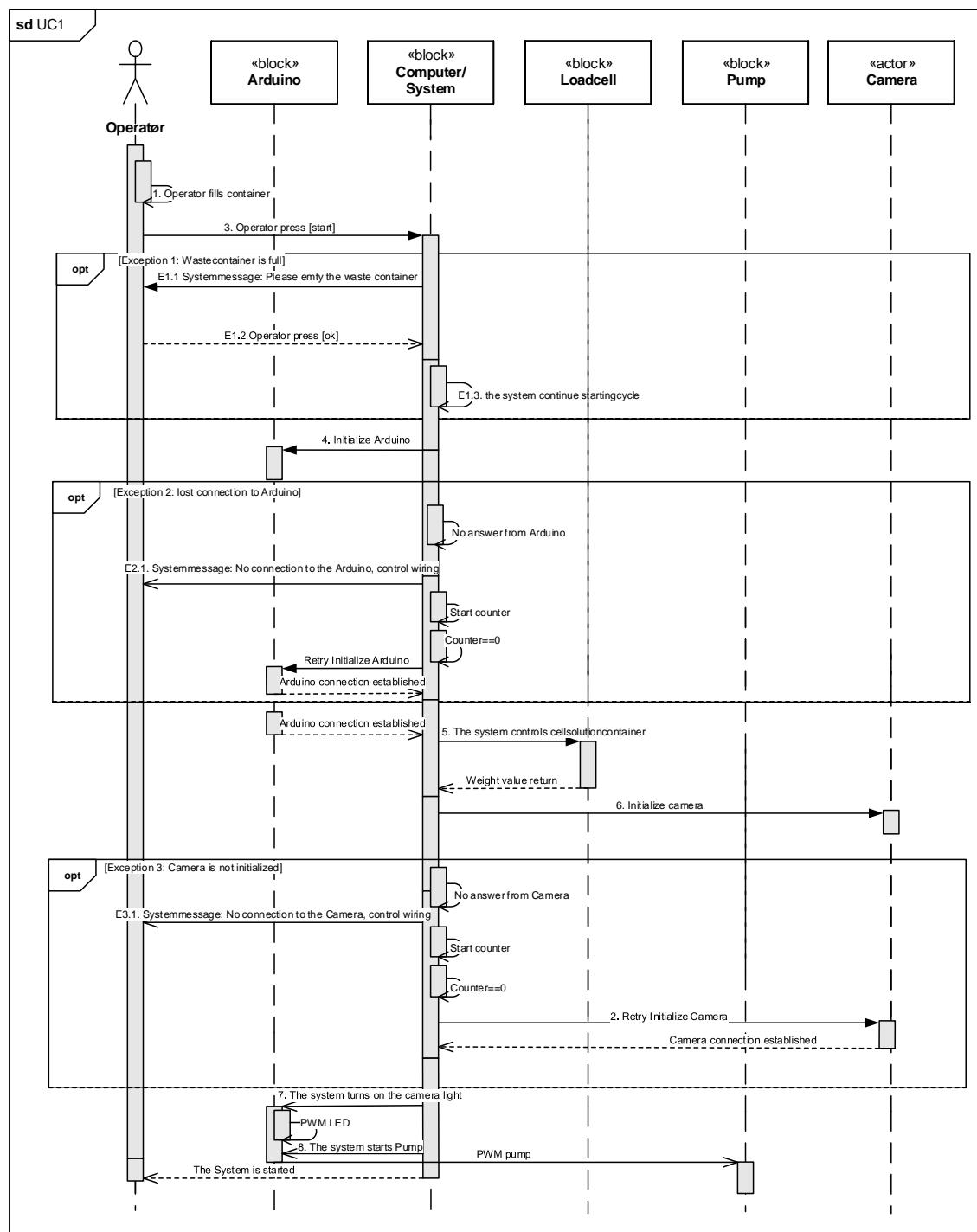


*Figur 3.37.* Flowchart diagram for Settings Callback

### 3.5 Sekvensdiagrammer

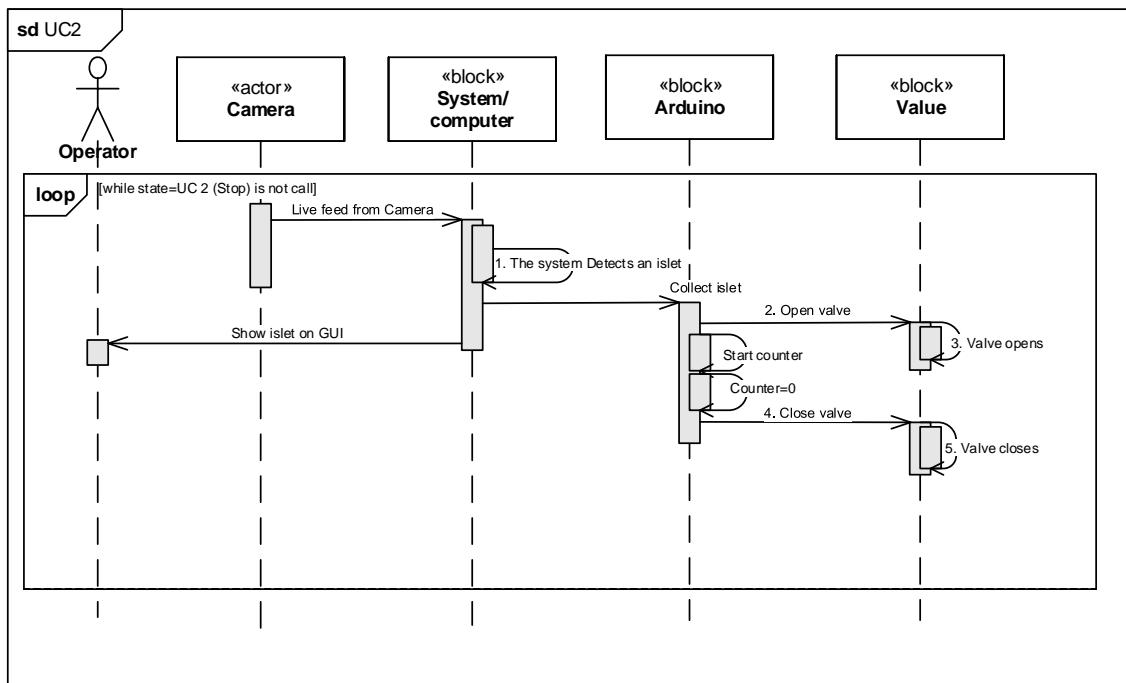
Sekvensdiagrammerne viser normalforløbene for de givne usecases i kravspecifikationen. På diagrammerne er også vist usecasenes undtagelser og udvidelser. Sekvensdiagrammerne bruges til udviklingen af produktet, hvor kan skabe sig et overblik omkring interaktionen af elementerne på et højere plan. De enkelte blokke svarer til de blokke som er specifiseret under block definitions afsnittet: 3.3.3.

### 3.5.1 Sekvensdiagram for usecase 1



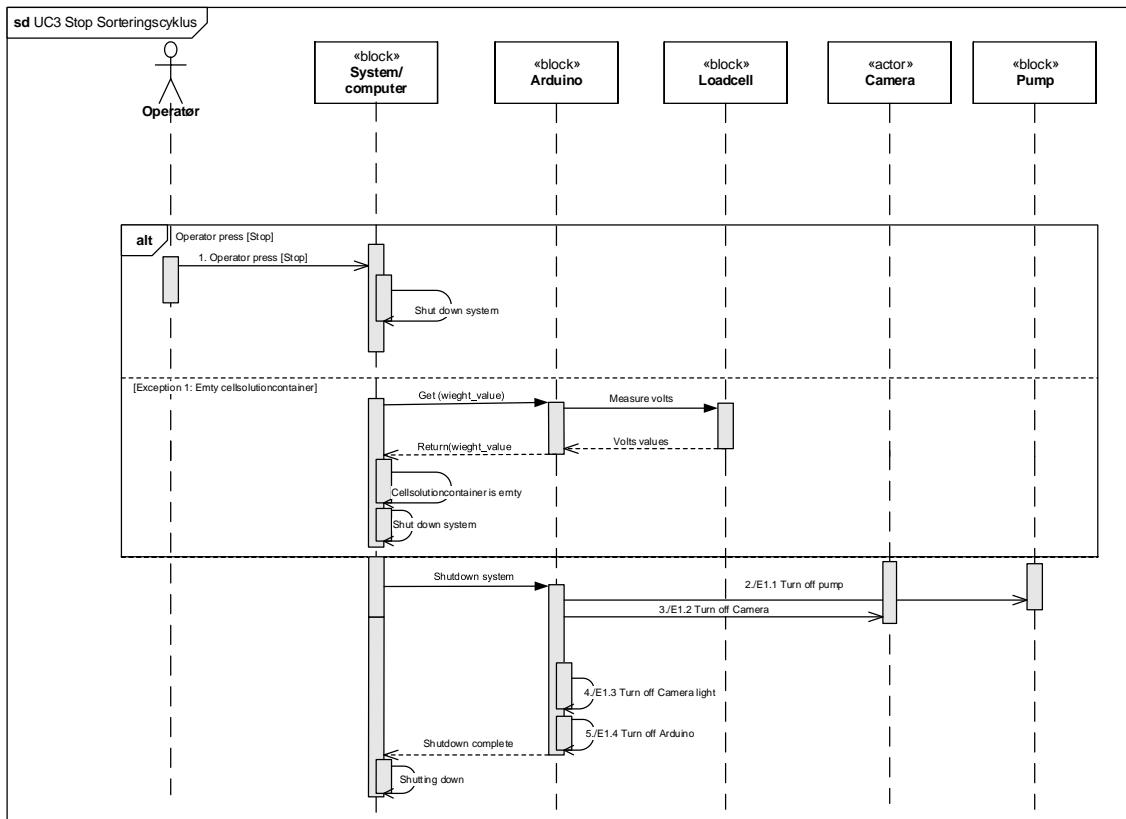
*Figur 3.38.* Sekvensdiagram for usecase 1

### 3.5.2 Sekvensdiagram for usecase 2



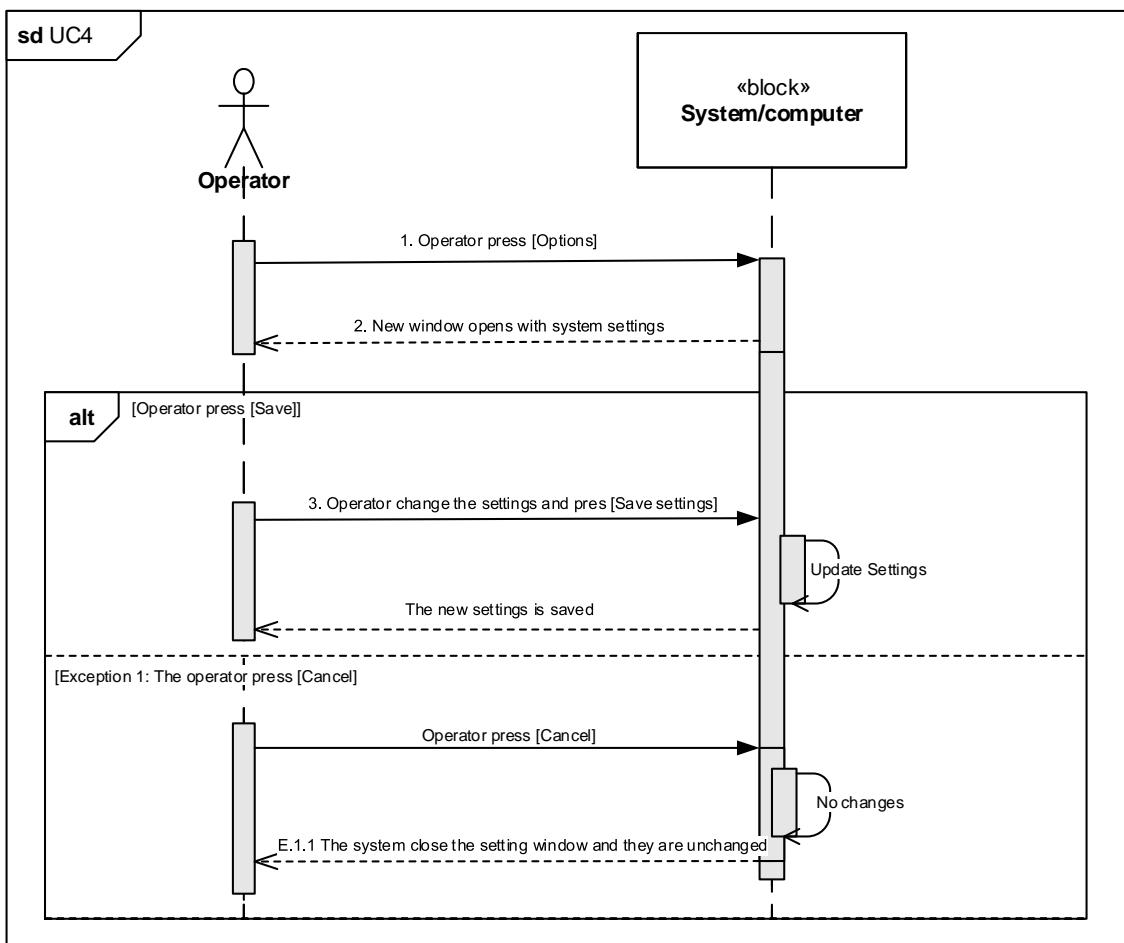
*Figur 3.39.* Sekvensdiagram for usecase 2

### 3.5.3 Sekvensdiagram for usecase 3



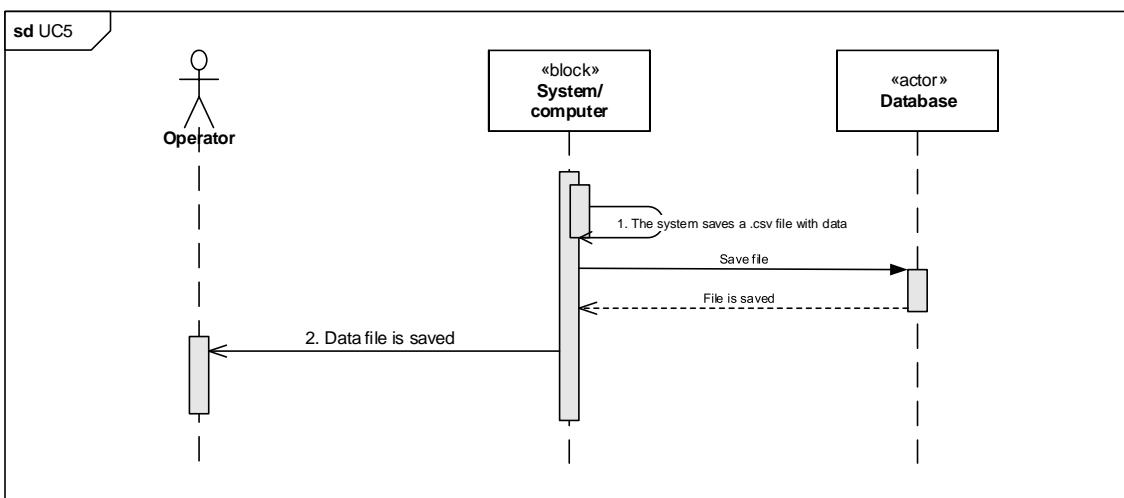
*Figur 3.40.* Sekvensdiagram for usecase 3

### 3.5.4 Sekvensdiagram for usecase 4



Figur 3.41. Sekvensdiagram for usecase 4

### 3.5.5 Sekvensdiagram for usecase 5



Figur 3.42. Sekvensdiagram for usecase 5



# Implementering 4

---

## 4.1 Indledning

Implementeringsdokumentet beskriver hvordan de enkelte dele i systemet *The Cell Collector* er implementeret, herunder hardware og software delene. Implementeringsfasen forløber sig efter en iterativ proces, hvor de enkelte dele er udviklet og testet før de sammensættes med andre del elementer. På denne måde er prototypen gradvist samlet og løbende testet i gennem forløbet.

### 4.1.1 Formål

Formålet med dette dokument er, at beskrive i detaljer hvordan de enkelte delelementer fra designdokumentet konkret er implementeret.

### 4.1.2 Læsevejledning

Dokumentet er overordnet opbygget i et hardware og software afsnit. Grundlæggende beskrives det hvordan de enkelte del elementer udvikles mod et færdig produkt. Der vil være en beskrivelse af hver komponent, samt enhedstest og dokumentation for hvordan det er implementeret. Til sidst i dokumentet er der en integrationstest, hvor delene samles til et produkt. Dokumentet er tæt bundet sammen med design dokumentet og kravspecifikationen.

### 4.1.3 Versionshistorik

Version	Dato	Beskrivelse	Initialer
0.1	13/11 2015	Dokument sendt til review	AE og AT
1.0	19/11 2015	Dokument efter review	AE og AT

## 4.2 Hardware

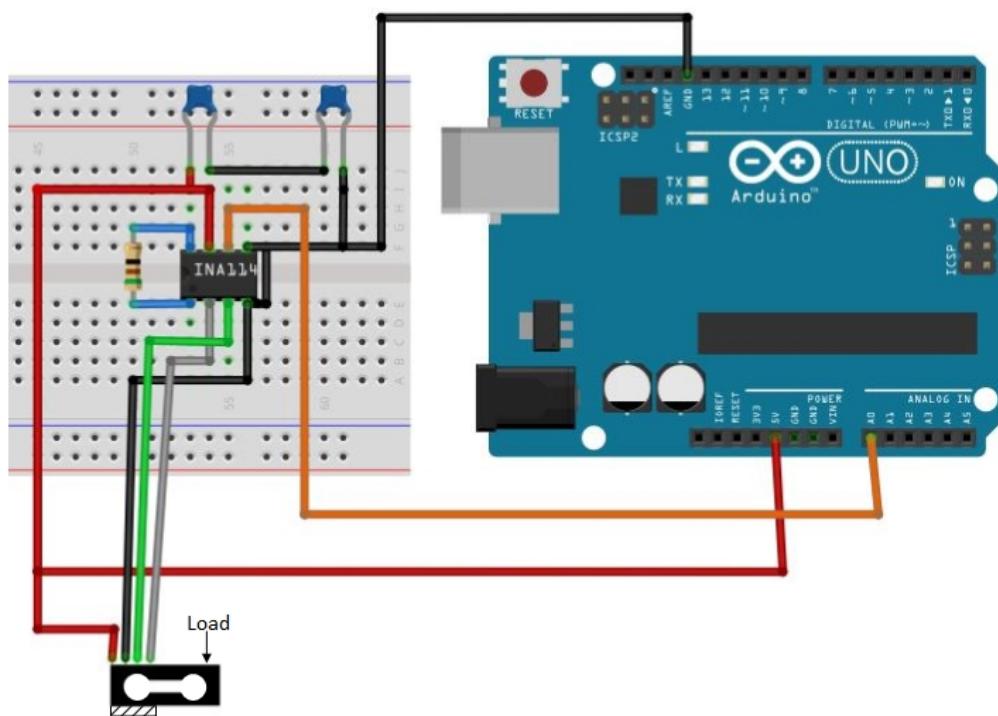
Efter den indledende identifikation og specifikation i design og kravspecifikationen kan implementeringsfasen nu udføres. Hardware implementering af *The Cell Collector* består af enhedstest af hver komponent, med følgende dokumentation. Enhedstestene er lavet i den skrevne rækkefølge og integreret i samme. I dette afsnit vil der være kredsløbsdiagrammer, teori, beregninger og beskrivelser, hvilket er udarbejdet sideløbende med udviklingen af produktet.

### 4.2.1 Vægtcelle

Vægtcellen skal kontroller om opløsningsbeholderen er tom. Dette afsnit er skrevet i tæt efterfølge af design dokumentet.

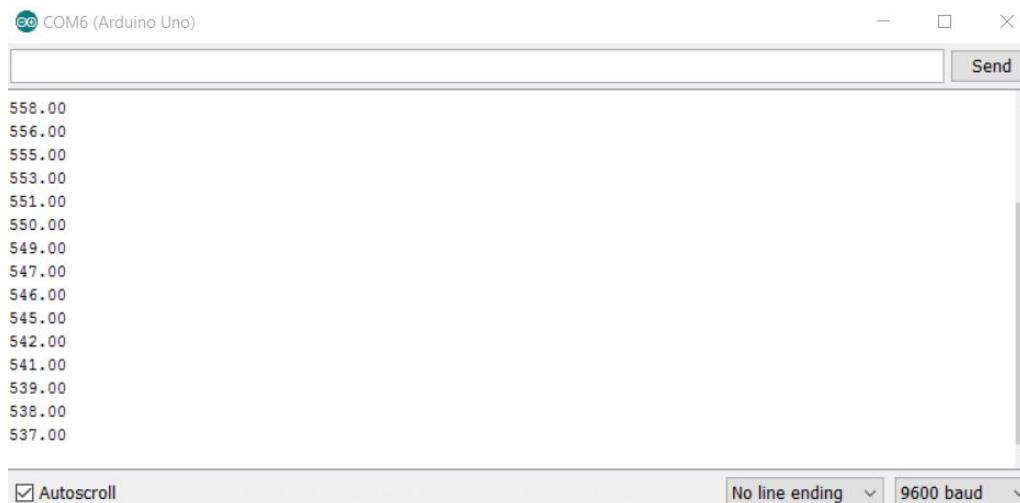
#### 4.2.1.1 Enhedstest af vægtcelle

Efter at diagrammet er fastlagt, testes forbindelserne nu på et *fumlebræt* se figur 4.1 for test opstilling



**Figur 4.1.** Test opstilling for vægtcelle

De primære dele af test koden har været `sensorValue = analogRead(A0);` og `Serial.println(sensorValue);`, hvor ved at inputtet på `A0` er læst i *serial monitor* som vist på figur 4.2 ved langsom tømning af beholderen.



**Figur 4.2.** Værdier fra A0 i *serial monitor*

Se bilag A.3.1 for at se hele koden til testen. Til enhedstesten er der brugt et voltmeter til, at måle udgangsspændingen på INA114 for, at se om arduinoens ADC læste rigtigt. Til at sammenligne med voltmeteret, blev 4.1 brugt til at konvertere ADC’ens bits værdi om til spænding.

$$\text{analogRead}(A_0) * \frac{5}{1024} = \text{spænding i volt} \quad (4.1)$$

Testopstilling af vægtcellen ser ud som på 4.3 med celleopløsningsbeholderen. I softwaren kræves det en kalibrering for at vægtcellen er præcis, dette er implementeret i afsnit 4.3.4.5.



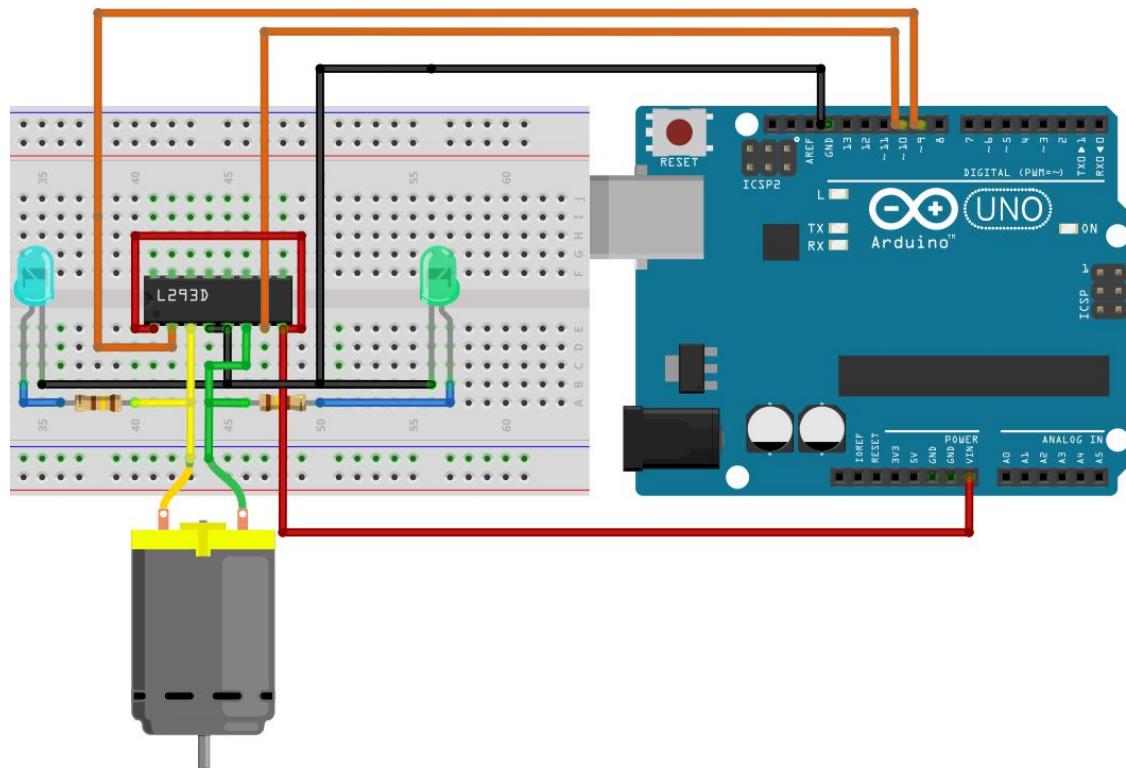
**Figur 4.3.** Illustration af opstilling med vægtcelle og celleopløsningsbeholder

## 4.2.2 Pumpe

Pumpen består af en DC motor og ud fra databladet kan det ses at den skal bruge 12VDC og 0.3A for at køre. Da arduinoen langt fra kan leverer den nødvendige strøm til at drive motoren, skal der bruges en motordriver med en ekstern strømforsyning.

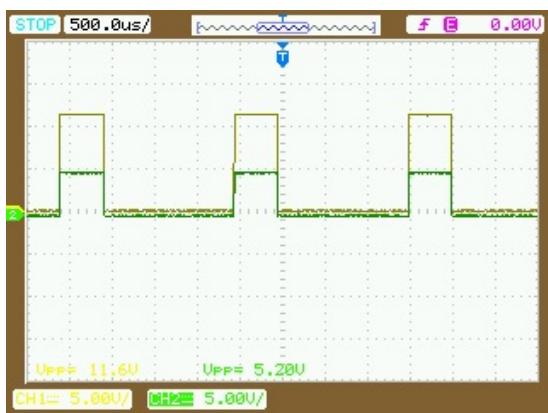
### 4.2.2.1 Enhedstest af motor og motordriver

På figur 4.4 kan motordriveren og arduinoens opstilling ses på et *fumlebraet*. I opstilling er der to lysindikatorer, som viser om motoren kører den ene eller den anden vej. Retningen kan styres ved at bytte om på hvilken udgang der sættes lav(0V) og hvilke der PWM konfigureres.

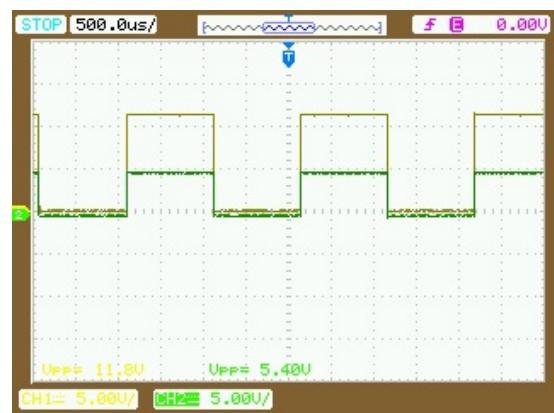


**Figur 4.4.** Illustration af opstilling med motor(pumpe), motordriver og arduino

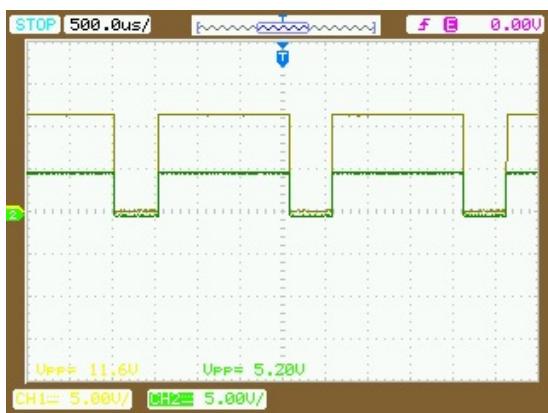
For at teste motordriverens evne til at lave PWM med 12 volt, er PWM signalet fra arduinoen og motordriveren sammenlignet med hinanden vha. et oscilloskop. Billeder fra denne test er vist på billederne 4.5, 4.6, 4.7 og 4.8 den høje gule graf er fra motordriveren og den grønne lave graf er fra arduinoen.



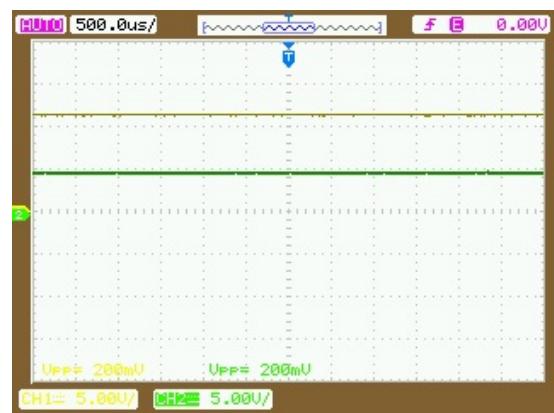
**Figur 4.5.** Arduino og motordriver med 25% PWM



**Figur 4.6.** Arduino og motordriver med 50% PWM



**Figur 4.7.** Arduino og motordriver med 75% PWM



**Figur 4.8.** Arduino og motordriver med 100% PWM

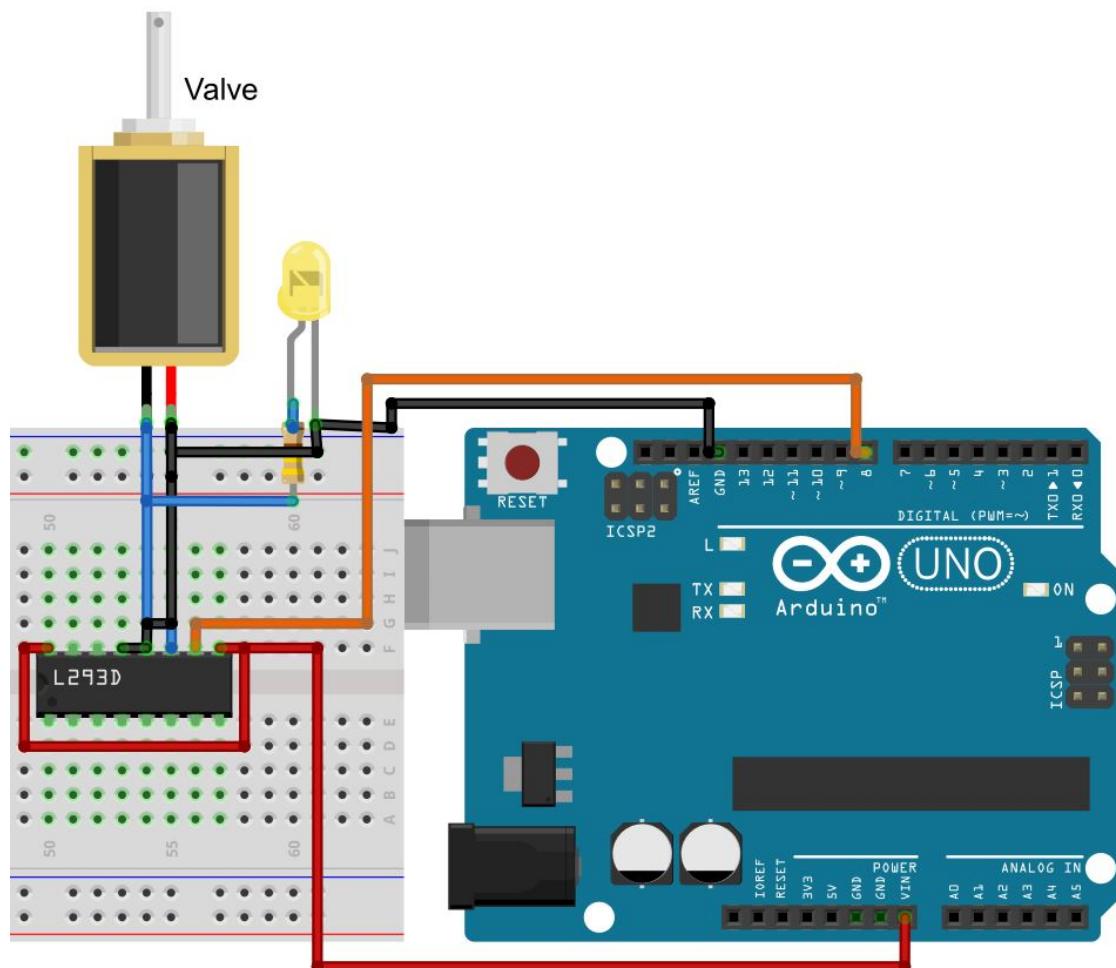
De elementære funktioner, som er brugt til test koden er `digitalWrite(9, LOW);` og `analogWrite(10, 127);`; se hele testkoden i bilag A.3.2

### 4.2.3 Ventil

Ventilen skal sortere de langerhanske øer fra det eksokrine væv, derfor er ventilen en vigtig del af hardwaren. I integrationstesten bør tidsintervallet med hvornår ventilen skal åbnes og lukkes udføres, for en sikring af at øerne bliver isoleret.

#### 4.2.3.1 Enhedstest for ventil

På figur 4.9 er det illustreret hvordan testopstilling er sat op på et fumlebræt. I testkoden er der hovedsagligt brugt `digitalWrite(8, LOW);` og `digitalWrite(8, HIGH);`, hvorved arduinoen sætter udgangen lav(0V) og høj(5V). Motordriveren bruger signalet til det at give ventilen 12V når den er høj. Se bilag A.3.3 for hele testkoden



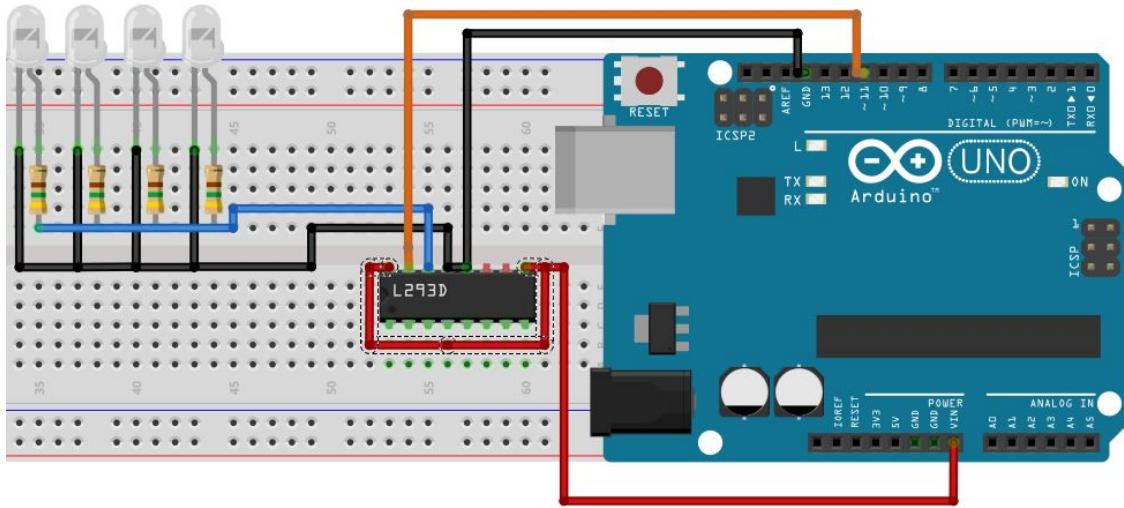
**Figur 4.9.** Illustration af testopstilling med ventil og motordriver

#### 4.2.4 Kameralys

Kameralyset skal hjælpe operatøren med at optimere lysforholdet til kameraet for, at maksimerer forholdene og muligheden for at detektere de langerhanske øer. Testen burde være lavet med lysdioderne inde i kamerahuset, for at optimere feedbacket fra kameraet ved at ændre lysstyrken. Grundet kameraets manglende kvalitet er denne test nedprioriteret.

#### 4.2.4.1 Enhedstest for Kameralet

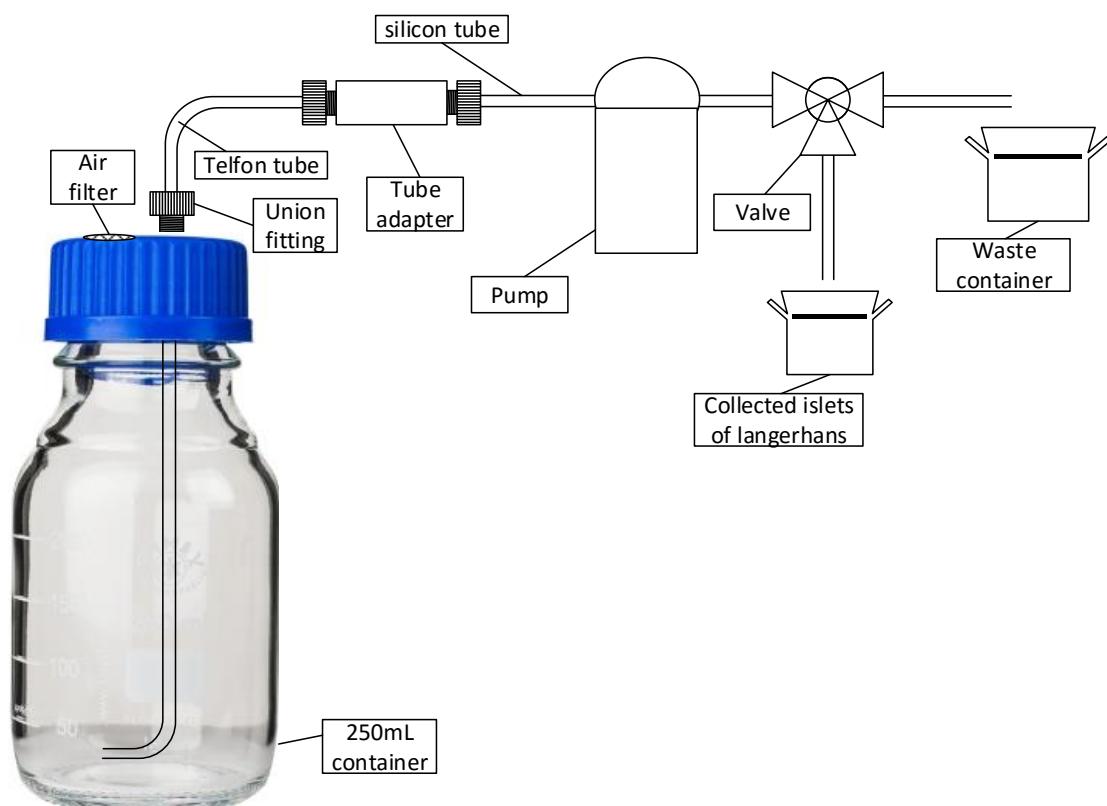
På figur 4.10 vises testopstilling på et *fumlebræt* for Kameralyset er der brugt samme testkode som til pumpen se afsnit 4.2.2.1.



**Figur 4.10.** Illustration af testopstilling med kameralys og motordriver

#### 4.2.5 Ikke-elektroniske dele

På figur 4.11 vises en illustration af de ikke-elektroniske dele for hele systemet. De ikke-elektroniske dele består af en opløsningsningsbeholder, som navnet fortæller, indeholder opløsningen med de langerhanske øer. Fra opløsningsbeholder går en teflonslange, den er hård hvilket er en fordel når den skal suge væsken op. For at en peristaltisk pumpe kan benyttes skal det være en silikoneslange, da den er eftergivelig. Overgangen fra teflonslangen til silikoneslangen er oprettet vha. en slangeadapter. Derefter føres silikone slangen til pumpen, til ventilen, hvor fra der går en silikoneslange fra hver af ventilens studser. Silikoneslangerne fra ventilen føres videre til wastebeholderen og beholderen til de isoleret øer.



**Figur 4.11.** Illustration af ikke elektroniske dele

Til test af ikke-elektroniske dele, er simuleringsvæsken med timianfrø og demineraliseret vand brugt. Timianfrøene der er brugt har en størrelse på 0.4mm hvilket burde kunne komme igennem, men frøene stopper ved ventilen. Derfor opfattes testen af ventilen som en fejlet test. Det bør undersøges hvorfor at frøene stopper ved ventilen, i datablad A.1.4 står der, at ventilens indgange er på 1mm.

#### 4.2.5.1 Stykliste

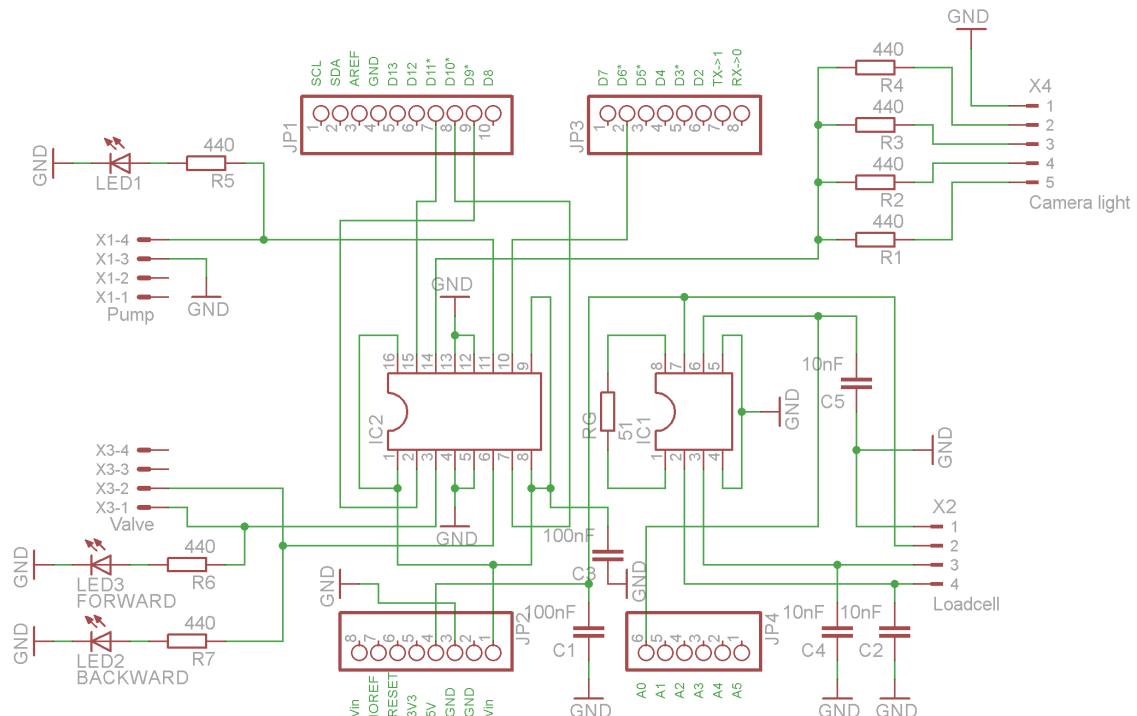
Beskrivelse	Type	Antal
Opløsningsbeholder	ML 33184	1 stk
Flaskeadapter låg	ML 75441	1 stk
1/4"Prop til låg	UP P309	1 stk
1/16"Skrue flangeløs	UP P201	1 stk
1/16"Ferrule flangeløs	UP P200	1 stk
Flaskeadapter filter	n/a	1 stk
Teflonslange	ML 94142	0.2 m
Slangeadapter	UP P630 + UP646	1 stk
Silikoneslange(0.5mm)	n/a	0.3 m
Slangestudser til ventil(10-32)	VP X210-60005	3stk

#### 4.2.6 PCB design

For at samle hardwaren i projektet er der lavet et printkort formet, som et shield til arduino platformen. På printet er motordriveren der driver pumpen, ventilen og kameralyset. Motordriveren forsyner igennem en ekstern strømforsyning, som derved forsyner delene motordriveren styrer. I kredsløbet er der en operationsforstærker, som har til opgave at forstærke signalet fra vægtcellen og leverer det til ADCén på arduinoen. Pinheaderne på printet er med lange ben for, at de passer ned i arduinoens pinheadere. Til at forbinde pumpen og ventilen, er der brugt stiktypen molex 5566-4 som kan trække en stor strøm. Til de to komponenter skal der kun bruges to ledninger, men for at sikre at stikkene ikke sættes forkert er der valgt et 4pin stik, hvor forbindelses benene er forskellige. Det betyder at hvis stikkene ved en fejl bliver ombyttet, vil hverken pumpen eller ventilen virke. Til kameralyset benyttes et 5 polet stik af typen molex 5268-05, stikket tåler en mindre strøm end de andre stik, men nok til kameralyset. Til vægtcellen er der valgt et stik af samme type, men i en 4 polet udgave. På printet er der lysindikatorer til at hjælpe med fejlfinding og service på systemet. Lysindikatorer består af 3 lysdiode; LED1 lyser når at ventilen er åben, LED2 lyser når motoren kører fremad og LED3 lyser hvis motoren kører baglæns. Ydermere er der placeret kondensatorer parallelt med forsyningerne til at afkoble støj fra miljøet omkring systemet.

#### 4.2.6.1 PCB kredsløbsdiagram

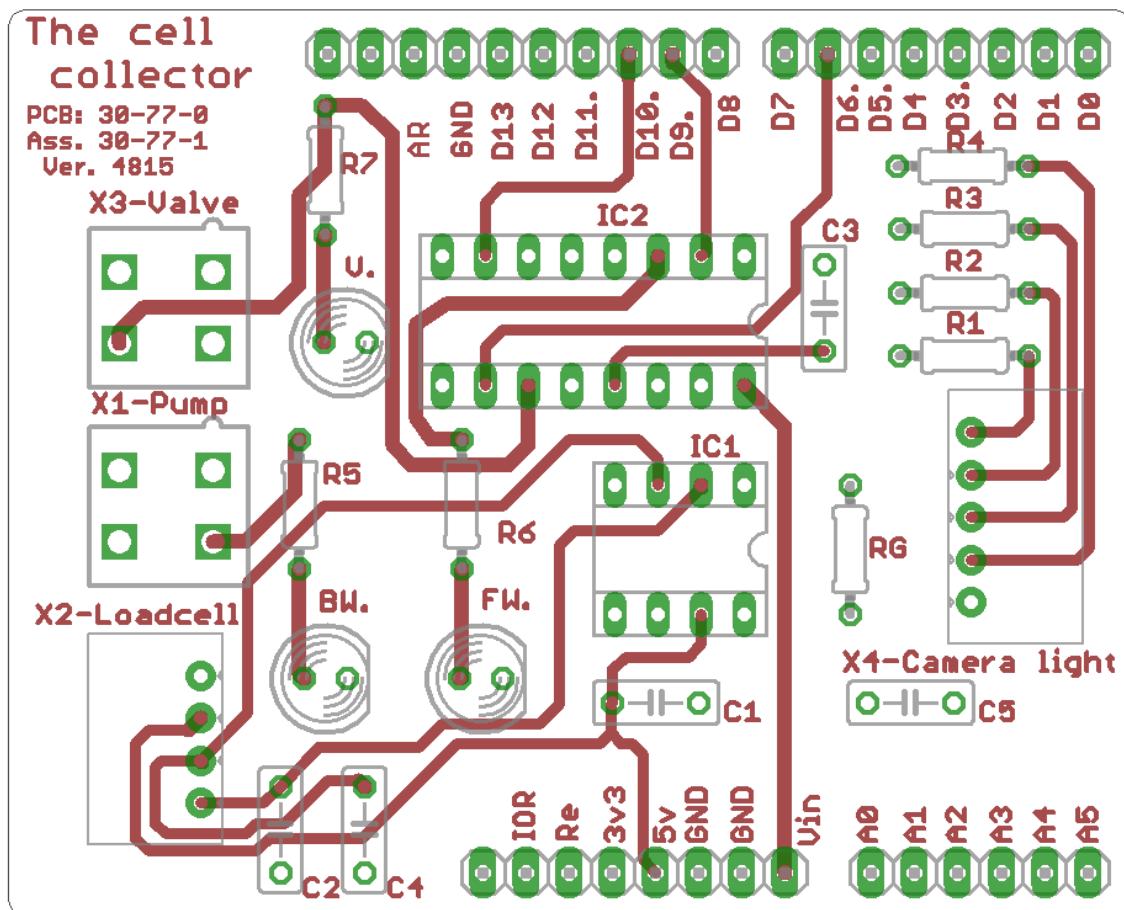
På figur 4.12 er vist kredsløbet for printkortet i *Eagle*. I diagrammet er der valgt at bruge GND symbolet i stedet for at forbinde signalerne til hinanden, dette er gjort for at simplificere diagrammet. JP er pinheaderne der forbinder signalet til arduinoen, stik er betegnet med X, C er kondensatorer og R er modstande.



Figur 4.12. Diagram for printkortet

#### 4.2.6.2 PCB toplayout

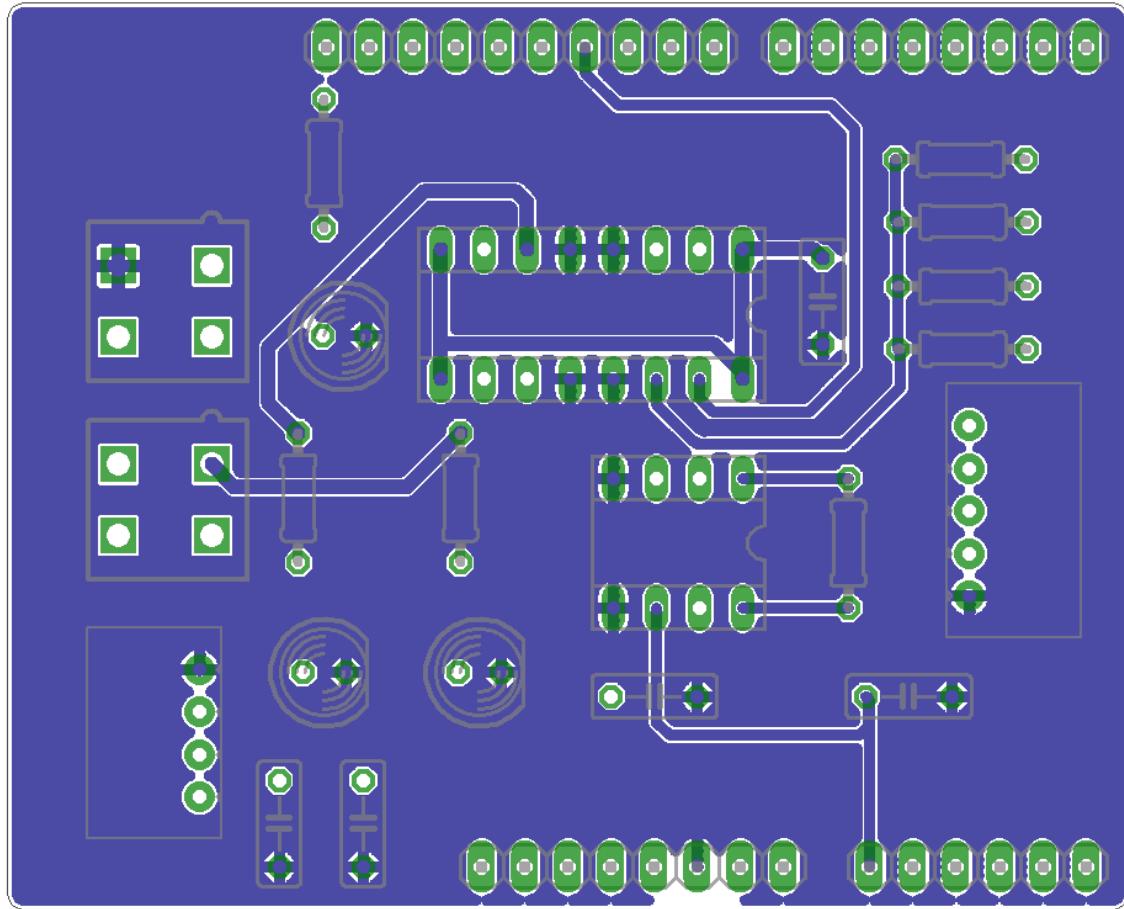
På topplanet vist på 4.13 er komponenterne til printet placeret. Der er forsøgt at lave de fleste forbindelser i dette plan for, at undgå brud i groundplanet. Sliketrykket er ikke lavet, som vist på figur 4.13 fordi det ikke laves på Ingeniørhøjskolens print produktion. Det er bla. derfor at alt den beskrivende tekst er lavet i top kobber laget.



Figur 4.13. PCB toplayout

#### 4.2.6.3 PCB bundlayout

I bundlaget er der valgt at bruge et groundplan af følgende grunde; for at mindske indstrålingen af EMC, bruge den som køleplade til L293D og mindske fremførelsen af ground banerne.



*Figur 4.14.* PCB bundlayout med groundplane

#### 4.2.6.4 Stykliste til PCB: 30-77-1

Til samling af printet skal der bruges komponenterne i nedenstående tabel.

Navn	Type	Antal
C2,C4,C5	10nF	3 stk
C1, C3	100nF	2 stk
IC1	INA114	1 stk
IC2	L293D	1 stk
Jp1-pinheader(stackable)	1x10	1 stk
Jp2,3-pinheader(stackable)	1x8	2 stk
Jp4-pinheader(stackable)	1x6	1 stk
LED1-Green	5mm(T13/4)	1 stk
LED2-Yellow	5mm(T13/4)	1 stk
LED3-Blue	5mm(T13/4)	1 stk
R1-R7	440Ω	7 stk
RG	51Ω	1 stk
X1,X3	molex 5566-4	2 stk
X2	molex 5268-04	1 stk
X4	molex 5268-05	1 stk

#### 4.2.6.5 Stykliste for stik til print

For at sætte pumpe, ventil, vægtcelle og kameralyset til printkortet er komponenterne i nedenstående tabel nødvendige.

Navn	Type	Antal
X1,X3	2x2 Minifit Jr.(5557)	2 stk
X1,X3	Crimp(5556)	4 stk
X2	Polantal 4(5037)	1 stk
X4	Polantal 5(5037)	1 stk
X2,X4	Crimp(5263)	9 stk

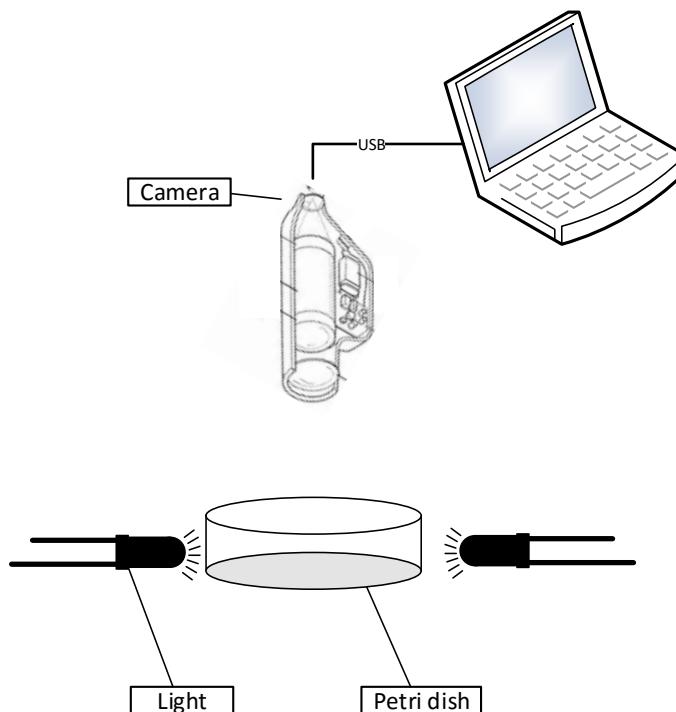
## 4.3 Software

Software implementeringen af *The Cell Collector* beskrives det i detaljer, hvordan de enkelte Matlab funktioner er implementeret. Herudover vil der i afsnittet være beskrivelser og resultater af enhedstest.

### 4.3.1 Kamera

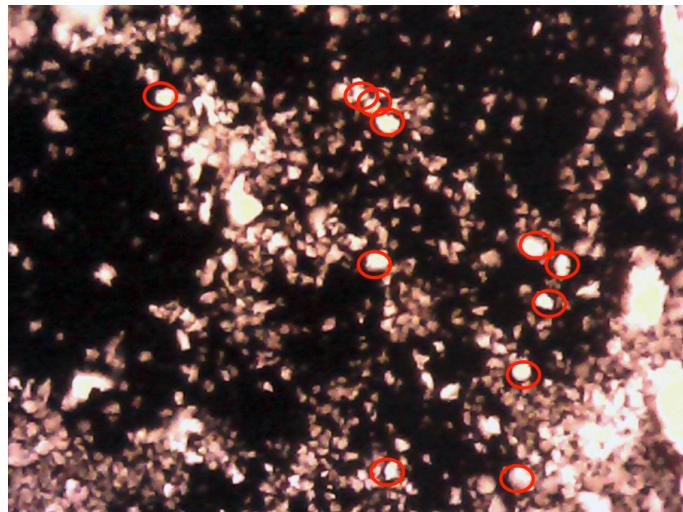
#### 4.3.1.1 Test

Det indkøbte kamera er testet ved at tage en serie af billeder af langerhanske øer. Billedserien skulle i første omgang danne grundlag for den videre billedbehandling. Første test bestod af 107 billeder taget af langerhanske øer, herunder billeder kun af isolerede øer og enkelte baggrunds billeder. Forsøgsopstillingen var en efterligning af den nuværende sorteringsproces, hvor opløsningen hældes i petriskål. Petriskålen placeres på en sort bagplade, hvorefter operatøren isolerer øerne ved at kigge gennem et mikroskop. Forsøgsopstillingen er vist i figur 4.15.



**Figur 4.15.** Forsøgsopstilling

I forsøgsopstillingen anvendtes der i stedet for mikroskopet det indkøbte kamera. En række billeder blev udvalgt til yderligere analyse, hvor Søren Gregersen udpegede hvilke elementer der var øer. På billede 4.16 er der markeret, hvor en ø er placeret.



**Figur 4.16.** Øer udpeget af Søren Gregersen

Efter nærmere analyse af billederne viste det sig at det indkøbte kamera ikke var af tilstrækkelig kvalitet til at lave segmentering på billederne. Når øerne blev observeret gennem et almindeligt mikroskop var der langt større kontrast i mellem øerne og det ekstra væv. På billederne fra kameraet er der ikke denne forskel, hvilket udelukker segmentering på denne parameter. Det kan ses på billede 4.16 at der er områder hvor lysintensiteten er lige så høj, som de steder hvor øerne er markeret. Dette gør, at segmentering på baggrund af lysintensiteten kan udelukkes. Herudover er det tydeligt, at billederne ikke er skarpe nok, hvilket gør at det er svært at vurdere hvad der er øer og ikke øer. De parametre, hvor kameraets kvalitet ikke er tilstrækkelig er derfor bl.a. autoeksponering og autofocus. En bedre styring af autoeksponeringen ville kunne give et bedre kontrast forhold i mellem øerne og det omkringliggende væv, mens en bedre autofocus ville hjælpe på skarpheden i billedet, hvilket muliggøre segmentering baseret på strukturer.

Herudover er der en polariserende effekt på billederne, hvor nogle af elementerne lyser meget kraftigt op grundet belysningen. Dette gør det er svært at vurdere strukturer og størrelser på elementerne i billedet.

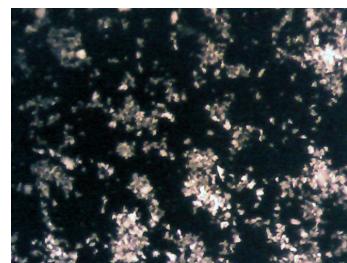
Efter den første test blev det i samarbejde med vejleder og kunde besluttet, at det indkøbte kamera ikke er af tilstrækkelig kvalitet til at detektere langerhanske øer. I dette system bliver der i stedet udviklet et sæt af billeder, som skal simulere flowet i slangerne. Billederne skal indeholde langerhanske øer, ekstra væv og tilfældig støj for at få dem til at ligne det man observerer gennem et almindeligt mikroskop. Udfra disse billeder skal langerhanske øer detekteres, hvorefter systemet skal åbne ventilen. Billedesættet skal ses som en simulering af kameraet. I en senere prototype vil et kamera af højere kvalitet være påkrævet. Her er det især krav til bedre autoeksponering og autofocus som er essentielle. Dette vil være nødvendigt for at øge kontrasten i mellem øerne og det omkringliggende væv for bedre segmentering. Herudover vil et kamera med polariseringsfilter være en mulighed til at fjerne evt. genskær fra belysningen. Hos Farnell er der andre producenter end det indkøbte Duratool mikroskop, som bl.a. har polariseringsfilter. En test af forskellige kameraer ville være oplagt til finde det ideelle kamera til optagelse af langerhanske øer.

### 4.3.1.2 Simulering af kamera

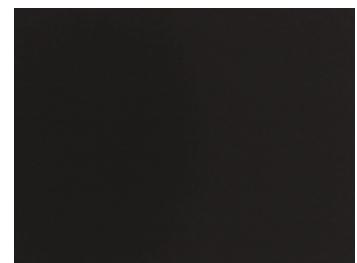
Til at generere et billedsæt, der simulerer langerhanske øer, er der udviklet et Matlab script. Scriptet består overordnet af 3 faser. Den første fase består i at segmentere langerhanske øer udfra et billede og oprette dem som en maske. I anden fase laves en maske bestående af ekstra væv, mens der i tredje fase simuleres flow. I den sidste fase gemmes også de enkelte billeder i formatet .png. De enkelte faser er nærmere beskrevet under deres afsnit. Der anvendes 3 billeder til grundlag for genereringen. Det ene billede viser isolerede øer. Det andet billede viser opløsningen indeholdende øer og ekstra væv. Det sidste billede er et baggrundsbillede uden øer eller ekstra væv. Dette billede anvendes som baggrund for de genererede billeder. Billederne der er valgt stammer fra det indkøbte kamera. Grunden til de kan anvendes som grundlag for genereringen af billeder, på trods af kameraets utilstrækkelige kvalitet, er at øerne og det ekstra væv er adskilt i de enkelte billeder. Dette muliggør en separat segmentering af øer og væv, som herefter kan sammensættes til billeder der ligger tæt op af det man observerer gennem et almindeligt mikroskop. De 3 billeder er vist herunder.



**Figur 4.17.** Billede indeholdende langerhanske øer



**Figur 4.18.** Billede indeholdende ekstra væv



**Figur 4.19.** Baggrundsbillede

### Fase 1

Segmenteringen af langerhanske øer sker ud fra billede 1 4.17, hvor funktionen circleFinder anvendes til at finde centrum og radius på af de fundne celler. Resultatet af circlefinder segmenteringen er vist i figur 4.20



**Figur 4.20.** Circlefinder til ø detektion

Circlefinder funktionen giver følgende funktion, som kan anvendes til at detektere cirkler med. Parametrene Sensitivity og edgeThreshold beskriver, hvor cirkulære objekterne er og generelt hvor følsom algoritmen er. Som det ses på figur 4.20 findes der kun én cirkel pr. ønsket  $\phi$ .

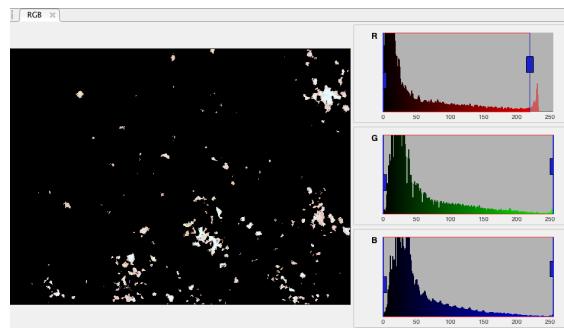
```

1 detectCircles = @(x) imfindcircles(x,[12 30], ...
2     'Sensitivity',0.8500, ...
3     'EdgeThreshold',0.20, ...
4     'Method','PhaseCode', ...
5     'ObjectPolarity','Bright'));
6 [centers, radii, metric] = detectCircles(im);

```

### Fase 2

I anden fase bliver det ekstra væv segmenteret ud fra billede 2 (figur 4.18). Til dette er der anvendt Color Threshold appen i Matlab. Ved hjælp af denne app er der lavet en funktion, som opretter en logisk maske af det ekstra væv. Opsætningen i appen er vist i figur 4.21. Yderligere er der anvendt morfologiske operationer til at fjerne uønskede objekter fra masken, samt fjerne støj. Til at fjerne uønskede objekter er funktionen *bwareafilt* anvendt, med parametrene 150 og 500. Dette fjerner alle objekter under 150 og over 500 sammenhørende pixels. Til at udfylde huller i de enkelte objekter er funktionen *imfill* brugt.



**Figur 4.21.** Color thresholder

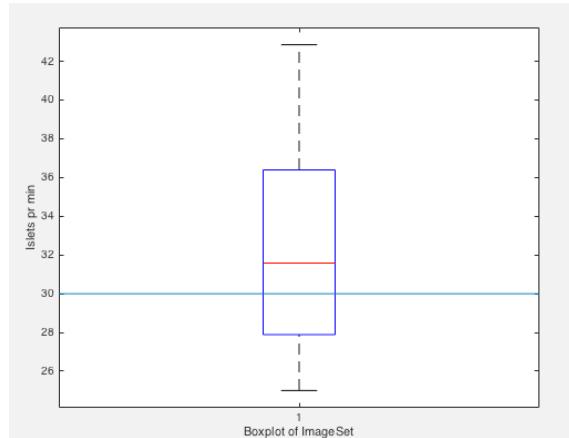
### Fase 3

I fase 3 sker selve flow simuleringen. Flowsimuleringen er opbygget på den måde, at den består af henholdsvis en sekvens indeholdende en langerhansk ø efterfulgt af en sekvens uden en ø. I selve programmet indlæses et nyt billede hvert 0,1 sekund. Derfor skal der genereres en passende mængde billeder, som programmet kan indlæse. Fase 3 er implementeret så der minimum genereres 252 eller maksimalt 432 billeder, hvilket giver en samlet sekvenslængde på 25,2 eller 43,2 sekund. Grunden til at antallet af billeder varierer er at længden af sekvensen uden en langerhansk ø bestemmes udfra en random variabel. Dette gøres for, at simulere at det kan være variabel tid mellem en ny ø kommer i gennem slangen. I scriptet genereres der i alt 18 fulde sekvenser. Det betyder, at der passerer i mellem 25 og 43 øer i minuttet. Antallet af øer der passerer pr. minut er bestemt ud fra formel 4.2:

$$\frac{18}{\frac{n}{10}} * 60 = \text{Antal øer pr. minut}, \text{ hvor } n \text{ er antallet af billeder i sættet} \quad (4.2)$$

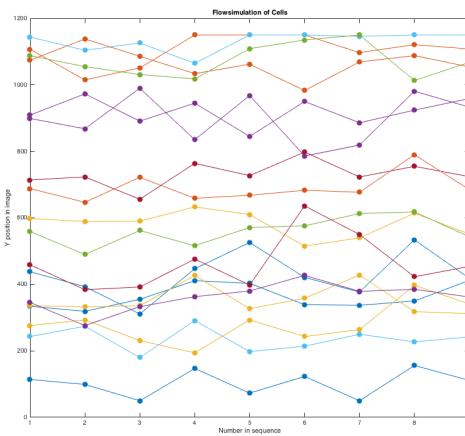
I figur 4.22 er vist et boxplot som viser distributionen af hvor mange øer der passerer i minuttet. Det ses at medianen ligger over 30 øer pr. minut, hvilket betyder at der

i gennemsnit vil komme over 30 øer pr. minut. De 30 øer pr. minut stammer fra hastighedskravet fra systemets kvalitetskrav (1.4.1).



**Figur 4.22.** Boxplot af distributionen af øer pr. minut

Selve flowsimuleringen sker i et for loop. Først udvælges en tilfældig celle ved hjælp af randi (uniform fordelt random variabel). I for loopet opdateres dens center position ud fra 2 variabler (newXPos og newYPos). X positionen springer med et fast interval for hver iteration (160 px). Inden for loopet fastsættes start positionen for cellen med *randi*, som giver et tal mellem 0 og 1200 px (højden på billedet). Herefter opdateres den nye Y position ved hjælp af randn (normal fordelt random variabel) med middelværdi sat til startpositionen og en standard afvigelse på 50 pixels. I figur 4.23 er flow simulationen illustreret for de i alt 18 sekvenser, med en graf for hver celle. Det ses at cellen flytter sin Y position tilfældigt, mens X positionen springer med et fast interval for hver iteration i for loopet.

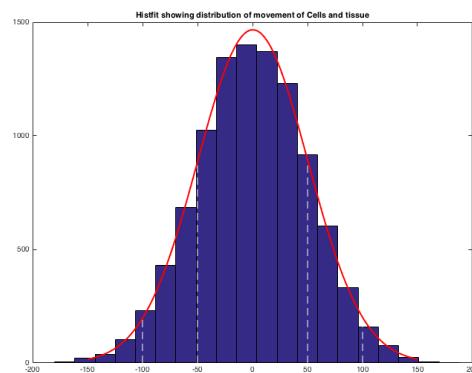


**Figur 4.23.** Illustration over flow simuleringen

Efter generering af ny x og y position oprettes en maske for cellen ved brug af funktionen createCirclesMask. Til at indsætte cellen i baggrundsbilledet indekseres baggrundsbilledet med masken for cellen, så gråtoneværdierne fra det oprindelige billede indsættes.

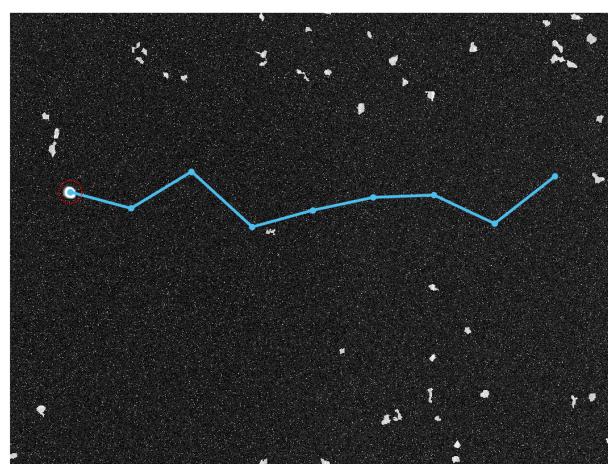
Yderligere er der tilføjet tilfældig støj til billedet i form af "Salt and pepper" og gausisk hvid støj. Her er Matlab funktionen *imnoise* anvendt.

Masken med ekstra væv flyttes for hver iteration med funktionen *circshift*, som bit shifter arrayet cirkulært. Parametrene definerer antallet af rækker og kolonner arrayet skal shiftes. Antallet af kolonner er fastdefineret til 160 px, mens rækkerne skiftes efter en normalfordelt random variabel med mean på 0 og standard afvigelse på 50 pixels. Nedenstående figur 4.24 viser fordelingen af antallet af rækker der flyttes. På figur 4.24 er der vist markører for standard afvigelsen ( $\sigma$ ) og 2\*standard afvigelsen ( $2\sigma$ ) for hver side af mean. I mellem  $-\sigma$  og  $\sigma$  er der 68,26 % sandsynlighed for, at den nye Y position ville ligge inden for dette område. For 2 sd afvigelse ( $2\sigma$ ) er der 95,45 % for, at den nye Y position vil ligge inden for dette område.



**Figur 4.24.** Histogram over fordelingen af ny Y position

Det endelige resultat af billedegeneringen er vist i figur 4.25. Til at illustrere hvordan øen flytter sig er der tilføjet en graf, som viser hvordan dens position ændres for hver iteration. Cellen er markeret med en rød ring på billedet.

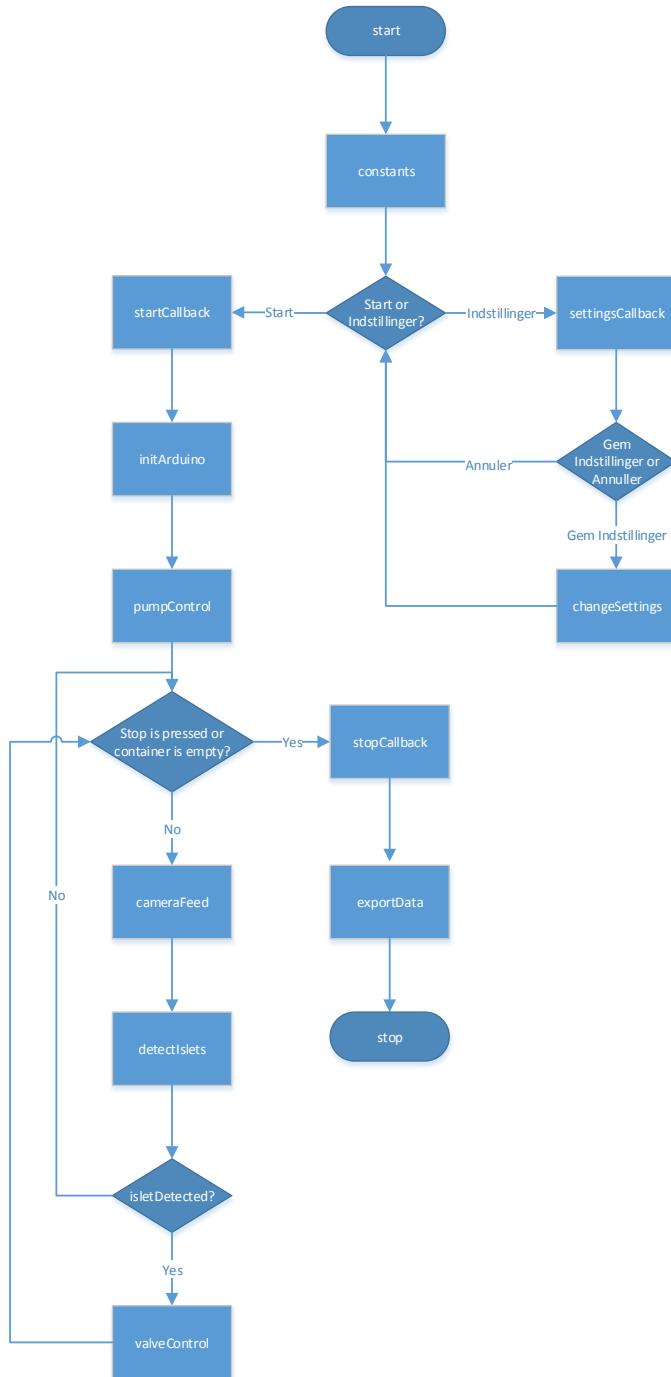


**Figur 4.25.** Endelige resultat for flowsimulering

Implementeringen af scriptet er vedlagt i bilag A.2.

### 4.3.2 Program flow

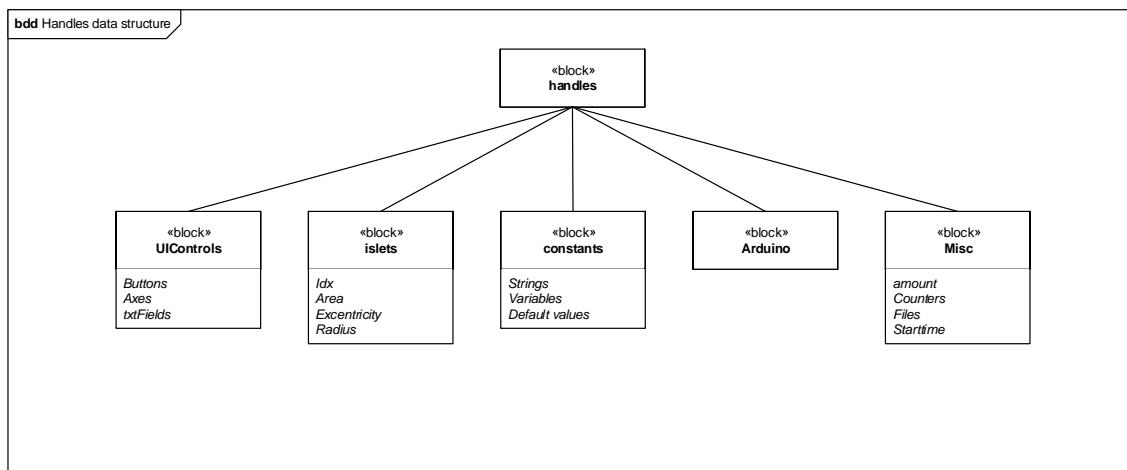
Flowchartet i figur 4.26 viser flowet i det implementerede Matlab program. De enkelte blokke indikerer de udviklede Matlab funktioner, mens beslutningsknuderne indikerer knaptryk, om beholderen eller om en ø er detekteret.



**Figur 4.26.** Flowchart for programmet

### 4.3.3 Data struktur

Dette afsnit beskriver data strukturen for det udviklede Matlab program. Når man arbejder med GUI i Matlab anvendes structed handles, som indeholder alle UI Controls såsom knapper, tekstfelter og axes. Figur 4.27 viser strukturen af handles. Udover UI controls indeholder handles structs med konstanter og data omkring de detekterede øer.



Figur 4.27. Data struktur

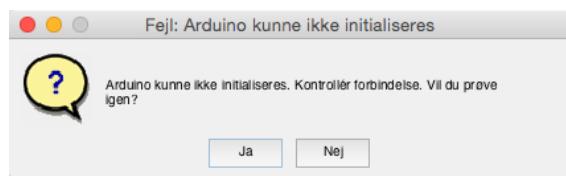
### 4.3.4 Matlab funktioner

I dette afsnit er implementeringen af de enkelte funktioner i Matlab nærmere beskrevet. Selve implementeringen tager udgangspunkt i beskrivelserne fra design dokumentet.

#### 4.3.4.1 initArduino

I denne funktion opsættes og initialiseres Arduinoen. Til dette anvendes funktionen *Arduino*, fra Arduino Support pakken. Som parameter til funktionen angives port navnet og board typen (i dette system en Arduino Uno). Arduino variablen gemmes herefter i handles, som en variabel.

Selv initialiseringen sker i et try/catch statement, hvis Arduinoen ikke kan initialiseres åbnes en dialogboks (figur: 4.28), hvor operatøren kan vælge om systemet skal forsøge at oprette forbindelse igen.



**Figur 4.28.** Dialogboks til systembesked

Implementeringen af initArduino er vist i nedenstående kode.

```

1 while true
2 % Try/Catch statement
3 try
4     % Initialize Arduino and save variable in handles
5     handles.a = arduino('/dev/cu.usbmodem1411', 'Uno');
6     % If succes - break while loop
7     break
8 catch
9     % Open quest dialog to inform operator that inializing failed
10    choice = questdlg('Arduino kunne ikke initialiseres. Kontroller forbindelse. Vil du proeve
11        igen?', 'Fejl: Arduino kunne ikke initialiseres', 'Ja', 'Nej', 'Ja');
12
13    switch choice
14        % Do nothing - retry inializing again
15        case 'Ja'
16
17        % Returns to end function
18        case 'Nej'
19        return
20    end
21 end

```

#### 4.3.4.2 constants

Denne funktion er en hjælpefunktion, som indeholder alle konstanter der anvendes rundt omkring i programmet, herunder strings og variabler. Dette gøres for, at mindske mængden af hard-kodede variabler ude i de enkelte Matlab funktioner og gøre programmet lettere at vedligeholde. Funktionen kaldes ved opstart af programmet i GUI'ens openingFcn callback funktion, som eksekveres inden GUI'en gøres synlig. I funktionen *constants* gemmes de specifiserede variabler i et struct kaldet *constants* i handles (se 4.3.3).

#### 4.3.4.3 cameraFeed

Denne funktion er implementeret udfra beskrivelsen i designdokumentet (3.4.2.1 s. 65). Dog er den ændret i forhold til ikke at modtage et feed fra kameraet, men i stedet indlæses et billede fra en prædefineret mappe. Funktionen har handles som input og output.

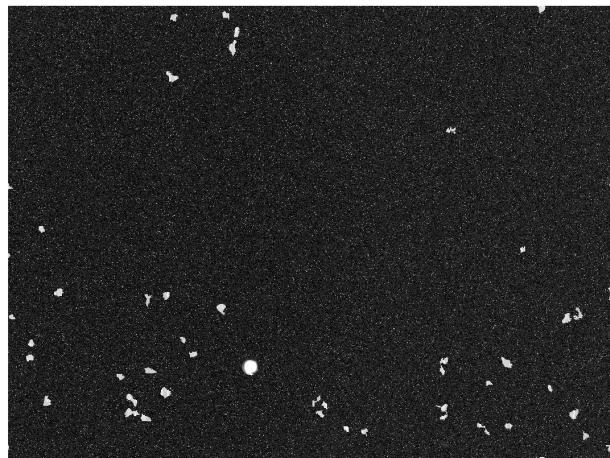
I et while loop itereres der gennem mappen, hvor en counter variabel definerer hvilket billede der skal indlæses. Efter dette nedskaleres billedet til halv størrelse med *imresize*. Dette gøres for, at optimere tiden det tager at behandle billedet, samt vise det på GUI. I denne forbindelse kunne der være en potentiel risiko for, at information gik tabt på billedet i nedskalleringen. Dette blev testet på en række billeder for, at se om algoritmen (beskrevet i 4.3.4.4 ville fejle på det skalerede billede, samt om tiden for billedebehandlingen blev forbedret. Billedet gives herefter som parameter til funktionen *detectIslet* (4.3.4.4).

Matlab implementeringen er vedlagt i bilag A.2.

#### 4.3.4.4 detectIslets

I denne funktion sker selve billedesegmenteringen af de langerhanske øer. Selve segmenteringen sker via en række morfologiske operationer, som er nærmere beskrevet i dette afsnit. For at illustrere effekten af de enkelte steps er der i løbet af afsnittet vist billeder.

I figur 4.29 er det oprindelige billede vist. Billedet danner udgangspunktet for selve segmenteringen. Billedet indeholder én langerhansk ø.



**Figur 4.29.** Oprindelige billede

Første step af billedesegmenteringen består i, at konvertere det oprindelige billede (4.29) til en logisk maske. Til dette er Matlab funktionen *im2bw* anvendt. Effekten af *im2bw* er vist i figur 4.30. Funktionen konverterer alle pixels med en lysstyrke over det angivne threshold (0,72) med 1, og pixels under med 0. Input billede er et 8 bit billede, dermed indikerer et threshold på 0,72, at pixel værdien skal være over 183 for at blive inkluderet i masken. I nedenstående kode er implementeringen vist.

```
1 % Converts the image to logical, based on threshold. 0.72 indicates pixels
2 % with luminance level above 0,72 (255*0.72 = 183) is converted to 1. Pixels below this
3 % level is converted to 0
4 bw = im2bw(im,0.72);
```

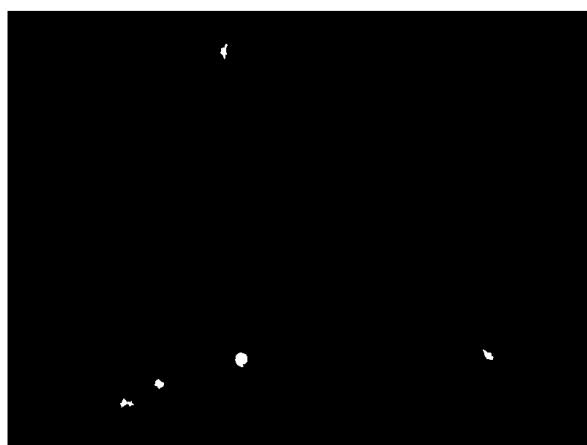


**Figur 4.30.** Billedet konverteret til logisk

I næste step bliver unødig støj fra masken fjernet. Til dette anvendes funktionen *bwareaopen*. Denne fjerner alle objekter i masken under 100 sammenhørende px. Dette fjerner de mindste støjkomponenter fra billedet. Effekten af denne funktion er vist i figur 4.31. Herudover kaldes funktionen *bwlabel*, som indeksere hvert af de sammenhørende objekter i masken. Dermed får hvert objekt i figur 4.31 et unikt index fra 1 til 5.

```

1 % Removes all objects below 100 connected pixels
2 bw = bwareaopen(bw,100);
3
4 % Label connected components
5 L = bwlabel(bw);
```



**Figur 4.31.** Små objekter fjernet

I næste step sker selve ø detekteringen. Detektionen baseres på, at øerne har en mere cirkulær form end resten af vævet. Derfor hentes der egenskaber fra objekterne i masken vha. matlab funktionen *regionsprops*. De egenskaber der hentes er med til, at beskrive hvor cirkulært objektet er. De egenskaber som hentes er arealet, center positionen, omkredsen og excentriciteten. Excentriciteten beskriver hvor langstrakt objektet er. Hvis excentriciteten er 0 er det en perfekt cirkel mens det vil være en langstrakt ellipse hvis værdien er 1.

Udfra arealet og omkredsen af objekterne anvendes de 2 nedenstående formler til bestemmelse af 2 værdier for radiusen af objektet:

$$Areal = R1^2 * \pi \Rightarrow R1 = \sqrt{\frac{Areal}{\pi}} \quad (4.3)$$

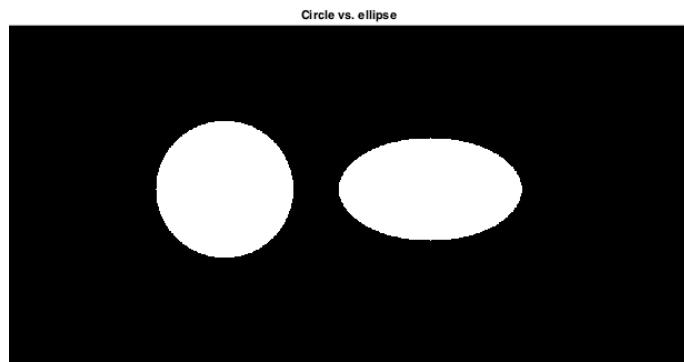
$$Omkreds = 2 * R2 * \pi \Rightarrow R2 = \frac{Omkreds}{2 * \pi} \quad (4.4)$$

Udfra disse 2 radius værdier, vil det objekt, hvor der er mindst forskel i mellem de 2, være det objekt der er mest cirkulært. Til at illustrere dette er der i figur 4.32 vist en cirkel og en ellipse begge med samme areal. Udfra *regionsprops* og de 2 formler for radius kan der bestemmes 2 radiuser for henholdsvis cirklen og ellipsen. Forskellen bestemmes ved, at trække de 2 beregnede værdier fra hinanden. I udregningerne nedenfor er den første kolonne værdier for cirklen, mens den anden er for ellipsen. Som det ses er forskellen mindst ved cirklen, hvilket indikerer, at den er mest cirkulær.

```

1 r1 = 100.0017 99.9380
2
3 r2 = 99.7978 105.7890
4
5 rDif = 0.2039  5.8510

```



**Figur 4.32.** Cirkel sammenlignet med ellipse

Indekset for det objekt med mindst forskel i radius og indekset for det objekt med den mindste excentricitet gemmes i variabler. I nedenstående kode er det vist, hvordan dette er implementeret i Matlab.

```

1 stats = regionprops(bw,'Area','Centroid','Perimeter','Eccentricity');
2
3 r1 = sqrt([stats.Area]/pi);
4 r2 = [stats.Perimeter]/(2*pi);
5 % The absolute difference between the 2 radius is calculated
6 rDif = abs(r1-r2);
7 % The index of the lowest difference is returned
8 [~, idx] = min(rDif);
9 % The index of the object with the lowest eccentricity is returned
10 [~, idx2] = min([stats.Eccentricity]);

```

I det sidste step i segmenteringen af den langerhanske ø kontrolleres det om radius forskellen og excentricitet ligger under nogle fast definerede grænseværdier. Værdierne er fundet ved, at analysere en række billeder og deres objekters radius og excentricitet. Radius og excentriciteten bliver gemt i logfilen, så de kan analyseres for senere at justere grænseværdierne. Hvis objektets værdier er under disse grænseværdier er en ø detekteret. Når en ø er detekteret illustreres det med en grøn ring på GUI. Til dette er funktionen *viscircles* anvendt, hvor center positionen og radius for objektet anvendes.

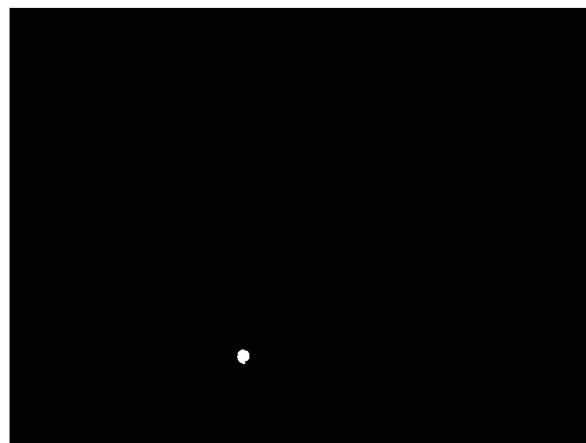
For at løse udfordringen med, at den samme ø vil optræde på det næste billede, er der implementeret et smalt detekteringsvindue (100 pixels bredt). Når øen er detekteret inden for dette område bliver variablen *isletDetected* sat til true. Denne variabel anvendes til styring af ventilen. Implementingen af dette er vist i nedenstående kode.

```

1 % If the difference between radius AND eccentricity is below the defined
2 % values an islet has been detected
3 if rDif(idx) < 0.45 && stats(idx2).Eccentricity < 0.51
4
5 % Show circle of cell on the 2 axes
6 viscircles(h,stats(idx2).Centroid,r1(idx2)+10,...
7 'LineStyle','-', 'edgecolor','g', 'LineWidth',1,'DrawBackgroundCircle',...
8 false);
9 viscircles(s,stats(idx2).Centroid,r1(idx2)+10,...
10 'LineStyle','-', 'edgecolor','g', 'LineWidth',1,'DrawBackgroundCircle',...
11 false);
12
13
14 % Detection window (0 to 100 pixels)
15 if stats(idx2).Centroid(1) <=100
16 handles.flag1 = true;
17 handles.count = handles.count+1;
18 set(handles.txtIslets,'String',num2str(handles.count));
19 handles.isletDetected = true;
20 end

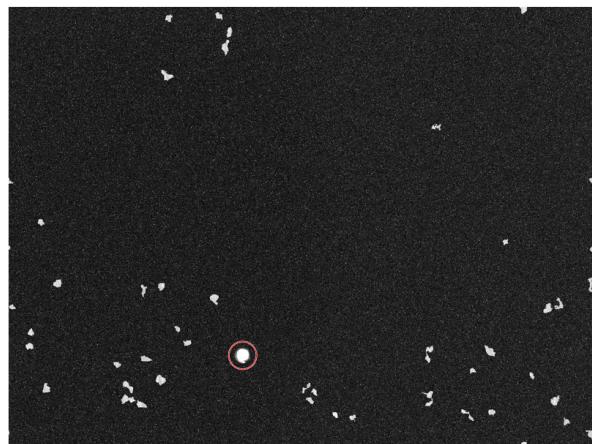
```

Figur 4.33 viser slutresultatet af segmenteringen, hvor masken kun indeholder den langerhanske ø fra det oprindelige billede.



*Figur 4.33.* Slutresultat af segmentering

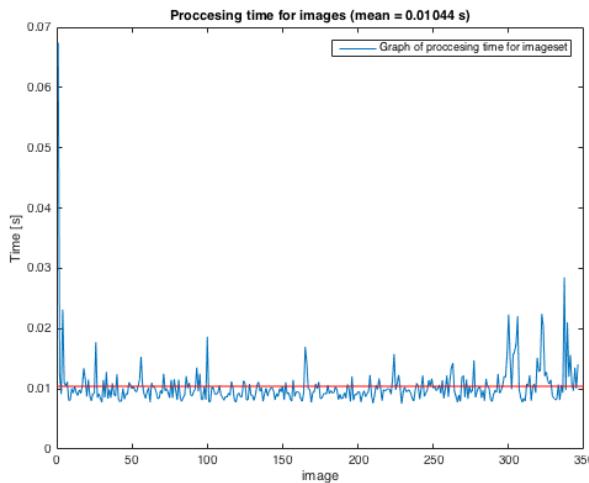
I figur 4.34 er det oprindelige billede vist, med markering af den detekterede  $\phi$ .



**Figur 4.34.** Oprindelige billede med detekteret  $\phi$

### Test af billeddeprocessering

En udfordring med billeddeprocesseringen er, at hastigheden ikke må være for langsom i forhold til frameraten for kameraet. Som beskrevet under funktionen cameraFeed (4.3.4.3) indlæses et nyt billede hvert 0,1 sekund. Billeddeprocesseringen skal dermed være hurtigere end dette for at følge med. Til test af dette er Matlab funktionen tic/toc anvendt, som mäter hvor langt tid billedprocessingen tager om at eksekvere. I figur 4.35 viser grafen hvor lang tid det har taget, at behandle de enkelte billeder. Den røde streg viser den gennemsnitlige tid for processeringen. For den nuværende implementering tager billeddeprocesseringen 0,0104 sekund pr. billede, hvilket er indenfor grænsen på 0,1 sekund. Tiden vil variere alt efter tilgængelig processorkraft og om der skal udføres andre opgaver, eksempelvis opdatere figurer på GUI.



**Figur 4.35.** Test af tidsforbrug for billeddeprocessering

#### 4.3.4.5 loadCell

Funktionen til load cellen er implementeret efter beskrivelsen i design dokumentet. Dens funktion er, at konvertere det analoge input (V) til indholdet (ml) i celleopløsningsbeholderen. Dette er implementeres ved en lineær model:

$$mL = a * \text{input} + b, \text{ hvor } a \text{ er hældningen og } b \text{ er skæringen med y aksen} \quad (4.5)$$

Det analoge input ganges altså med en faktor a plus et offset b for at konvertere spænding til antal ml. Nedenstående tabel viser indgangsspændingen for forskellige mængder i beholderen. Udfra disse data er der lavet en lineær regression for at finde hældningen a og skæringen b.

ml i beholder	Analog input
0 ml	1,9487 V
25 ml	2,0440 V
50 ml	2,1320 V
75 ml	2,2297 V
100 ml	2,3109 V
125 ml	2,4071 V
150 ml	2,4961 V
175 ml	2,5821 V
200 ml	2,6760 V
225 ml	2,7654 V
250 ml	2,8587 V

Tabel 4.5: Kalibreringsdata for loadcellen

I Matlab er funktionen *fitlm* anvendt til at finde det bedste lineære fit. Regressionen er baseret på Least Square metoden [Matlab, 2015]. Ud fra beregningerne i Matlab er hældningen a og skæringen b fundet til hhv:

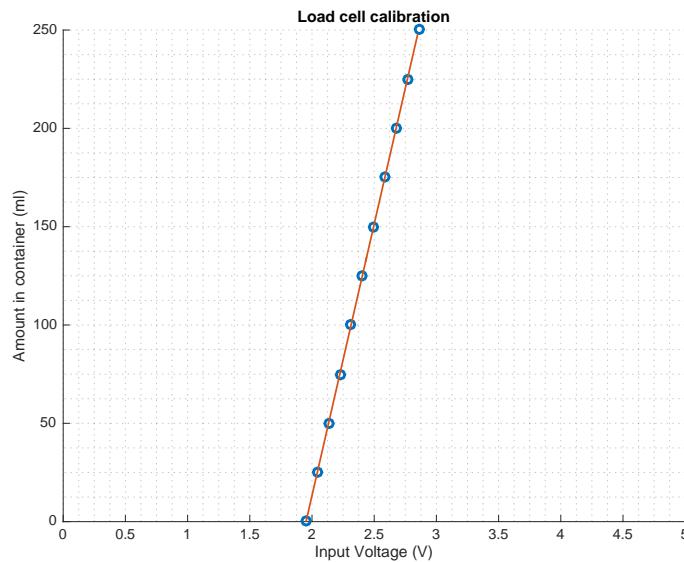
$$a = 276.14 \quad (4.6)$$

$$b = -539.02 \quad (4.7)$$

Den endelige funktion er dermed givet ved:

$$ml = 276.14 * \text{input} - 539.02 \quad (4.8)$$

Figur 4.36 viser den lineære funktion, samt de enkelte data punkter fra tabellen.

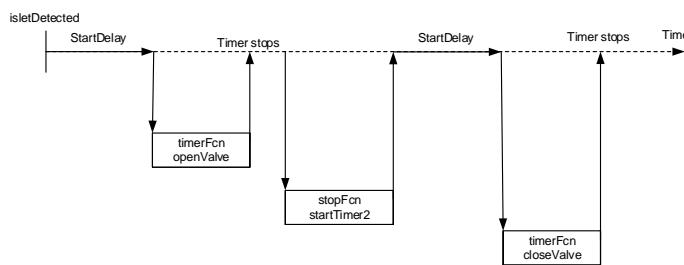


**Figur 4.36.** Kalibrering af load cell

For at reducere støj og mindske følsomheden overfor hurtigere ændringer i indgangsspændingen er der implementeret en midling af de seneste 10 målinger. Dette er med til, at gøre konverteringen mere robust overfor støj.

#### 4.3.4.6 valveControl

I denne funktion styres åbning og lukning af ventilen. Til dette anvendes funktionen `writeDigitalPin` fra Arduino Support pakken, som bruges til at sætte pin'en høj (5V) og lav (0V). For at time hvornår ventilen skal lukke igen anvendes 2 timer objekter. En timer har den fordel, at den kan køre i baggrunden og dermed ikke blokkerer eksekvering af anden kode i programmet. Til et timer objekt kan der knyttes 3 callback funktioner, hhv.: **startFcn**, **TimerFcn** og **stopFcn**. Callback funktionerne specificerer hvad der skal ske ved forskellige tidspunkter. Timerens **startFcn** anvendes ikke ved de 2 timer objekter. I stedet defineres et **startDelay** som specificerer, hvor langt tid der skal gå før timerens egentlige callback funktion eksekveres. Det er dette delay der afgør, hvornår ventilen skal åbne eller lukke igen. Timerens **TimerFcn** sætter altså pin'en høj(5 V) for åbning og lav(0 V) for at lukke den igen. Figur 4.37 viser rækkefølgen for hvornår de enkelte timer funktioner eksekveres.



**Figur 4.37.** Timer synkronisering

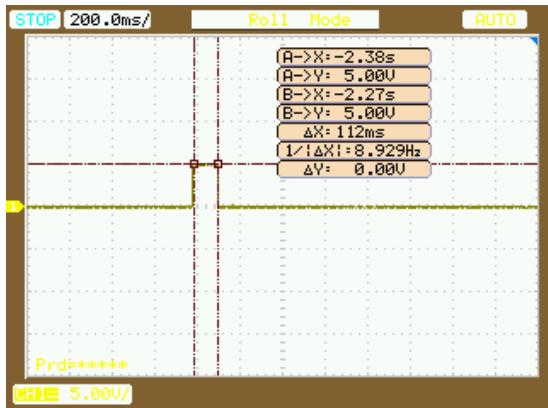
Den første timers opgave er at åbne for ventilen. I denne timers **stopFcn**, startes den anden timer, som lukker ventilen igen. I ventilens design dokument, afsnit 3.3.8, blev det beregnet hvor langt tid der skulle gå fra en ø var detekteret til ventilen skulle åbne (54,475 ms). Ligeledes blev det beregnet, hvor længe ventilen skulle stå åben før dens volumen er tömt igen (144 ms). Til at beregning af timernes **startDelay** er nedenstående formler anvendt:

$$\text{Timer 1 delay} = 54,475\text{ms} - 20\text{ms} - 26\text{ms} = 8,475\text{ms} \quad (4.9)$$

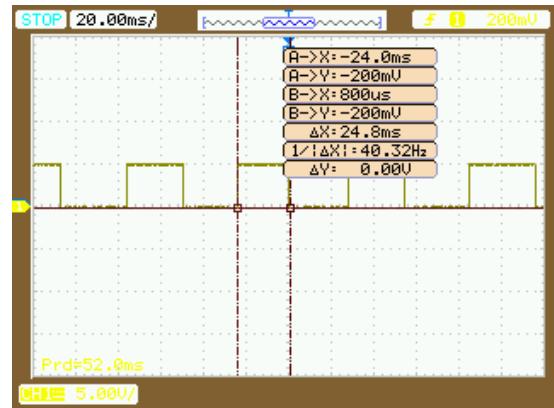
$$\text{Timer 2 delay} = 144\text{ms} - 30\text{ms} - 26\text{ms} = 88\text{ms} \quad (4.10)$$

Fra ventilens datablad (bilag A.1.4) ses det, at åbningstiden for ventilen er op til 20ms og en lukketid på op til 30ms. Disse tider trækkes fra i begge tidsforsinkelser. Herudover trækkes forsinkelsen over seriell forbindelsen mellem Matlab og Arduinoen. Denne tid er fundet via en unittest af ventilen, som vist i figur 4.39.

I figur 4.38 er enhedstesten for ventilen vist. Her er timerne implementeret med ovenstående delays. Som det ses er tiden, hvor ventilen er åben målt til 112 ms. Dette resultat stemmer fint overens med Timer 2 delay på 88 ms (formel 4.10) plus forsinkelsen på 26 ms som det tager at sende en kommando fra Matlab til Arduinoen. Dette giver en total tid på 114 ms, som ligger tæt på de målte 112 ms.



**Figur 4.38.** Tid hvor ventilen er åben. Timer 1 sætter pin'en høj (5V), for herefter at starte Timer 2. Når Timer 2's tid delay er gået sættes pin'en lav (0V). Den endelige tid ventilen er åben er 112 ms.



**Figur 4.39.** Tidsforsinkelse over seriell forbindelsen mellem Matlab og Arduino. På figuren sendes skiftevis en kommando til at sætte pin'en høj (5V) og en kommando til at sætte pin'en lav (0V) til Arduinoen. Periodetiden er på 52 ms.

Nedenstående kode viser, hvordan timerne er implementeret i Matlab.

```

1 % Create timers - with startDelay (defined in handles.constants)
2 tHigh = timer('StartDelay',handles.constants.tHighDelay);
3 tLow = timer('StartDelay',handles.constants.tLowDelay);
4
5 % Assign timerfunction - to set pin HIGH (5V) to open valve and LOW (0V)
6 % to close valve
7 tHigh.TimerFcn = @(x,y) a.writeDigitalPin(handles.constants.valvePin,1);
8 tLow.TimerFcn = @(x,y) a.writeDigitalPin(handles.constants.valvePin,0);
9
10 % StopFcn - starts tLow when tHigh is done executing
11 tHigh.StopFcn = @(x,y)start(tLow);
12
13 % Start first timer
14 start(tHigh)
15
16 % isletDetected boolean is set to false
17 handles.isletDetected=false;

```

Implementeringen er vedlagt i bilag A.2.

#### 4.3.4.7 pumpControl

Funktionen til styring af pumpen anvendes funktionen `writePWMDutyCycle` fra Arduino support pakken. Som input til denne funktion angives pin'en og PWM værdien, som en værdi mellem 0 og 1. Som standard er pumpen sat til en flow hastighed på 50 ml/min, hvilket er svarer til en PWM værdi på 0.9.

#### 4.3.4.8 exportData

Denne funktion gemmer data omkring sorteringscyklussen, som en .csv fil. Opbygningen af filen er nærmere beskrevet i kravspecifikationen (1.4.3.1). Til at gemme filen anvendes Matlab funktionen `writetable`. Da data om sorteringen er gemt, som et struct i handles konverteres structet til en tabel med funktionen `struct2table`. Som filnavn anvendes starttiden for sorteringscyklussen. Denne værdi hentes fra handles.

Når filen er gemt informeres operatøren via en messageboks.

#### 4.3.4.9 areaConverter

Denne funktion bruges til at konvertere minimum og maksimum størrelserne defineret i systemets indstillingerne til et minimum og maksimum areal i pixels. Indstillingsværdierne der er angivet i  $\mu\text{m}$  skal konverteres til antal pixels. Da billederne er genereret er det ikke målt, hvor mange  $\mu\text{m}$  der er pr. pixel, derfor er det implementeret, at 10  $\mu\text{m}$  svarer til 1 pixels på billedet. Nedenstående formel viser, hvordan konverteringen fra  $\mu\text{m}$  til pixels er bestemt:

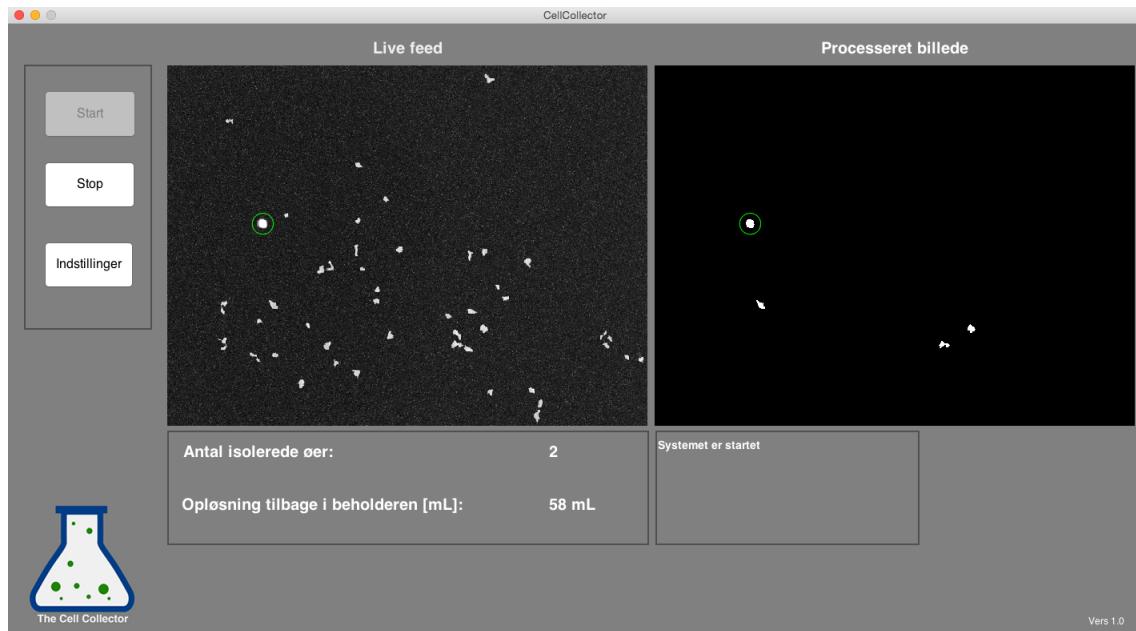
$$\text{Areal i pixels} = \left( \frac{\text{Diameter i } \mu\text{m}}{\frac{2}{10}} \right)^2 * \pi \quad (4.11)$$

En konvertering med en diameter på 100  $\mu\text{m}$  giver dermed et areal på 78 pixels. Ud fra data om øerne i billederne har den mindste ø et areal på 100 pixels.

### 4.3.5 GUI

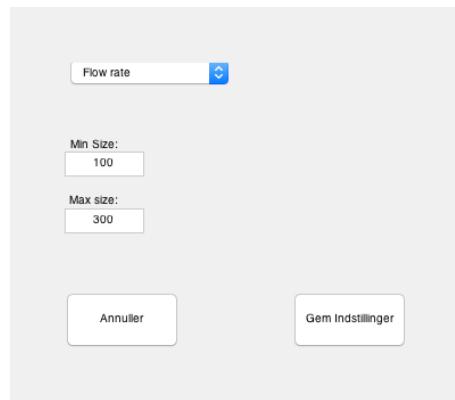
Dette afsnit indeholder en oversigt over hvordan de forskellige GUI vinduer er implementeret.

#### 4.3.5.1 Hovedvindue



*Figur 4.40.* Hovedvinduet på GUI

#### 4.3.5.2 Indstillingsvindue



*Figur 4.41.* Indstillingsvindue på GUI

## 4.4 Integrationstest

Til sidst i implementeringsfasen udføres en integrationstest, som er udført i literationer hver gang en del har været testet. Derfor er systemet langsomt blevet samlet både hardware og software mæssig.

### 4.4.1 Software

Dette afsnit beskriver hvordan softwaren er testet i implementeringsfasen. I gennem udviklingen af de enkelte funktioner er der løbende testet om funktionerne giver de forventede outputs, som beskrevet i designdokumentet. Til dette er der anvendt breakpoints. Matlabs *tic/toc* funktion er også anvendt til at sikre eksekveringen af tidskritiske funktioner ikke tager for langt tid.

Til at teste softwarens interface med Arduinoen er LED'er brugt til at koble på de pinne hvor ventil, pumpe og kameralys skal monteres. Dette gav mulighed for visuelt at se om pinnene på Arduinoen blev sat høj (5V) og lav(0V) som forventet. Herudover blev et potentiometer brugt som erstatning til vægtcellen. Dermed kunne indgangsspændingen manuelt justeres og det kunne dermed visuelt observeres at beholderens indhold blev ændret.

Til sidst i integrationstesten blev de egentlige hardware komponenter koblet på Arduinoens pinne og det blev testet om softwaren fortsat kørte efter hensigten.

### 4.4.2 Hardware

I implementeringsfasen er hardwaren testet vha af generelle metoder ved brug af multimeter og oscilloskop. Derudover er der udarbejdet simpelt testkode til at sikre at hardwaren virker, inden det er implementeret til resten af systemet. Til sidst i hardwaren er dele integreret i et færdigt print.

# Litteratur

---

**Arduino.** Arduino. *Microcontrollerboard— Arduino, UNO.* URL <https://www.arduino.cc/en/Main/ArduinoBoardUno>. [Online; accessed 30-October-2015].

**baslerweb.** baslerweb. *White paper — Camera selection, How can I find the right camera for my image processing system?* URL [http://www.baslerweb.com/media/documents/BAS1408\\_White\\_paper\\_Camera\\_Selection\\_EN.pdf](http://www.baslerweb.com/media/documents/BAS1408_White_paper_Camera_Selection_EN.pdf). [Online; accessed 30-October-2015].

**Ebay, a.** Ebay. *Vægtcelle- Ebay,*. URL <http://www.ebay.com/itm/281311660424>. [Online; accessed 30-October-2015].

**Ebay, b.** Ebay. *Pumpe- Ebay,*. URL <http://www.ebay.com/itm/6V-Peristaltic-Pump-Dosing-Water-Pump-DC-Motor-Tube-For-Aquarium-Lab-Analytical-/131367703927?hash=item1e96201577>. [Online; accessed 30-October-2015].

**Ebay, c.** Ebay. *Microscope Kamera- Ebay,*. URL <http://www.ebay.com/itm/2MP-800X-USB-Microscope-Digital-Mikroskop-Endoskop-Lupe-Kamera-CMOS-Kamera-/400736026036?hash=item5d4dba7db4:g:aygAAOSwu4BVwegS>. [Online; accessed 07-November-2015].

**Farnell, a.** Farnell. *Microscope AVEN ZIPCODE kamera- Farnell,*. URL <http://dk.farnell.com/aven/26700-300/zipscope-usb-digital-microscope/dp/2098929>. [Online; accessed 07-November-2015].

**Farnell, b.** Farnell. *Farnell — Kamera, microscope digital.* URL <http://dk.farnell.com/duratool/bw788/microscope-digital-usb-25x-200x/dp/2319420>. [Online; accessed 30-October-2015].

**Hambley, 2011.** Allan R. Hambley. *Electrical Engineering, 5th edition.* ISBN: 978-0-13-215516-8, Paperback. PEARSON, 2011.

**Matlab, 2015.** Matlab. *Least Square Fitting.* <http://se.mathworks.com/help/curvefit/least-squares-fitting.html>, 2015. Downloadet: 10-12-2015.

**Perlovsky, 2009.** Leonid Perlovsky. *Advanced Educational Technologies, 5th edition.* ISBN: 978-960-474-092-5, Paperback. WSEAS Press, 2009.

**Thomas, 2012.** Roland E. Thomas. *The analysis and design of linear circuits, 7th edition.* ISBN: 978-1-118-06558-7, Paperback. John Wiley and sons, INC., 2012.

**Webster, 2010.** John G. Webster. *Medical instrumentation, 4th edition.* ISBN: 978-0471-67600-3, Paperback. John Wiley and sons, INC., 2010.

**Rettelser**

Note: Tjek lige op på det her . . . . .	61
Note: Tjek lige op på det her . . . . .	62

# Bilag A

---

## A.1 Datablade

A.1.1 INA114

A.1.2 L293D

A.1.3 L5-W55N-BVW

A.1.4 161T031

## A.2 Matlab kode

Alt Matlab kode er vedlagt som .m filer i mappen: Matlab kode. Alle afsnitoverskrifter i dokumentationen er navngivet efter den pågældende .m fil.

## A.3 Arduino Testkode

A.3.1 Kode til enhedstest til vægtcelle.pdf

A.3.2 Kode til enhedstest til pumpe.pdf

A.3.3 Kode til enhedstest til ventil.pdf