NTNU
Norwegian University of
Science and Technology

RoeBot

The roe picking robot

BACHELOR THESIS 2018
Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Automation Engineering IE303612

Yngve Brathaug(10031)
Per Espen Aarseth(10023)
Kristoffer Hildrestrand(10040)
Kristian André Lilleindset(10002)

Supervisor 1: Ottar L. Osen(NTNU)
Supervisor 2: Ibrahim A.Hameed(NTNU)

Pages / Appendix
116 / 290

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Ottar L. Osen, Ibrahim A. Hameed

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒ **ja**	☐ **nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut)	☐ **ja**	☒ **nei**
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐ **ja**	☐ **nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐ **ja**	☒ **nei**

Dato: 31.01.2018

Preface

In a time where the demand for salmon is high and increasing, and the industry is looking for streamlining the production. Monitoring and removing dead roe is a resource consuming operation in the hatching process that is today performed manually. The thesis presented elaborates the development of a robot prototype automating this process, with the intention of laying a foundation of ideas and solutions to build further on.

This bachelor thesis is written at NTNU campus Aalesund for NTNU as part of the automation study program during the time period January to June 2018. Four persons have contributed to this thesis. Three have a practical background with a certificate of apprenticeship in automation and one member has a theoretical background from general studies. We would like to inform the reader, that a general understanding of engineering and automation is required to fully understand the content.

Acknowledgment

We would like to thank all contributors who have helped us during this project, and especially we would like to thank:

- Our supervisors Ottar L. Osen And Ibrahim A. Hameed for guidance throughout the project.
- Arne Styve - at NTNU for guidance with the programming.
- Anders Sætersmoen - at NTNU for the help with ordering parts and lending equipment.
- Alvestad Automation - for advices to the automated solution.
- Jan Sunde - at Møreforsking AS for sharing knowledge about aquaculture.
- Marine Harvest AS at Ytre Standal for welcoming us to their facilities and sharing knowledge about fish farming.
- Optimar A/S - for the electrical cabinet.

Terminology

Consepts:

Actuator Component creating motion.

Binarizing Transformation of a colored image to a binary image.

Biodegradable Capable of being decomposed rapidly by action of microorganisms.

Cartesian coordinate system Coordinate systems with three degrees of freedom, commonly known as X, Y and Z.

Day-degrees is the summed amount of degrees within a number of day.

Degrees of freedom is the number of independent joints allowing the robot to move in several directions.

End-effector Component at the end of a robotic arm.

Extrude To form a material with a desired cross section by forcing it through a die.

Heuristic A strategy or approach which can be applied to solve complex problems.

Image analysis Information extraction from images.

Roe Fish egg.

Acronyms:

ANN Artificial Neural Networks

BLOB Binary Large Object

BOM Bill Of Material

CV Computer Vision

DOF Degrees Of Freedom

FOV Field Of View

FTDI Future Technology Devices International

GUI Graphical User Interface

HMI Human Machine Interface

I/O Input and Output

I^2C Inter-Integrated Circuit

OS Operating System

PWM Pulse Width Modulation

RMS Root Mean Square

TCP Tool Center Point

TSP Traveling Salesman Problem

TTL Transistor to Transistor Logic

USB Universal Serial Bus

Executive summary

Salmon is today a big part of the food commodities traded from aquaculture industry. However, there are several challenges with the industry today, and increasing yield and cutting costs throughout the supply chain is essential. Salmon roe is stored in trays and has a mortality rate which corresponds to millions of dead roe yearly. When roe dies, it will rot which in turn pollutes the surroundings and harms the remaining roe. Therefore, it is vital to remove the dead roe to minimize wastage. Today, they are removed by hand, which is a costly process that introduces a lot of stress to the fry and roe.

The aim of this thesis is to develop a robot as a proof of concept, for the optimization of early stage fish farming. In order to perform this task, the robot will require a high level of precision and accuracy. Furthermore, the technique aims to reduce the amount of stress the roe is exposed to as this can lead to abnormal growth.

The results left us confident that we have proved the concept, as well as creating a substantial foundation for a full-scale operation in an industrialized context. This project shows that locating and removing dead roe is possible by the use of robots and automated systems, and that the solutions and technology presented in this thesis can be used to reduce the cost in salmon production.

Contents

Preface . . . iii
Acknowledgment . . . iv
Terminology . . . v
Executive summary . . . vii

1 Introduction **1**
1.1 Background . . . 1
1.2 Project introduction . . . 1
1.3 Aim and objectives . . . 1
1.4 Report content . . . 2

2 Theory **4**
2.1 Roe in aquaculture . . . 4
2.1.1 Roe mortality . . . 5
2.1.2 Abnormal growth . . . 5
2.1.3 Death rate . . . 5
2.2 Industrial robots . . . 6
2.2.1 Cartesian robot . . . 6
2.2.2 SCARA robot . . . 6
2.2.3 Delta parallel robot . . . 7
2.3 Robot kinematics . . . 7
2.4 Linear motion . . . 8
2.4.1 Belt-driven actuators . . . 8
2.4.2 Ball screw driven actuators . . . 8
2.4.3 Rack and pinion driven actuators . . . 8
2.5 Electric motors . . . 9
2.5.1 Stepper . . . 9
2.5.2 Servo . . . 11
2.5.3 Motor driver . . . 11
2.6 Venturi vacuum generator . . . 11
2.7 Communication protocols . . . 12
2.7.1 Serial . . . 12
2.7.2 I2C serial protocol . . . 12
2.8 Machine vision . . . 12
2.8.1 Camera . . . 13
2.8.2 Surroundings . . . 14

2.8.3 Data extraction . . . 14
2.9 Image processing . . . 14
2.9.1 Segmentation . . . 15
2.9.2 Morphology . . . 15
2.10 Traveling salesman problem . . . 16
2.10.1 Random tour . . . 16
2.10.2 Nearest neighbor . . . 16
2.10.3 Genetic algorithm . . . 17
2.11 Bresenhams algorithm . . . 17

3 Approach **18**
3.1 Project approach . . . 18
3.2 Robot . . . 19
3.2.1 Robot selection basis . . . 19
3.2.2 Roe treatment . . . 19
3.3 Review . . . 20
3.4 Testing . . . 20
3.4.1 Illumination test . . . 20
3.4.2 Image processing testing . . . 20
3.4.3 Communication testing . . . 21
3.4.4 Suction testing . . . 21
3.4.5 Electrical cabinet testing . . . 21
3.4.6 Robot testing . . . 21

4 Materials **22**
4.1 Data collection . . . 22
4.2 Robot selection . . . 22
4.3 Motion method selection . . . 23
4.4 Motor selection . . . 24
4.4.1 Motors for horizontal motion . . . 24
4.4.2 Motors for vertical motion . . . 26
4.5 Motor driver selection . . . 27
4.5.1 DRV8825 . . . 27
4.5.2 DM556 . . . 27
4.5.3 KeyeStudio driver shield . . . 27
4.6 Vacuum source selection . . . 27
4.7 Choice of sensors . . . 28
4.7.1 Mechanical endstop . . . 28
4.7.2 Line detectors . . . 28
4.7.3 Optical sensors . . . 28
4.7.4 Incremental rotary encoders . . . 28
4.8 Choice of controllers . . . 29
4.8.1 Odroid-XU4 . . . 29
4.8.2 Supplementary controllers for I/O . . . 29
4.8.3 Horizontal system controller . . . 29

4.8.4 Vertical system controller . . . 29
4.9 Communication protocol . . . 30
4.9.1 Connecting the controllers via USB . . . 30
4.10 Machine vision . . . 30
4.10.1 Camera selection . . . 30
4.10.2 Illumination . . . 31
4.11 Construction . . . 31
4.12 Electrical cabinet and components . . . 31
4.12.1 Power supplies and ampere calculation . . . 31
4.12.2 Fuses . . . 32
4.13 Hatchery imitation . . . 32
4.13.1 Test medium . . . 33
4.13.2 Rack system . . . 33
4.14 Software and libraries . . . 34
4.14.1 Software . . . 34
4.14.2 Libraries . . . 35

5 Design **36**
5.1 Main frame . . . 37
5.1.1 Designed parts . . . 37
5.2 Horizontal system . . . 38
5.2.1 Designed parts . . . 38
5.2.2 Horizontal system electrical layout . . . 41
5.3 Vertical system . . . 42
5.3.1 Designed parts . . . 42
5.3.2 Vertical system electrical layout . . . 45
5.4 Venturi vacuum generator . . . 45
5.5 Electrical cabinet . . . 46
5.5.1 Placement of controllers . . . 46

6 Implementation **48**
6.1 System overview . . . 48
6.1.1 1 - Human machine interface (HMI) . . . 48
6.1.2 2 - Computer / processing . . . 49
6.1.3 3 - Robot . . . 50
6.1.4 3.1 - Vertical system . . . 51
6.1.5 3.2 - Horizontal system . . . 51
6.2 Robot Program . . . 51
6.2.1 Program Structure . . . 51
6.2.2 Operation flow . . . 52
6.2.3 Operation state flow . . . 53
6.2.4 Java classes . . . 53
6.2.5 Arduino . . . 55
6.3 Robot calibration . . . 57
6.3.1 Vertical robot calibration . . . 57

6.4 Robot motion . . . 58
6.4.1 Distance transformation . . . 58
6.4.2 Vertical motion . . . 58
6.4.3 Horizontal motion . . . 59
6.5 Tool center point . . . 59
6.5.1 IR-beam offset . . . 59
6.5.2 Magnet offset . . . 60
6.5.3 Camera offset . . . 61
6.6 Machine vision . . . 63
6.6.1 Camera . . . 63
6.6.2 Environment . . . 63
6.7 Image processing development . . . 66
6.7.1 Roe . . . 66
6.7.2 Roe imitation . . . 69
6.8 Path optimization . . . 72
6.8.1 Nearest neighbor . . . 72
6.8.2 Genetic algorithm . . . 72
6.9 Tray handling mechanism . . . 74
6.10 Roe removal . . . 75
6.11 Relays . . . 75
6.12 Emergency stop . . . 76
6.13 Graphical user interface . . . 76

7 Reviews **79**
7.1 Venturi generator testing . . . 79
7.2 Timing belt stretch test . . . 79
7.3 Illumination test . . . 81
7.4 Electrical cabinet testing . . . 82
7.5 Vertical system testing . . . 82
7.6 Horizontal system testing . . . 83
7.7 I2C as communication protocol . . . 83
7.7.1 I2C communication via Arduino . . . 83
7.7.2 I2C communication via Odroid . . . 83
7.7.3 Voltage-level translator . . . 84
7.7.4 I2C communication on robot construction . . . 84
7.7.5 Changing the communication protocol to Serial . . . 87
7.8 Design reviews . . . 87
7.8.1 First iteration . . . 88
7.8.2 Second iteration . . . 88
7.8.3 Third iteration . . . 89
7.8.4 Vacuum generator review . . . 89
7.9 Software reviews . . . 90
7.9.1 First iteration . . . 90
7.9.2 Second iteration . . . 90
7.9.3 Third iteration . . . 91

7.9.4 Fourth iteration . . . 91

8 Final results **92**
8.1 How the robot works . . . 93
8.2 Workspace of the robot . . . 93
8.3 Vertical system motion . . . 94
8.4 Horizontal system motion . . . 94
8.4.1 Horizontal system movement . . . 95
8.4.2 Horizontal system accuracy . . . 95
8.4.3 Horizontal system repeatability . . . 95
8.5 Tray handling . . . 96
8.6 Machine vision . . . 96
8.6.1 Image processing result of real roe . . . 96
8.6.2 Image processing of imitated roe . . . 98
8.7 Pattern optimizer . . . 100
8.7.1 Result with 25 destinations data set . . . 101
8.7.2 Result with 47 destinations data set . . . 103
8.8 Removing roe . . . 105

9 Discussion **106**
9.1 Results from testing . . . 106
9.1.1 Motion . . . 106
9.1.2 Rack system and tray handling . . . 106
9.1.3 Image processing . . . 107
9.1.4 Pattern optimizer . . . 107
9.2 Software . . . 107
9.3 Physical structure and design . . . 108
9.4 Improvements to further the prototype . . . 108
9.5 Separations from the prototype to a functional solution . . . 109
9.6 Experiences . . . 109
9.6.1 Planning . . . 109
9.6.2 Division of labor . . . 109
9.7 Data collection . . . 109
9.8 Hypothetical profitability . . . 110

10 Conclusion **111**

Bibliography **112**

Appendix A Project planning **117**
A.1 Pre project . . . 117
A.2 Original project plan . . . 130
A.3 Final project plan . . . 131
A.4 Timesheet . . . 133

Appendix B Marine harvest visit **136**

Appendix C Images from assembling **139**

Appendix D Bill of material (BOM) **151**
D.1 BOM 151
D.2 URL for bill of material 155

Appendix E Electrical drawings **159**

Appendix F Keye Studio V4.0 I/O **176**

Appendix G Timesheet **178**

Appendix H Java Code **181**
H.1 Java Class Diagram 181
H.2 Roe Robot Main 183
H.3 Serial Communication 262
H.4 Image Processing 286
H.5 Path Optimalization 298
H.6 HMI 312

Appendix I Arduino Code **349**
I.1 Horizontal System Code 349
I.2 Vertical System Code 378

Appendix J Matlab code **407**

List of Figures

2.1 Life cycle of salmon 4
2.2 Pipette roe removal 5
2.3 Hatchery rack system 5
2.4 Death of roe from 2015 to 2017 5
2.5 Selection of industrial robots 6
2.6 Forward kinematics 7
2.7 Inverse kinematics 7
2.8 Selection of industrial robots 8
2.9 Full step revolution 9
2.10 Half step revolution 9
2.11 Microstepping and torque 10
2.12 Venturi generator 11
2.13 Camera field of view 13
2.14 Image processing 15
2.15 Morphology erosion and dilation example 16
2.16 GA flow chart 17

4.1 Cartesian robot joints 23
4.2 Timing belts 24
4.3 Horizontal system 25
4.4 Nema 14 stepper motor 25
4.5 Vertical system 26
4.6 Nema 23 stepper motor with brake 26
4.7 Driver shield 27
4.8 Venturi chamber 27
4.9 Chosen sensors 28
4.10 Logitech C920 HD PRO 30
4.11 Illumination 31
4.12 Roe 33
4.13 Roe imitation 33
4.14 Rack imitation 34
4.15 Open drawer with roe 34

5.1 The RoeBot 36
5.2 Main frame 37

5.3 Frame T-connector . . . 37
5.4 Horizontal system . . . 38
5.5 X motor holder . . . 39
5.6 Y-axis motor bracket . . . 39
5.7 Timing belt lock . . . 39
5.8 X direction trolley . . . 40
5.9 End-effector . . . 40
5.10 Horizontal system electrical layout . . . 41
5.11 Vertical system design . . . 42
5.12 Trolley z-direction . . . 42
5.13 Vertical system pulley holder . . . 43
5.14 Vertical system timing pulley . . . 43
5.15 Vertical system pulley radius . . . 43
5.16 Timing belt carriage in place . . . 44
5.17 Timing belt carriage . . . 44
5.18 Bracket for optical sensor . . . 44
5.19 Bracket for optical sensor in position . . . 44
5.20 Vertical system electrical layout . . . 45
5.21 Venturi chamber design . . . 46
5.22 Electrical cabinet . . . 46
5.23 Arduino Nano with driver shield in ventilated box . . . 47

6.1 System overview . . . 48
6.2 1 - Human machine interface . . . 49
6.3 2 - Computer / processing . . . 49
6.4 3 - Robot . . . 50
6.5 Overview of physical connections . . . 50
6.6 Main state flowchart . . . 52
6.7 Operation state flowchart . . . 53
6.8 HMI class and main robot class interaction . . . 55
6.9 Main robot classes and Arduino controllers interaction . . . 55
6.10 Movement of the system . . . 58
6.11 Offset from IR-beams to TCP . . . 60
6.12 Offset from magnets to TCP . . . 60
6.13 Camera TCP offsets x-, y- and z-direction . . . 61
6.14 Image origin to TCP offsets x- and y-direction . . . 61
6.15 Camera placement . . . 63
6.16 Robot with lights . . . 63
6.17 Image layout . . . 64
6.18 Original RGB image . . . 66
6.19 Gray image . . . 67
6.20 Binary image . . . 67
6.21 Eroded binary image . . . 68
6.22 Dilated . . . 68
6.23 Table of centroids from image processing . . . 68

6.24 Original RGB image . . . 69
6.25 Gray image . . . 69
6.26 Manual binarizing . . . 70
6.27 Adaptive binarizing . . . 70
6.28 Eroded image . . . 71
6.29 Dilated image . . . 71
6.30 Random pattern optimizer flow chart . . . 72
6.31 Genetic algorithm flow chart . . . 73
6.32 Tray handling mechanism . . . 74
6.33 Magnet switch . . . 74
6.34 Vacuum tube mounted to TCP . . . 75
6.35 Suction process diagram . . . 75
6.36 Graphical user interface - calibration . . . 76
6.37 Graphical user interface - main frame . . . 77
6.38 Light regulation panel . . . 78

7.1 Timing belt stretch test . . . 80
7.2 Timing belt stretch test snapped . . . 80
7.3 Timing belt stretch test results . . . 80
7.4 9mm HTDM5 belt snapped . . . 81
7.5 Illumination [0, 3, 4] . . . 82
7.6 Illumination [0, 2, 4] . . . 82
7.7 SDA line with drivers running . . . 85
7.8 SDA line without noise . . . 85
7.9 Master trying to initiate transfer with slaves . . . 87
7.10 Bracing of y-axis . . . 88
7.11 Optical sensor holder . . . 89
7.12 Old y-axis pulley holder . . . 89
7.13 New y-axis pulley holder. . . . 89
7.14 Venturi chamber design early version . . . 90

8.1 The RoeBot . . . 92
8.2 Robot workspace dimensions x and z plane . . . 93
8.3 Robot workspace dimensions x and y plane . . . 94
8.4 How the horizontal system moves . . . 95
8.5 Horizontal system accuracy . . . 96
8.6 Pre processing real roe . . . 97
8.7 Post processing real roe . . . 97
8.8 Image processing real roe confusion matrix . . . 98
8.9 Pre processing imitated roe . . . 99
8.10 Post processing imitated roe . . . 99
8.11 Image processing imitation roe confusion matrix . . . 100
8.12 Result of GA . . . 101
8.13 Original tour size 25 . . . 101
8.14 Tour optimized with nearest neighbor. Size 25 . . . 102

8.15 Tour optimized with GA. Size 25 102
8.16 Result of GA 103
8.17 Original tour. Size 25 104
8.18 Tour optimized with nearest neighbor. Size 25 104
8.19 Tour optimized with GA. Size 25 104
8.20 Removing roe 105

A.1 Original project plan 130

List of Tables

4.1 Arduino table 29
4.2 Theoretical current calculation 32
4.3 Total current calculation 32
4.4 Fuses current and voltage rating 32

6.1 Status - Answer from the Arduino controllers 56
6.2 Commando - Tasks for the Arduino controllers to perform 57

7.1 Timing belt stretch test 81
7.2 Vertical system test 82
7.3 Horizontal system test 83

8.1 Image processing result, real roe 97
8.2 Image processing result, roe imitation 99

Chapter 1

Introduction

1.1 Background

In 2016 the fish industry sat a record when it passed twenty kilograms in supply per person worldwide. To this date, fish is one of the most traded food commodities worldwide [1], with half of the total supply originating from aquaculture. The importance of the aquaculture industry is indisputable. Trying to match the need through commercial fishing would lead to over-fishing and extinction of entire species [2]. This business represents sources of food, livelihoods, and earnings for millions of people around the world. To ensure profitability and to keep the production aligned with the increasing demand, effectivization and automatization of the fish industry is needed [3].

1.2 Project introduction

NTNU in Aalesund seeks to prove that an autonomous industrial robot can optimize the first part of the production in fish farms. Because the demand of fish is high, the supply of roe must be equivalent. A problem in the fish industry is roe dying in its premature phase, putting the surrounding ones in danger of pollution. Today the solution for saving the surrounding roe, is people removing the dead ones manually, a time consuming and costly operation. A solution to this problem can be an automated roe picking robot. Because the roe is small and vulnerable, the robot must be able to operate gently with high accuracy. In this paper, the emphasis is on design, image processing, and finally removing dead roe. During consideration of different choices, the priority will be in low cost and handle ability.

1.3 Aim and objectives

The aim for this project is to build a prototype of an autonomous industrial robot as proof of concept for future automated solutions in the fish hatching process. The robot will operate on a rack system, where the roe substitute is stored. The main task will be to detect and remove dead roe as efficient as possible. Core objectives for creating this robot is mentioned below.

- Design and develop a prototype robot suited to operate in a hatchery-like facility. It needs to have high enough accuracy to remove dead roe substitutes. During development, safety will

be taken into account.

- Program the robot so it can position it self within a three dimensional coordinate system.
- Develop software for locating and removing dead roe by the use of machine vision and image processing.
- Develop a system limiting the amount of stressful exposure for the roe by gentle handling.
- Assemble the prototype and test it according to specified requirements.
- Design and implement a user interface that can easily be operated by instructed personnel.

1.4 Report content

This technical report does not follow the recommended standard layout for a bachelor thesis. Instead of six chapters, this report has ten. To make the report content orderly to our work, we factored the elementary sub-chapters into own chapters. The deviations from the standard layout includes dividing the materials and method chapter into two chapters named *Approach* and *Materials*. The result-chapter has been separated into four chapters: *Design*, *Reviews*, *Implementation*, and *Final results*.

Chapter 2 - Theory contains the theoretical content needed to make decisions and solve problems later on in the report.

Chapter 3 - Approach is a description of what is deemed as necessary specifications as well as how to approach tasks and problems to develop the robot. At last, the tests to perform on individual parts of the robot are described.

Chapter 4 - Materials contains the materials, components and information used for creating the prototype in this project. At first critical information is gathered of what complications and requirements this kind of robot would be imposed to. Then what kind of robot to be built is decided, as well as all the components it will be composed of.

Chapter 5 - Design elaborates the prototype design, and how complex parts were engineered.

Chapter 6 - Implementation elaborates on how the system controlling the robot is implemented and its functions. Also explains how components were installed, and how problems and tasks were solved.

Chapter 7 - Reviews is a chapter concerning the tests that were performed on the individual systems of the robot. The reviews elaborates how the prototype changed during the project, due to results of the tests and reviews performed.

Chapter 8 - Final results explains how the finished product and individual systems functions and performs.

Chapter 9 - Discussion discusses the results, problems encountered along the way, improvements and what separates this prototype from a final product.

Chapter 10 - Conclusion contains the conclusion from the projects implementation and answers the problem statement.

Chapter 2

Theory

This chapter contains the theoretical basis that is needed for making decisions throughout the project.

2.1 Roe in aquaculture

Production of fish is a process that consists of the phases illustrated in figure 2.1. It takes about two to three years from the hatching process of roe to the fish is fillet. The cycle starts in rack systems where the roe is stored in trays. The eggs, which has a size of 5.3 to 6.7 millimeters, are stored for about five hundred day-degrees before it hatches and becomes fry. The fry grows a rucksack on their abdomen and will from this have nourishment for the next four to six weeks. Fry develops into the next phase of the cycle, which is called parr, and is then moved to a tub. In the tub it gets vaccinated and held for ten to sixteen months while the parr gradually grows into smolt and is moved to sea cages. When the fish has a weight of four to six kilograms, it will be taken out of the cage, filleted, and sold to stores worldwide [4].

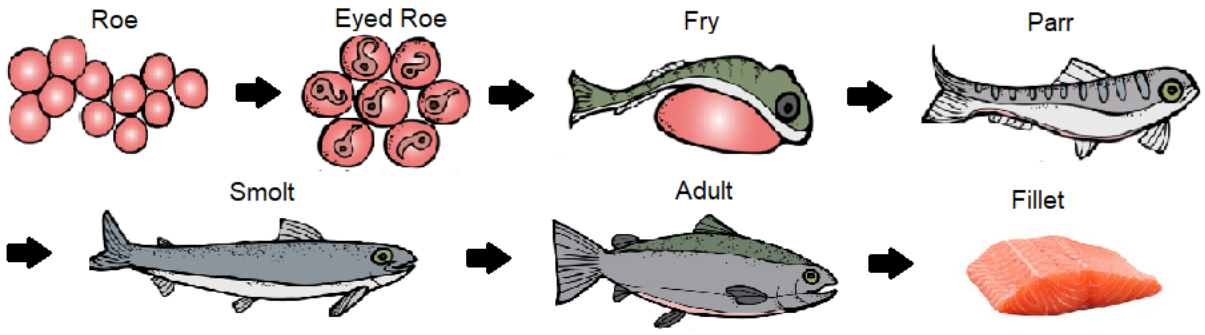


Figure 2.1: Life cycle of salmon
[5]

2.1.1 Roe mortality

Death amongst roe occurs. When this happens, the dead roe will rot and pollute its surroundings. It is thus essential that the dead roe is removed as quickly as possible [6, p. 23]. A time-consuming job, which requires human resources and interaction. The removal itself is difficult as it takes a calm presence and accuracy to complete this. The eggs can withstand little, and are vulnerable to any physical impact at this part of the process. In addition, they are sensitive to brightness, noise and mechanical influence [6, p. 25]. Today, different types of removal techniques are being used, the most common is pipette, as shown in figure 2.2. The person responsible for the removal, opens the tray systems manually and searches for dead eggs(figure 2.3). The dead eggs are easily located as they have a distinct white color (Appendix B).

Figure 2.2: Pipette roe removal
[7]

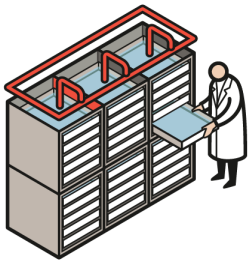


Figure 2.3: Hatchery rack system
[8]

2.1.2 Abnormal growth

Abnormal growth is a consequence of stressful impacts in the early stages of the fish production. When this occurs the stressed fry swims faster and drains the nutrition sack earlier than it is supposed to, which leads to a nutritional imbalance between the stressed and non-stressed fry. The rapid growth of the stressed fry leads to it missing vital building blocks in its internal organs and deprives them of a healthy life. Adult fish in the aquaculture industry should be equal in size and health. To avoid stress, a necessity is to control the environment of the batch regarding noise, light, and motion [6].

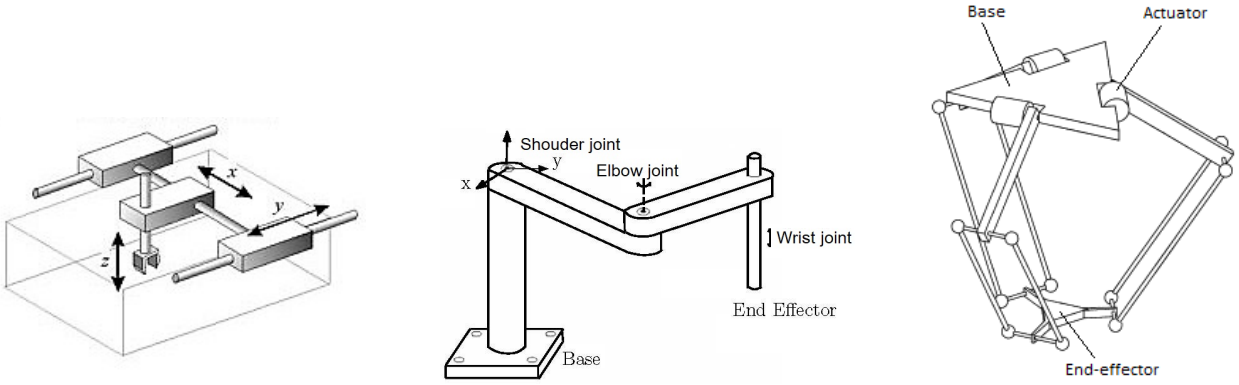
2.1.3 Death rate

Statistics from 2015 to 2017 imply that production of roe the two previous seasons has been higher than ever before. Figure 2.4 shows the survival statistics fifty-eight hatcheries in Norway during these two seasons. Out of the 56 248 932 roe delivered to hatcheries during 2016-2017, 93.5% survived. Thus, 6.5%, equal to 3 556 181, died [9]. Of these 6.5%, scientists estimate that around 10 to 15% die as a consequence of pollution from other dead roe [10].

Figure 2.4: Death of roe from 2015 to 2017
[4, p. 5]

2.2 Industrial robots

An industrial robot is a mechanical and programmable unit that can perform repetitive tasks. These robots possess at least three degrees of freedom, left-right, up-down and forward-backward. In addition, to move a robot freely in a three-dimensional space, three degrees-of-freedom need to be added. These are called yaw, pitch and roll, providing rotation around the x, y, and z-axis [11]. Robots achieve motion using joints controlled by actuators. Two types of joints can be applied. The first is a prismatic joint (P) which provides linear motion, and the other is a revolute joint (R) that provides rotational motion. Every robot has a base and a free end. The base works as a fixed link to which all the other links are relative to. A free end is where the end-effector or a tool can be assembled [12]. Three types of industrial robots often used for pick and place are described beneath.



(a) Cartesian robot [13] (b) Scara robot [14] (c) Delta robot [15]

Figure 2.5: Selection of industrial robots

2.2.1 Cartesian robot

Unique for cartesian robots is that they have three independent prismatic joints (PPP), which leads to a linear kinematic. They are constructed to move in linear motion with high repeatability through the full workspace. Although this robot often involves a considerable floor space, it reaches a large area within this space. Therefore it considered to be workspace efficient. A drawback, cartesian robots are one of the slower manipulators on the marked. A basic illustration of the robot is shown in figure 2.5a [16].

2.2.2 SCARA robot

When SCARA robots are defined they often get compared to a human arm. These robots have both a ”shoulder” and ”elbow” joints along with a ”wrist”. The arm is manipular and contains two revolute joints and one prismatic (RRP), as seen in figure 2.5b. The SCARA is ideal for applications that require a fast, repeatable and articulate point-to-point movement. On the downside, they are tall and restrict a large volume [16].

2.2.3 Delta parallel robot

Delta parallel robots are one of the most successful parallel robot designs ever made [15]. It consists of three links independently connected to actuators at the base. These actuators can either provide revolute or linear motion depending on the configuration. Benefits with the delta robot is the high payload, speed, and accuracy. Consequently, it is a good choice for pick and place operations. A disadvantage is the small workspace relative to the large size of the robot. A common configuration of a delta robot is shown in figure 2.5c [16].

2.3 Robot kinematics

Kinematics is the study of motion of a robot regarding position, velocity, and acceleration of the manipulator joints. Kinematics does not consider the force or torque required for generating motion. It is used for transforming a coordinate system to another, and back. Figure 2.6 and 2.7 illustrates the transformation from a joint coordinate system to a coordinate system representing real-world coordinates. A joint coordinate system defines each joint with its position or angle. The real world coordinate system uses Cartesian coordinates and has its origin at the base of the robots frame [17]. In robot kinematics there are two main problems that needs to be solved, which is forward kinematics and inverse kinematics.

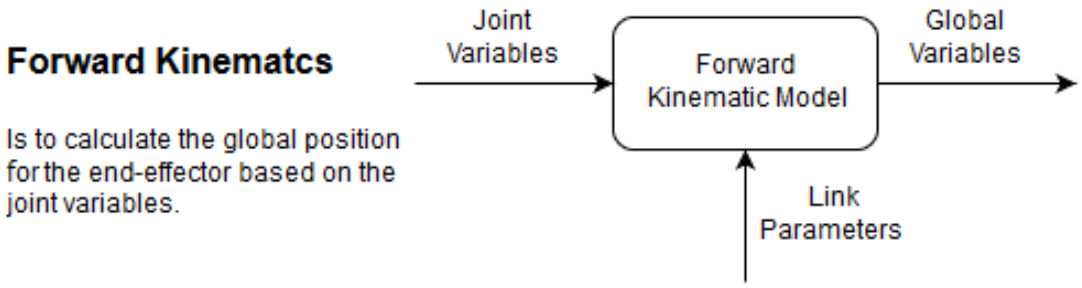


Figure 2.6: Forward kinematics

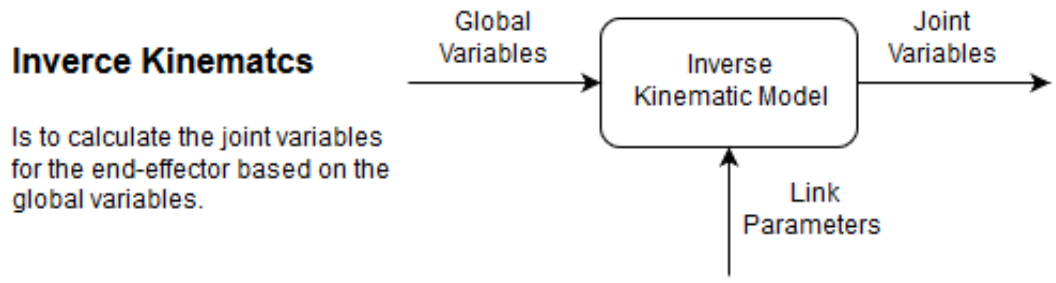


Figure 2.7: Inverse kinematics

2.4 Linear motion

Some electrical machines require linear movement to work efficient. Because most electrical actuators provide rotational motion, it is necessary to convert rotational motion into accurate linear motion. There are several ways of achieving this conversion. Some of the commonly used techniques are either by belt-driven actuators, by ball screw driven actuators or by rack and pinion driven actuators. The three actuator implementations, all apply linear motion by rotary motors, but they differ in strengths and limitations [18]. Accuracy provided by the different methods depends on the actuator providing the revolution motion.

(a) Belt driven actuator [19]

(b) Ball screw actuator [20]

(c) Linear gear actuator [21]

Figure 2.8: Selection of industrial robots

2.4.1 Belt-driven actuators

Belt-driven actuators (figure 2.8a) are often used in applications with movement of low weight components over longer distances. They use a timing belt between two or more circular pulleys and a clamp for moving a carry along the motion of the timing belt. Motion is applied by turning one of the pulleys, thus moving the belt. The distance traveled by the trolley is relative to the size of the circular pulley where motion is applied [22]. The distance a belt moves is given by the pulley pitch circumference using the pitch diameter [23].

2.4.2 Ball screw driven actuators

Ball screw driven actuators (figure 2.8b) are used in applications that require heavy lifting and accurate positioning. The actuator is composed by a lead screw and a ball nut. The lead screw is fixed with bearings in the machines construction. Motion is applied by rotating the lead screw, causing the ball nut and components connected to it to move along the lead screw [22].

2.4.3 Rack and pinion driven actuators

Rack and pinion driven actuators (figure 2.8c) are suited for applications where heavy systems are moved over long distances. As the torque and speed are easily manipulated by changing the pinion it can also be used for low weight applications. Rack and pinon actuators are implemented by mounting the rack along the direction of movement and the pinion connected to a rotary motor mounted on the machine applying motion to the actuator. The greatest advantage with rack and pinion actuators is the unlimited travel distance that can be created by extending the racks length [24].

2.5 Electric motors

Electric motors are machines that convert electrical energy to mechanical energy [25]. Two types of electric motors widely used in the industrial world are the Stepper motor and Servo motor. They both have qualities in some areas and challenges others. The two motors follow the NEMA-standard, which will tell the size of the motor, but likewise covers the rated current and power [26].

2.5.1 Stepper

The stepper motor moves with a stepping motion and not with constant flow. Figure 2.9 illustrates one full revolution of the motor. Depending on the stepper motor driver, one step can be divided into several smaller steps, this is called microstepping. It works by forcing the motor to rotate a fraction of a full step for each step moved. The size of a microstep is calculated by equation (2.1). A microstepping factor is a natural number like 1, 2, 4, 8, and so on [27]. Microstepping will reduce mechanical noise and give a more gently mechanical actuation. On the downside, the motors holding torque will drop as a function of the microstep number, shown in figure 2.11. Figure 2.10 illustrates the operation of a stepper motor configured with half step microstepping resolution.

Calculating the size of a microstep:

$$m_s = \frac{1}{f_m} \tag{2.1}$$

$m_s =$ Size of a microstep
$f_m =$ Microstepping factor

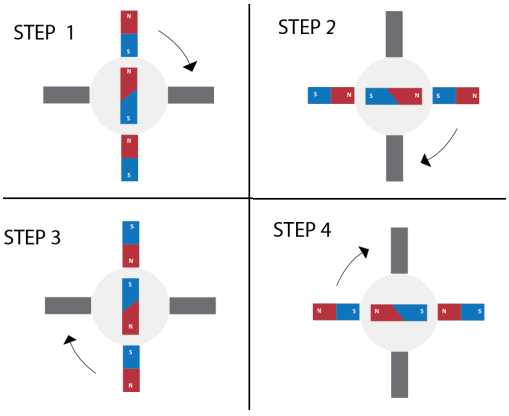


Figure 2.9: Full step revolution

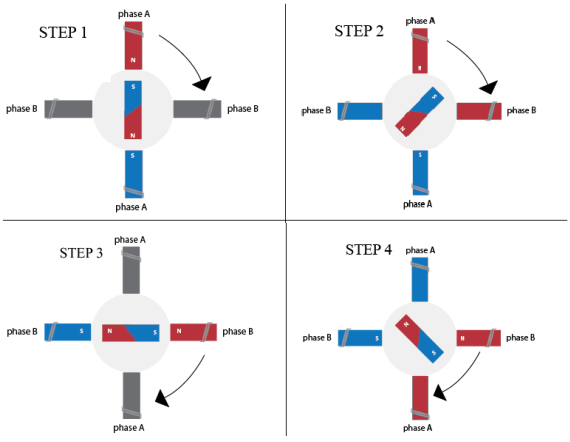


Figure 2.10: Half step revolution

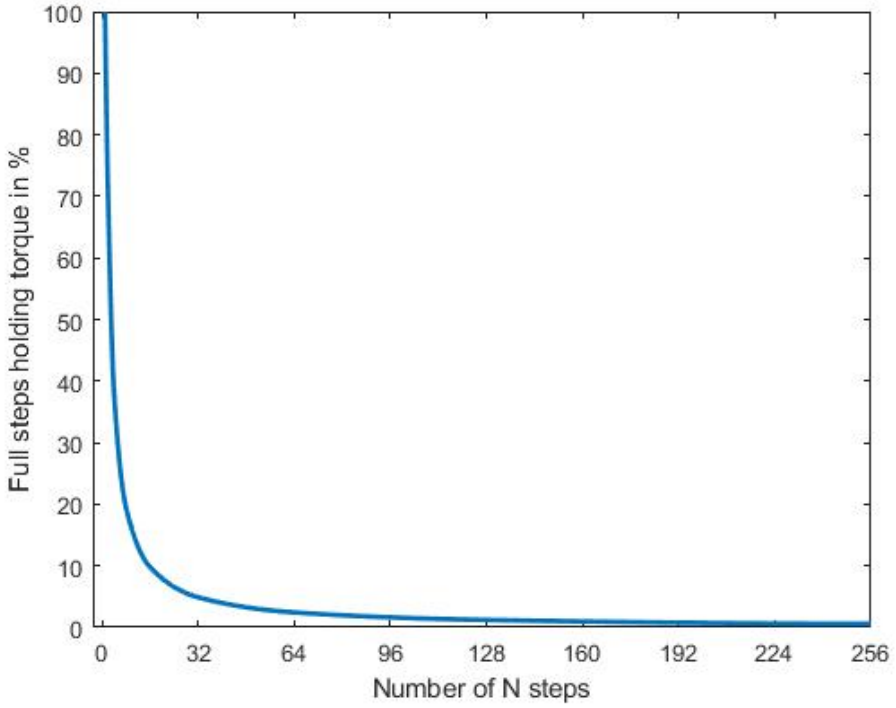


Figure 2.11: Microstepping and torque

Every stepper motor comes with predetermined values that can be found in the motors datasheet. This includes step-angle, rated voltage, rated current and phase numbers. Out of this the other various motor specifications can be calculated. For starters the steps per revolution will be found from the angle of one whole revolution divided with the step angle for one single step:

$$spr = \frac{\frac{360 \quad Degree}{1 \quad revolution}}{\frac{Degree}{Step}} \tag{2.2}$$

Computing the speed, expressed as maximum revolutions per second, is done by dividing seconds per step with steps per revolution.

$$\frac{Rev}{sec} = \frac{V}{L \times 2I_{max} \times spr} \tag{2.3}$$

$I_{max} =$ Max current
$V =$ Applied voltage
$Rev =$ Revolutions
$spr =$ Steps per revolutions
$L =$ Motor inductance

2.5.2 Servo

Servo motors allows for precise control of angular or linear position, velocity and acceleration. It operates on a closed-loop system that use position feedback from an encoder to control motion and position. Servo motors are suited for applications that require high speed as it creates high torque at high speeds [28].

2.5.3 Motor driver

Motor drivers controls the torque, speed, and direction of rotary and linear electric motors. They function by taking a low-current control signal and turning it into a higher-current signal that drives the motor [29].

2.6 Venturi vacuum generator

Venturi vacuum generators operates by the venturi principle. It generates vacuum by sending motive fluid into the venturi chamber through a narrow nozzle. Both compressed gas and liquid, can be used as motive fluid. Because the nozzle has smaller volume than the input port, the motive fluid will have higher speed and lower pressure inside the chamber. This can be derived from the Continuity equation (2.4) and Bernoullis equation (2.5). Figure 2.12 illustrates how the venturi generator works.

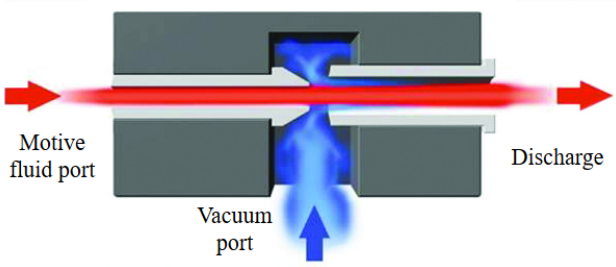


Figure 2.12: Venturi generator
[30]

Continuity Equation:

$$A_1 v_1 = A_2 v_2 \tag{2.4}$$

Bernoulli's Equation:

$$P_1 + \frac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho v_2^2 + \rho g h_2 \tag{2.5}$$

Because the atmospheric pressure is greater than the low pressure on the venturi, an under pressure will be generated at the vacuum port.

2.7 Communication protocols

Communication provides the ability of transmitting data between two or more devices. A protocol in its most basic form, should contain a set of rules of how the communication is going to be started, stopped and transmission of data.

2.7.1 Serial

What characterizes serial communication is that it transmits one bit at a time, compared to parallel communication which sends several bits at a time. The two main types of serial protocols are based on Synchronous and Asynchronous communication.

Synchronous Serial has at least two wires, where one of them is a clock signal wire paired with a data signal wire. This means all the devices follows the external clock and all the transfers will be based on this clock. It makes synchronizing the speeds easier and the protocol more straightforward. Some of the other synchronous protocols are I2C and Serial Peripheral Interface Bus (SPI) [31].

The second type, Asynchronous Serial simply means that data is transferred without an external clock being synced. Because of this, more steps is taken in the protocol to ensure reliable transfer and receiving of data [31].

2.7.2 I2C serial protocol

I2C is a communication protocol based on the Master-Slave principle. The master can communicate with multiple slaves and one slave can communicate with multiple masters(multi-master system). I2C requires two connections - Serial Clock Line(SCL) and Serial Data Line(SDA) where SCL is the clock syncing and SDA the data connection. Sending bits is performed by keeping track of the SCL and setting SDA HIGH or LOW during the SCL pulse. Initiating a data transfer will need to use both of the connections. When initiating a start condition for a packet, the sending master pulls SDA LOW and leaves SCL HIGH. All slave devices are then put in listening mode, and the next sequence of seven bits are the address of the receiving slave device. All other devices then the specified device will discard the incoming data. The slave sends an acknowledgement(ACK) that it received the message, after the address and the *Read/Write* parameter has been sent from the master. This is done by holding the SDA line LOW for one SCL sequence after the address sequence. An indication of the data sequence is sent over the line and eight bits(one byte) is transmitted over the line [32].

2.8 Machine vision

Machine vision is a machines way of solving complex problems without human interaction. This is done by gaining visual information from digital images. All applications that use information gained from image processing in order to make decisions, uses machine vision [33].

2.8.1 Camera

Cameras are the sensors in machine vision, and are used for producing images for further processing. Various factors in the application affects which camera is best suited for image capturing. These factors is among other things, light, surrounding medium, available space and weight. Regarding camera types, they vary in features as resolution, image sensor, frame rate, shutting technique and the ability to capture color.

Two-dimensional images provides the ability of calculating distance between pixels in the image. These distances are then relative to how much of the surface one pixel covers, and not in a measurable SI unit. By knowing the cameras Field Of View (FOV)-angle, and the distance between the camera lens and the image plane, it is possible to calculate the size of the surface the image covers [34]. From there it is possible to convert pixel distances to distances in a measurable unit. Figure 2.13 illustrates the FOV relative to the camera lens.

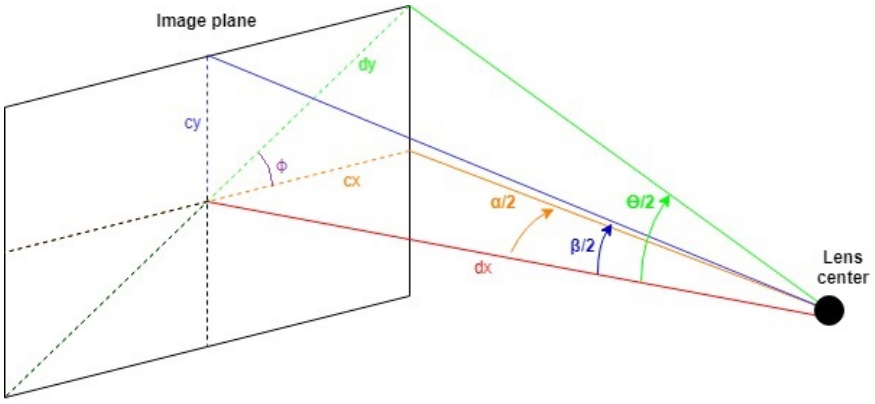


Figure 2.13: Camera field of view

Calculate size of image surface (Field Of View):

$$dy = dx \times \tan\frac{\theta}{2} \tag{2.6}$$

$$\phi = \arctan\Delta \tag{2.7}$$

$$cx = \cos\phi \times dy \tag{2.8}$$

$$cy = \sin\phi \times dy \tag{2.9}$$

$cy =$ Height of surface
$cx =$ Width of surface
$dy =$ Diagonal distance from center to edge
$dx =$ Distance from lens to surface
$\Delta =$ Ratio of pixels in camera (y/x)
$\phi =$ Diagonal field of view

2.8.2 Surroundings

When implementing machine vision it is important to ensure that the surroundings of the machine provides conditions, that does not negatively affect the quality of images being captured. Various conditions might impact the result from the machine vision, amongst these conditions we find background, illumination and the medium between camera and the object to be captured.

- Background of a image can be crucial when trying to locate something specific in it. The background should have a different color from what is being searched for, as it can be difficult to distinguish between foreground and background if they have the same color. On the other hand, a proper background might even provide the possibility of extracting more specs from a image, such as size relative to a mark in the background.
- Illumination can provide both advantages and trouble for the processing of a captured image. When tuned correctly one can easy the task of finding objects in a image. This can be done by using the correct combination of colors in the illumination, making the same colors in the image stand out as it is reflected. On the other hand, a poorly tuned light setting can remove the desired color from being detected or making it impossible to detect anything as the light is reflected as shimmer from a surface, such as water [35].
- The medium between camera lens and surface to capture can create trouble if not considered carefully as it concerns the visibility. Air conditions such as fog and smoke are specially hard to operate in, as they remove the visibility of what is in front of the camera [36].

All of the conditions above and more are crucial factors for a machine vision system. As a rule of thumb for most implementations, it is easier to provide a well suited environment, than it is to handle a poor environment in the processing of images [36].

2.8.3 Data extraction

Last step of machine vision is extracting data from the images captured. This process is called digital image processing, and can be read about in chapter 2.9.

2.9 Image processing

Image processing consists of processing and extraction information from images. The use of image processing can be separated into two application areas: human vision applications and computer vision. What these application areas have in common is that they are both used for image analysis (figure 2.14). Human vision applications have humans as the end user, thus limiting the amount of information that can be extracted from the image to what is possible to see with the eye. With computer vision applications, the end user is a computer. Large amount of information is possible to extract, as a computer is capable of revealing almost the entire electromagnetic spectrum, while the human eye is limited to detecting only visible wavelengths. Another advantage computer vision has over human vision is the ability to neglect the features that are irrelevant for the task at hand.

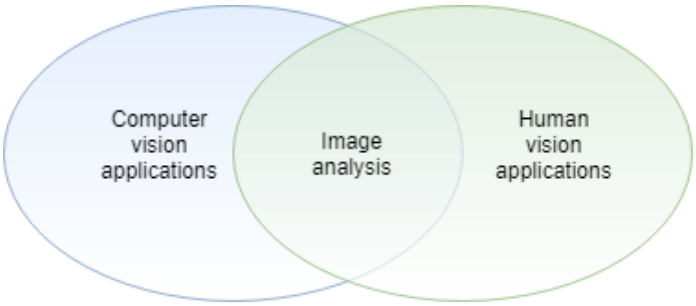


Figure 2.14: Image processing
[37]

Computer vision consists of a variety of methods and techniques used for manipulating and extracting information from digital images. Often this image processing is used for machine vision (chapter 2.8) where the result of the processor determines the next action of the machine. Processing techniques as *segmentation* and *morphology*, are used for finding features in images [37].

2.9.1 Segmentation

Segmentation is the techniques used for partitioning images into segments, finding objects or boundaries in images, the segments are then labeled making them easier to separate from the image.

Tresholding is the easiest way for segmenting or binarizing a image. It uses the gray-level or color-level for selecting what should be retained when making the image into binary values, 0 and 1. The gray-level used for deciding what to retain and what to discard is either set manually by the user or by adaptive methods using local or global features in the image. Manual thresholding is a point-operation, meaning that it checks the gray-level of each pixel. If the gray-level is above the threshold level the pixel is set to the binary value 1, otherwise it is set to the binary value 0. Adaptive thresholding techniques either calculates a thresholding level for the entire image, or for all local areas in the image. The thresholding value is calculated by a predefined feature, such as skew or variance.

2.9.2 Morphology

Morphological operations are used for changing pixel values in gray-scale and binary images relative to the value of the neighbour pixels. As the operations uses the neighbour pixels for alternating a pixel value, and not the pixel value itself, it is well suited for use on binary images. A structure element also called template is used for controlling which neighbour pixels that are being considered.

Erosion and dilation are two basic morphology techniques used for alternating the size of the blobs in binary images. The erosion operation has the effect of shrinking the size of binary objects, while dilation has the opposite effect and enlarges the size of binary objects. Figure 2.15 shows a small binary image before and after the morphological operations erosion and dilation has been used. The same structure element is used for both erosion and dilation.

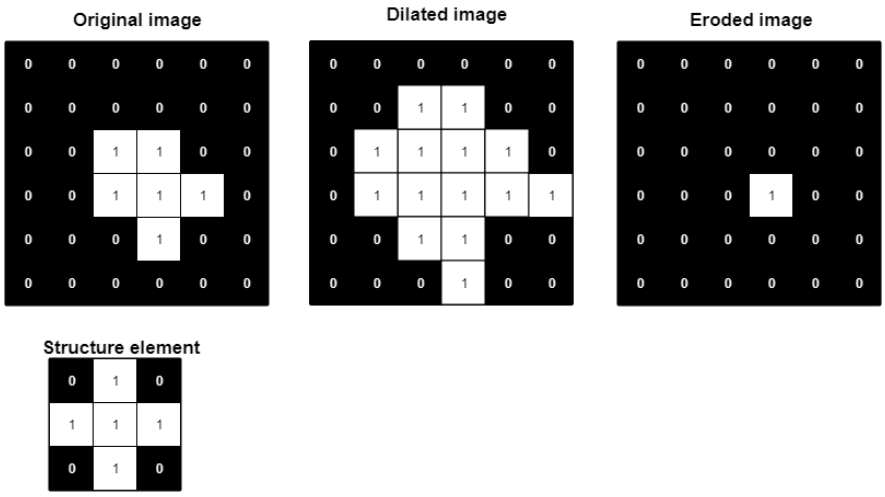


Figure 2.15: Morphology erosion and dilation example

2.10 Traveling salesman problem

A well know optimization problem is the traveling salesman problem (TSP). In short terms, the problem is to find the shortest or fastest rout from destination A to B while visiting a number of additional destinations. This route is referred to as the fittest tour. For solving the TSP, several different tour construction heuristics can be used. These heuristics have benefits and drawback considering time and accuracy. To measure the accuracy of a heuristic we need to know the optimum solution, yet, if we did know the optimum solution there is no need to try to find it. Therefore some methods can be used to estimate the time or distance the optimum solution will have. One of these methods is the Held-Karp lower bound, which is a mathematical approach giving a estimated optimum solution [38]. By implementing construction heuristics one will usually get within 10-25% of Held-Karp lower bound. If this do not satisfy the requirements, a improvement heuristic will have to be added [39]. Two construction heuristics are explained in chapter 2.10.1 and 2.10.2. An improvement heuristic is explained in chapter 2.10.3.

2.10.1 Random tour

This approach is a fast way of estimating the shortest tour. It starts by finding a start-destination, that either can be predefined or selected by random. It then generates N random tours holding all destinations in random order. By calculating total length of each tour, the shortest can be found. Finding the fittest tour by random works fine for applications where the number of destinations are small and N is high. Frequently this approach will return a solution far from optimum [40].

2.10.2 Nearest neighbor

Nearest neighbor might be the easiest implemented solution to TSP. The algorithm starts by finding a random destination to start at. From there it finds the nearest destination which has not been visited, and move there. This is repeated until all destinations are visited, it then returns to the start destination. This algorithm will give a solution within 25% of the Held-Karp lower bound [39].

2.10.3 Genetic algorithm

The two tour constructors mentioned in chapter 2.10.1 and 2.10.2 will give a solution far from optimum. For that reason, it might be a necessity to use a tour improvement algorithm to increase the fitness of the result. A Genetic Algorithm (GA) can be used to achieve tour improvement. GA is a class of stochastic search algorithms based on biological evolution. If a problem is clearly defined a GA can be used for finding a optimum solution by representing a set of different solutions as binary strings [41].

A GA starts by randomly generating a populations of chromosomes with fixed number of genes, where each chromosome can be a solution to the problem at hand. In TSP a chromosome represents a tour, and a gene represents a destination. A fitness function is defined to measure the fitness or performance of a chromosome. Fitness function for TSP is calculating the total travel distance. When a population is generated, a GA is used for training the population. First crossover with the possibility P_c will be conducted for all chromosomes. Crossover is done for finding a new and improved solution. Two parents are chosen for mating, the choosing can be done by different methods such as roulette wheel or tournament selection. Tournament selection creates a tournament with a given size. Then random chromosomes from the population are chosen to participate and the fittest will win. The two parents breeds two children by taking a random number of genes from parent one and add it to the child, then adding the missing genes from parent two. To make shore all solutions can be accomplished, mutation with possibility P_m is applied to all chromosomes. In TSP mutation is done by switching the place of two random genes in a chromosome. Training will continue until the termination criterion is satisfied. Termination criterion can be a given time expired, number of generations or until the result is as good as required [42]. A way to ensure improvement, is to add an elitism option. This option will take two good chromosomes and add them to the new population without changing them. Often these are the two fittest chromosomes from the previous population.

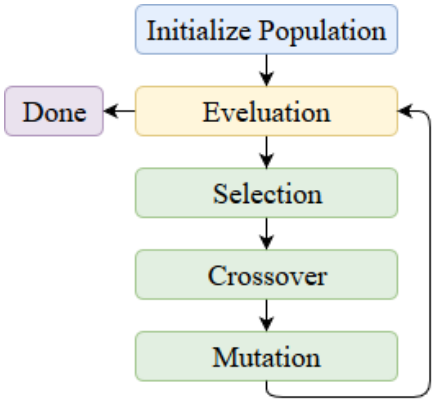


Figure 2.16: GA flow chart

2.11 Bresenhams algorithm

The Bresenhams algorithm is used for creating a digital straight line between two points. It is originally used for calculating which pixel to illuminate when drawing a straight line in a graphical representation, thus working with integers only [43].

Chapter 3

Approach

Chapter three is a subjective project approach. It gives an understanding of what has been important to the project regarding desired quality, requirements and specifications. This includes group organization, selection of robot, motion solution, roe removal technique and their respective testing methods.

3.1 Project approach

The group will have one project leader and one secretary. Once a week the project leader will have a comprehensive check of how far the group representatives have come in their individual tasks. The secretary will book meetings and write a summary of these. Even though the group has a leader, there will be practiced a flat organization where decisions and agreements shall be taken together. Each team member will be responsible for one or more main tasks and report to the leader. This will relieve the leader so it is easier to keep an overview.

Every other week the group will request a meeting with the supervisors. During these meetings, we will discuss the progress, solutions to possible challenges, and suggestions for possible improvements. In the early stages of the project, we will establish a plan based on the task given by NTNU. To ensure a good result and efficient approach a detailed project plan has been made. This plan contains detailed information about when subtasks shall start and end. A short version of our plan is listed below. The thesis will be written simultaneously as all these points.

- Make a detailed concept drawing of the total system. It contains kinematic, measurements and a basic design. This will give all the team members an overview of task needed to be done and a basic idea of what the result should be in the end. All concept drawings are going to be drawn by hand.
- Two group members will start researching and finding components while the two other starts designing the prototype. The robot will be designed using 3D CAD, which will provide us with the opportunity of simulating and verifying the design before building it. Hopefully, this leads to fewer flaws. All parts will be ordered early in the project and most of the designed parts will be completed. Consequently, we can create parts needed to be manufactured while waiting for the parts ordered.

- Programming will start when all parts are ordered. One will complete the rest of the design while the other three is set to start the programming. The first will start making the program structure. The second will start programming the communication part and the last one will start with the robot motion. Since a lot of the parts will be 3D printed which takes time, the production of these parts will be done along the programming.
- Assembling the robot will be started when the ordered parts are received and the manufactured parts are finished.
- Testing will be performed when the robot is assembled and the necessary programs are completed. All objectives from chapter 1.3 will be tested separately before tested combined.

The full version of the project plan can be found in appendix A.1.

3.2 Robot

To design and build a robot to this applications, several elements need to be taken into account. The most general specifications are mentioned beneath.

3.2.1 Robot selection basis

The robot needs to be designed in a way suited for the facilities found in a hatchery. As can be read in appendix Appendix B hatcheries can have a limited amount of space. This means that the robot needs to be as space efficient as possible. Also since the there are people working with other task in the same facilities, the robot needs to be designed with low risk of harming them.

For making a robot, one or more methods providing motion is needed. An industrial robot has to possess at least three degrees of freedom (chapter 2.2). Thus the robot needs to perform at least three motions. When choosing the method for creating motion there are two criteria that needs to be met. First criteria being accuracy for operation. Second criteria being that the environment cannot be exposed to contamination of oil and grease. For removing roe, the accuracy of the robot needs to be high enough to position itself at the center of a roe, with the size of roe ranging from 5.3 to 6.7 millimeters in diameter [6].

For generating motion, some kind of actuator is required. To find fitting actuators, it is important to calculate the torque needed for operating each motion. When an actuator has been selected, a fitting driver is necessary for operating it.

3.2.2 Roe treatment

The robots main task is to remove dead roe from trays in the hatchery. Because roe and fry are fragile, it is important to handle it in a gentle manner. Therefor the robot needs to be able to operate at low speeds while handling the trays where roe is stationed.

In order to remove the dead roe, the robot must be able to locate it, and for this machine vision can be used. The machine vision needs to be capable of finding roe with the challenges provided by the hatcheries environment, with respect to the dimmed illumination (chapter 2.1.2).

After the machine vision has found one or more dead roe, it needs to be able to remove it as smooth as possible for avoiding stressing the roe and fry (chapter 2.1.2). The most critical part of removal, is to avoid damaging the roe that is next to the one being removed. For that reason, the method used for removing roe has to be gentle.

Another way to avoid stressing the roe and fry is by limiting the time it is exposed to the robots operation. This can be done by optimizing the route that the robot travels when performing the removal.

3.3 Review

Throughout the implementation of the project we are going to perform reviews of both hardware, software and design. The reviews will be performed in order to make sure that the work performed and choices made are the best option for creating a machine that operates as desired.

- Software review will include checking that the code follows the rule of thumb with high cohesion and low coupling. Tasks of each controller and programs should be well defined.
- Hardware review is needed for controlling that the total system of hardware works well together with a functionality that is as fail safe as possible.
- Design review will be performed for assuring that components are well placed avoiding collision, and that the various solutions are as simple and effective as possible. A focus with the design is to make the prototype with as few parts as possible, leading to few sources of error and easy to service.

3.4 Testing

Several tests have to be performed during the project. Some test approaches are explained beneath.

3.4.1 Illumination test

With illumination being necessary for capturing images of the roe, we need to test what light yields the best result when sorting dead roe from living. Criteria for the illumination is that the brightness should be kept to a minimum as this creates stress with the roe and fry (chapter 2.1.2. Another criteria is avoiding glare from the light reflecting on the mediums surface (chapter 2.8.2).

3.4.2 Image processing testing

The image processing techniques developed, needs to be tested in order to verify that it can recognize the dead roe. By applying a set of images where the number of dead roe is known it can be

compared to the number of dead roe detected from the image processing and thereby calculate the accuracy of the techniques. The result of this test will reveal how good the image processing is at finding dead roe, but equally important is it that the number of living roe miss-classified as dead is kept to a minimum.

3.4.3 Communication testing

The communication used will at first be tested in small scale with only the controller nodes, to verify that the method works with the chosen controllers for the project. To ensure that the method used is robust and optimal for this application it needs to be tested in the environment that it will operate in during normal operation. This can be a harsh environment for communication as it includes power supplies and motor drivers that generates electromagnetic noise.

3.4.4 Suction testing

The suction method has to be tested to verify that it suits the implementation. A few criteria is considered important when testing and choosing the suction method and technique. The first criteria being that the suction provides enough vacuum to pick roe from the trays. The second criteria is how controllable and precise the suction is, this needs to be good enough to pick just one roe at a time. The last criteria is the noise level the suction provides, as this is critical for avoiding stressing the roe and fry (chapter 2.1.2).

3.4.5 Electrical cabinet testing

The electrical installation needs to be controlled before it is powered. This is necessary to avoid damaging components by applying wrong voltage levels or short circuits. Controlling the wiring can be done by using a multimeter for measuring the resistance from point to point in the circuits. Before applying power one should disconnect all sensitive electronics and thereafter measure that the voltage supplied is correct before reconnecting them.

3.4.6 Robot testing

For the robot to function as specified it has to perform a set of tasks correctly. The joints have to be tested independently before the complete robot can be tested and deemed operational.
These tasks are:

- Move to a specific position
- Stop if it moves outside of its boundaries
- Recognize trays
- Open and close trays
- Recognize and find dead roe
- Remove dead roe

Chapter 4

Materials

This chapter gives insight and explanation of choices that have been made. We are taking a stand in which type of robot to be built, and why. The components that were chosen will also be elaborated. Since the project is a proof of concept, all of the components are chosen based on their price without considering the environment where it is supposed to be implemented.

4.1 Data collection

One of the leading participants in Norwegian aquaculture, Marine Harvest at Ytre Standal, let us into one of their many facilities along the coastline. Besides letting us get explicit pictures, they shared experiences and necessary knowledge of the roe. For a more detailed description of our field trip, see Appendix B. The pictures are essential in developing the image processing, and it is important to have material that is current affairs. Further on we use statistics to study the death of roe in its entirety. To verify the information from Marine Harvest, we have cross-checked against national statistics retrieved from the roe-report from SalmoBreed and StofnFiskur from 2017.

4.2 Robot selection

By analyzing the environment where the final product will operate, we found a robot suited for this application. The selection of robot types was made up of the Cartesian robot, Scara robot, and the Delta robot. The Cartesian robot would make it possible to perform heavy operations in three DOF, as all joints are independent of each other. It is able to reach a large volume relative to the area possessed. The Scara robot would be a good option for a fast application, its downside is the available roof height needing to be twice the height of the highest point to reach. The last option, Delta robot would make for a fast and strong machine. Its greatest disadvantage is the space needed outside the reach of the robot for moving.

With the various criteria taken into consideration, we chose the Cartesian robot (chapter 2.2.1). This robot gives us an effective workspace compared to the usage of space, and makes it possible to handle trays without an external device. The Cartesian robot chosen, has two parallel prismatic joints moving vertically along the z-axis and two prismatic joint moving horizontal where one is for the x-axis and the other for the y-axis. This setup will provide three degrees of freedom in

a Cartesian coordinate system. It also provides the ability of defining the robot as two separate systems, being the horizontal- and vertical system. An illustration of the robots kinematics van be found in figure 4.1.

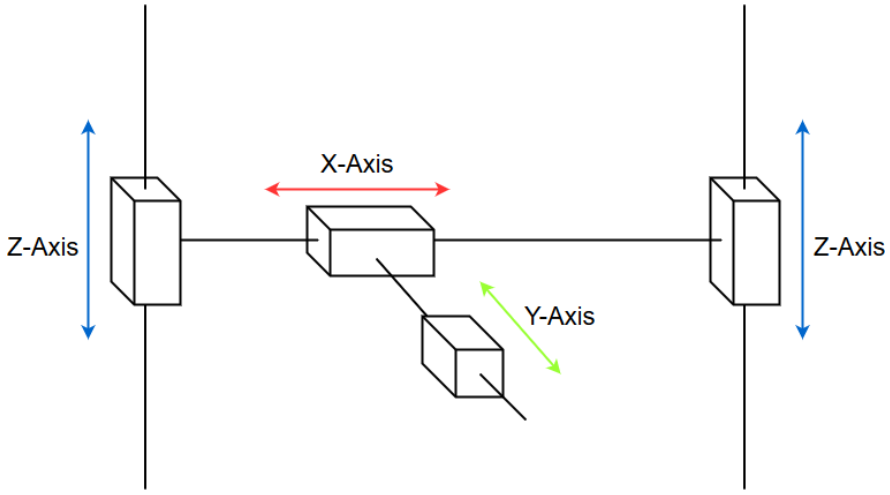
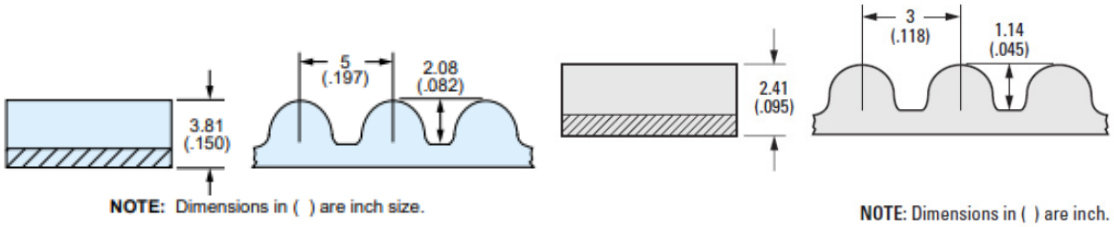


Figure 4.1: Cartesian robot joints

4.3 Motion method selection

For creating linear motion to the machine we considered using ball screw driven actuators, belt-driven actuators or rack and pinion driven actuators. By using ball screw we would able to move heavy components with high accuracy at low speeds. The downside by using ball screw actuators is their steep price, and the need for grease if the bearings needs lubrication. Belt drive is suited for moving light weight components over significant distances at various speeds. A drawback to belt drive is elongation of the belt making for a imprecise or faulty movement. On the other hand, belt drive is easy to implement and the price is low. The last option was rack and pinion. This motion method can carry heavy weight over long distances with high accuracy. The greatest drawback to rack and pinion systems is the need for lubrication on the entire rack for smooth operation. Another drawback is that the motors needs to be mounted on the components moving, making it a heavier system to move.

By combining the limitation set by the area of application being no expose of lubrication and our budget being low, our choice for creating linear motion fell on the belt driven actuator. For the vertical motion the ball screw actuator was extensively considered, as belt for this motion could be prone to elongation from the robots weight. As the robot is not intended for heavy lifting, we considered belt drive to be well suited due to the relatively low weight of the horizontal system. The belts chosen needs to be strong enough for operating its respective axis without elongation becoming a problem. For the vertical axis we chose a 9mm wide HTD5M timing belt. For simplicity we chose to use the same belt type on both horizontal axes. The choice for these axes was 6mm wide GT2 timing belts. Figure 4.2a and 4.2b illustrates the shape of the chosen timing belts.



(a) HTD5M timing belt [44] (b) GT2 timing belt [45]

Figure 4.2: Timing belts

4.4 Motor selection

As we chose belt drive as motion method, we were in need of actuators that could run the belts. Rotary motors were the obvious choice of actuator, as motion of the belts are generated by rotating the pulleys. While researching for the choice of rotary motors, two types of motors stood out. The first was servo motors. Servo motors are made for high torque, high accuracy and high speed, but does not work as good for running at low speeds. (chapter 2.5.2). The second option was the stepper motor. Stepper motors are more suited for running at low speeds, and is not bounded to run a closed loop with encoders. A drawback considering stepper motors is the speed and torque characteristics. When the speed goes up, the torque is reduced drastically (chapter 2.5.1). On the other hand, if the stepper motors are working with low payload, it is able to maintain high precision.

With the above taken into consideration, our choice fell on the stepper motor as actuator for our prototype. Stepper motors will provide high accuracy at normal operation, and even higher accuracy if combined with microstepping.

4.4.1 Motors for horizontal motion

For the horizontal movement we decided to use NEMA 14 sized stepper motors which can be seen in figure 4.4. This was because it complied the two important criteria for our project. The first criteria was the holding torque. It is important that the motors are able to move the load without difficulty. The second criteria was the motor size, as a result of the limited space. For simplicity the motor choice for x- and y-directional movement were the same. For this reason the calculations below are based on the load for the x-directional motion being the motor pulling the most weight.

By using Newton's first law and knowing the sum of all weights for all components moving in x-direction, the radius of the pulley and using a standard ball bearing friction coefficient $\mu = 0.0015$ [46], we can estimate the torque needed. In figure 4.3 one of the axis in the horizontal system is illustrated. The friction provided by the pulley (R_p) will not be taken in to account as it is almost equal to zero.

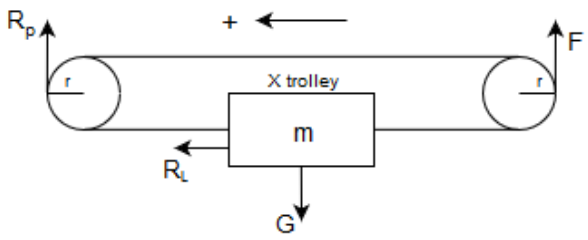


Figure 4.3: Horizontal system

Friction of the linear trolley

$$R_L = \mu \times m \times g = 0.015 \times 1.08 \times 9.8 = 0.16N$$

Torque needs to exceed

$$\tau = R \times r = 0.16 \times 12.2 = 1.92mNm$$

$g = 9.8N$ Gravity
$r = 12.2mm$ Radius of pulley used on motor.
$m = 1.08Kg$ Total weight moving in x direction
$\mu = 0.015$ Friction coefficient
$R(N)$ Friction
$\tau(Nm)$ torque

From this deriving we found that the motors holding torque needs to be greater than 1.92 mNm. Explained in section 2.5.1, a stepper motor decreases its torque by increasing the speed, thus it is recommended to choose a motor with higher holding torque than the absolute necessary to be able to move faster. The motor we have chosen is rated with holding torque 150 mNm, which will be more than enough. Data sheet for the motor can be found in the bibliography under section [47].

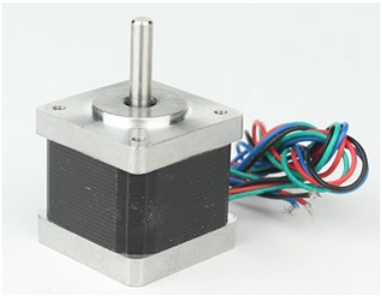


Figure 4.4: Nema 14 stepper motor

4.4.2 Motors for vertical motion

When we chose the motors for vertical motion, the most important criteria was to be able to hold the payload of the horizontal system in a still position while operating on a tray. We decided to use a Nema 23 sized motor because this was the most cost efficient stepper motor with electrical brakes built in. The chosen motor can be seen in figure 4.6. The brakes makes it possible for the motor to hold its position while the vertical system is at rest. By not powering the motors phases, we avoid overheating and unnecessary power consumption. To achieve stability for our robot we decided to use two motors for handling z directional movement. The payload is distributed equally on both motors.

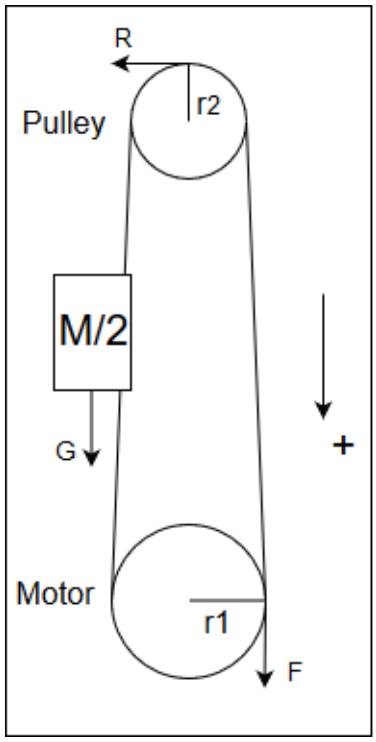


Figure 4.5: Vertical system

To find the torque needed for operating the vertical motion we used Newtons first law. A illustration of the vertical system can be seen in figure 4.5. We assumed that the weight of the horizontal system would be 6 kg which leads to 3kg payload for each motor. We use a standard ball bearing friction coefficient $\mu = 0.0015$ [46]. Because of the bearings low price we were uncertain of their quality, hence we decided to used $\mu = 0.01$ to be on the safe side.

Calculating the total force needed to keep the system at rest.

$$F = G + R = mg \times m\mu = 3 \times 9.8 + 3 \times 0.01 = 29.43N$$

Finding the total torque the motor needs to generate for moving the system without slipping.

$$\tau = F \times r = 29.43 \times 12.16 = 357.9mNm$$

$g = 9.8N$ Gravity
$r = 12.16mm$ Radius of pulley used on motor.
$m = 3Kg$ Weight moving in z direction on each side
$\mu = 0.01$ Friction coefficient
$R(N)$ Friction
$\tau(Nm)$ torque

To be able to apply motion to the vertical system, the motor needs to generate more than 357.9 mNm. The motor we have chosen has the rated holding torque of 1.3 Nm, and the built in brake has the rated holding torque of 0.6 Nm [49]. More details about the motor can be found in the data sheet [50].

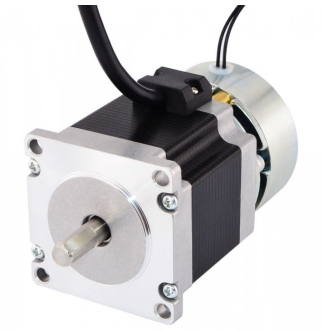


Figure 4.6: Nema 23 stepper motor with brake
[48]

4.5 Motor driver selection

4.5.1 DRV8825

The DRV8825 motor driver was chosen for controlling the NEMA 14 stepper motors. These drivers can supply the power needed and has the highest microstepping resolution in this size. As the accuracy needed unknown at this time, and the high-resolution microstepping this driver provides give us the best resolution within the price range. The driver is possible to stack in driver shields.

4.5.2 DM556

DM556 stepper motor drivers were used for controlling the NEMA 23 stepper motors operating the vertical system. The drivers are 2-phase digital stepper motor drivers built in protection for over-voltage and over-current. These drivers are controlled by digital signals from a controller, setting the direction and speed of revolution. In addition, the drivers need an enable-signal to power the motors, this feature let us mechanically stop the motors by operating an emergency stop switch breaking the enable-signal.

4.5.3 KeyeStudio driver shield

The driver shield was chosen because it is able to stack the DRV8825 motor drives and connect them directly to an controller. This shield is made to fit three drivers and gives access to digital and analog pins plus some extra power pins. It has 12V input power which supplies the drivers and the controller. Information about this shields I/O can be found in appendix G.

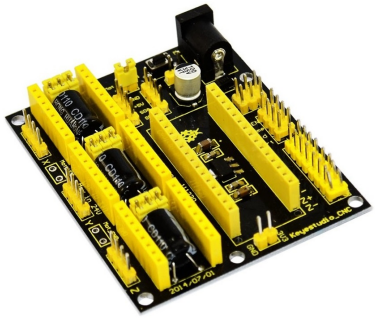


Figure 4.7: Driver shield [51]

4.6 Vacuum source selection

To remove the roe, it was chosen to use vacuum. To create vacuum a vacuum pump can be used, or a vacuum generator. The roe needs to be sucked through the hose and into a bucket. Since we could not find any vacuum pumps that could carry the roe, it was decided to create a venturi generator for generating vacuum, see figure 4.8. Consequently it could be designed so the roe can pass through it and work with both water and air as motive fluid. Which motive fluid is best suited needs to be tested.

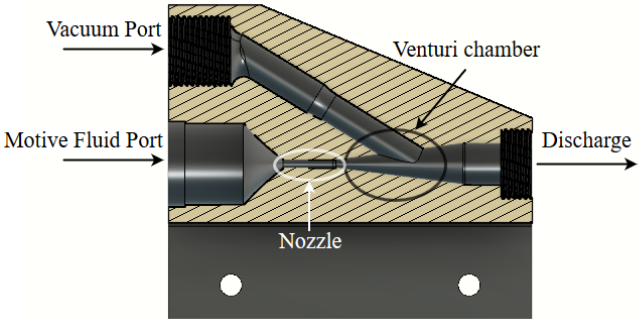


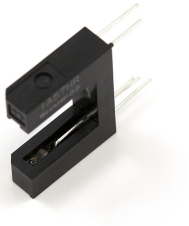
Figure 4.8: Venturi chamber

4.7 Choice of sensors

For controlling the robot safely within the operating ranges, sensors were needed. It was desirable to use optical sensors since they do not wear off by being in contact with what is being measured. On the other hand their need for extra wiring made them less attractive for use in confined spaces.

(a) Mechanical endstop [52]

(b) IR sensor [53]



(c) Optical endstop [54]

Figure 4.9: Chosen sensors

4.7.1 Mechanical endstop

Micro switches (figure 4.9a) were chosen for detecting the outermost points the robot can reach in x- and y-direction. Easy to install, minimum amount of wires and space made these the choice for sensors for the horizontal system.

4.7.2 Line detectors

The optical sensors seen in figure 4.9b, has a 5 meter detection range and was chosen for safeguarding the robot from crashing when moving up and down. These sensors are made of two parts that is mounted separately, where one is the IR emitting diode and the other is the receiver.

4.7.3 Optical sensors

Optical line detectors, seen in figure 4.9c has a 1 cm detection range and is used for detecting when the top and bottom of the vertical system is reached. The sensors are compact with the diode and the receiver mounted in the same component.

4.7.4 Incremental rotary encoders

Photoelectric incremental rotary encoders were chosen for safely controlling the motion of the vertical system. These encoders have a resolution of 600 pulses per revolution.

4.8 Choice of controllers

The programmable controllers are a vital part of the robot movement. We needed different controllers to control the operation of each axis. Multiple controllers made it possible to separate our system, like mentioned in section 4.2.

4.8.1 Odroid-XU4

The Odroid will operate as our main controller. It runs on Linux and provide good support of different coding languages and compatibility for Computer Vision, which is needed for the image processing. It also holds integrated options for USB, ethernet and GPIO pins for connecting hardware.

4.8.2 Supplementary controllers for I/O

Even though the Odroid is a good choice for us, it lacks inputs and outputs (I/O). Therefore it was necessary to run microcontrollers alongside for handling input and output interaction. This interaction was needed to control both systems of the robot. All group members were familiar with the Arduino series of microcontrollers, hence these were the obvious choice. The Arduinos must hold the required I/O for controlling their system.

Model	**Digital I/O**	**Analog I/O**	**Processor**
Arduino Mega	DI/O 54	AI 16 / AO 15	ATmega2560
Arduino Uno	DI/O 14	AI 6 / AO 6	ATmega328P
Arduino Nano	DI/O 14	AI 8 / AO 4	ATmega328V
Arduino Micro	DI/O 20	AO7 / AI14	ATmega32U4

Table 4.1: Arduino table
[55] [56] [57] [58]

4.8.3 Horizontal system controller

The Arduino Nano is small and therefore easy to install in tight spaces. It holds the required amount of I/O needed for controlling the horizontal system. Compatible shields that fits our stepper motor drivers are pre-built and mountable on the Nano. Therefore, the choice for the horizontal system controller landed on an Arduino Nano.

4.8.4 Vertical system controller

The Arduino Mega was chosen to control the vertical system of the robot. It was chosen because of its high number of I/O connections that were needed to control the vertical system safely.

4.9 Communication protocol

Serial using TTL was the chosen protocol for the communication between the controllers. This was chosen because its supported by all our controllers. With serial there is no need for a bus, and a point to point connection is easy to maintain and handle. By using USB connection from the Odroid to the other controllers the cables are by standard, shielded from start to end. This eliminates some noise which was wanted because the Serial with TTL is not very noise resistant, and the stepper drivers emits a small amount of electromagnetic noise to the system. As the Odroid communicates with each device on separate lines, it also eliminates collision and cross talk between the devices.

4.9.1 Connecting the controllers via USB

For connecting Serial via USB-port to the I/O pins on any controller, the USB interface has to be converted to Transistor-Transistor-Logic(TTL). For connecting the Arduino Nano to the Odroid we decided to use a shielded cable connected to the *RX/TX* pins on the Nano. Therefore a five volts based FTDI cable that converts the USB to TTL was used [59]. The other Arduino Mega was connected directly from its USB connector to the Odroid USB.

4.10 Machine vision

A crucial factor to the prototypes success, was the ability of locating dead roe. For this machine vision was chosen as detection method.

4.10.1 Camera selection

The camera chosen to use for the image capturing, is a Logitech C920 HD PRO webcamera (figure 4.10). Logitech C920 is a FULL-HD webcamera with 1920 x 1080 pixels and 78°field of view [60]. The cameras high resolution and field of view makes it very convenient for capturing detailed pictures.

Figure 4.10: Logitech C920 HD PRO
[60]

4.10.2 Illumination

An external light source had to be implemented for illuminating. This is because the robot will work in a dark environment where the camera cant see. For illumination, addressable RGB led strips were chosen. Each led on the strips can be controlled separately and has stepless regulation of red, green and blue color. No specific quality was required, therefore the WS2812B led strips available at the lab were used.

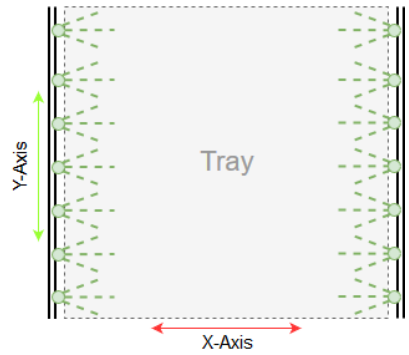


Figure 4.11: Illumination

4.11 Construction

When choosing material for the main construction there were some options, the selection consisted of stainless steel and aluminum. One essential criteria needed to be satisfied, we needed to contemplate the materials handle-ability. Stainless steel would need a lot of rework from our side, the pieces would have to be measured, cut and welded. The pieces would have to be measured, cut, and then welded together. This, in comparison to aluminum, which could be bought as profiles that already were cut and extruded to our measurement.

By considering these points, aluminum profiles became the apparent choice for the main construction. All parts used for assembling the frame, that is not self produced was made of aluminum. Self produced parts was 3D printed in plastic or cut in plexiglas.

4.12 Electrical cabinet and components

The materials used in the electrical cabinet were mainly composed of what could be gathered from the lab or bought online. How the components where connected and how they where used can be seen in Appendix E.

4.12.1 Power supplies and ampere calculation

When designing an electrical system, the supplies has to be chosen from the required voltages and power consumption of the electrical components in the system. In this system there is a range of different voltages. Because of this, multiple power supplies with different voltages are needed or voltage converters. Voltage converters can be single converter components or integrated circuits. To choose supplies we had to look at the consumption of the different voltage levels. The following currents in table 4.2 were calculated from RMS and is a theoretical maximum estimate of the power consumption of all the components combined. The idle or regular consumption will be lower than this estimate.

Component	Voltage	Ampere
Nema 23 motors - 2.3A each	24V	4.6A
Nema 23 Break - 0.16 each	24V	0.32A(measured)
Nema 14 motors - 0.8 each	12V	1.6A
Arduino Mega with I/O	12V	2.16A
Arduino Nano with I/O	12V	0.89A
IR-Sensors	12V	0.6A
Odroid XU4	5V	4A

Table 4.2: Theoretical current calculation
[48] [61] [55] [62] [63] [64]

The chosen supplies are the ones which were available to us from the lab inventory and delivered enough power. The supplies rating and their respective voltage rails can be seen in table 4.3.

Voltage rail	Ampere	Supply rating
24V	4.6A	10A
12V	5.24A	3.33A
5V	4A	2.4A

Table 4.3: Total current calculation

4.12.2 Fuses

Fuses were selected based on their breaking power. We chose one fuse for each power supply, this is to secure the supplies from over-current and short circuits. All fuses has type C characteristics, and rated under the maximum current consumption of the supply it is protecting. The rating of the fuses can be seen in table 4.4.

Voltage	Breaking ampere
24V	6A
12V	3A
5V	3A

Table 4.4: Fuses current and voltage rating

4.13 Hatchery imitation

For being able to test our robot, we chose to create a simple imitation of a rack system used in hatcheries.

4.13.1 Test medium

As a replication of the living roe, we selected soft orange water ball.White plastic air soft bullets were the most convenient choice for simulating the dead roe. The orange balls vary from 5 to 9 millimeter in diameter and the white bullets is 5 millimeter in diameter. The white plastic bullets and orange water balls makes for a good imitation of the roe at breeding facilities as can be seen in figure 4.12 and 4.13.

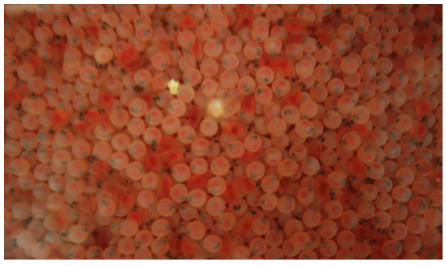


Figure 4.12: Roe

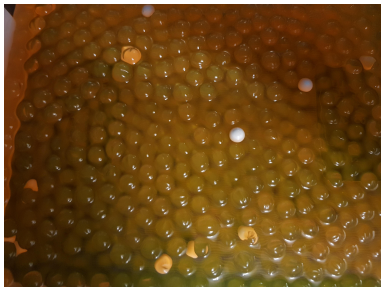


Figure 4.13: Roe imitation

4.13.2 Rack system

We decided to use a chest of drawers as a rack to test our application. Its total size is 110cm wide, 570mm deep and 970mm high. The chest holds three drawers. Each drawer has an area of 900mm x 470mm, and a full extension of 325mm. For being able to use this chest as our rack system it needed some modification. Images of the altered chest can be seen in figure 4.14 and 4.15. The following adjustments were done for making the chest suitable as a hatching system:

- The three trays were rebuilt by trimming the top of each drawer and raising the bottom. This created a more realistic work area, in comparison to the real trays filled with roe.
- The bottom of the tray was made able to contain water by adding $390 \times 790\ mm$ large plastic trays. This is necessary for holding the roe substitute and water. The shape of the plastic trays makes the roe stay inside a $650 \times 200\ mm$ large area (figure 4.15).
- Metal plates on the front of each tray have been added, which will be used for handling the trays with magnets.

Figure 4.14: Rack imitation

Figure 4.15: Open drawer with roe

4.14 Software and libraries

The following software and libraries has been used throughout this project.

4.14.1 Software

- **NetBeans 8.2** - is a free software development tool. It was used for developing Java software, which is its main supported language.
- **ShareLaTeX** - a free web-based tool for writing in LaTeX. Some of its features are spell checking and cloud-based storing. The cloud-based storage makes real-time editing possible and thereof its possible to cooperate in the same files.
- **Arduino IDE** - free environment for programming and interacting with the arduino micro controllers. Released and made by the arduino corporation. All arduinos has been programmed using this IDE.
- **PCSchematic Automation School v19** - Electrical CAD software. Has been used to draw all electrical schematics and our electrical layout diagrams. The school version does not contain the same limits as the free version.
- **Fusion 360** - 3D designing software.
- **Cura** - Slice STL files and generates G-code for 3D printing.
- **Rhinoceros 5** - 3D modeling program.
- **Draw.io** - For making simple diagrams and illustrations, web based.
- **MATLAB** - Programming platform

4.14.2 Libraries

The following libraries has been used in combination with NetBeans, MATLAB and Arduino IDE.

- **Open CV 3.4** - Open Source Vision Library is a free library used for computer vision and machine learning. The library has C, C++, Python and Java interfaces [65].
- **Pi4J** is a library providing access the full I/O capabilities on a Odroid or Raspberry Pi platform [66].
- **DipImage** is an add on toolbox for MATLAB used for image analysis and processing [67].
- **jSerialComm** A platform independent java library for using serial ports [68].
- **Stepper** to control unipolar and bipolar stepper motors [69].
- **FastLED** for controlling LED chipsets [70].

Chapter 5

Design

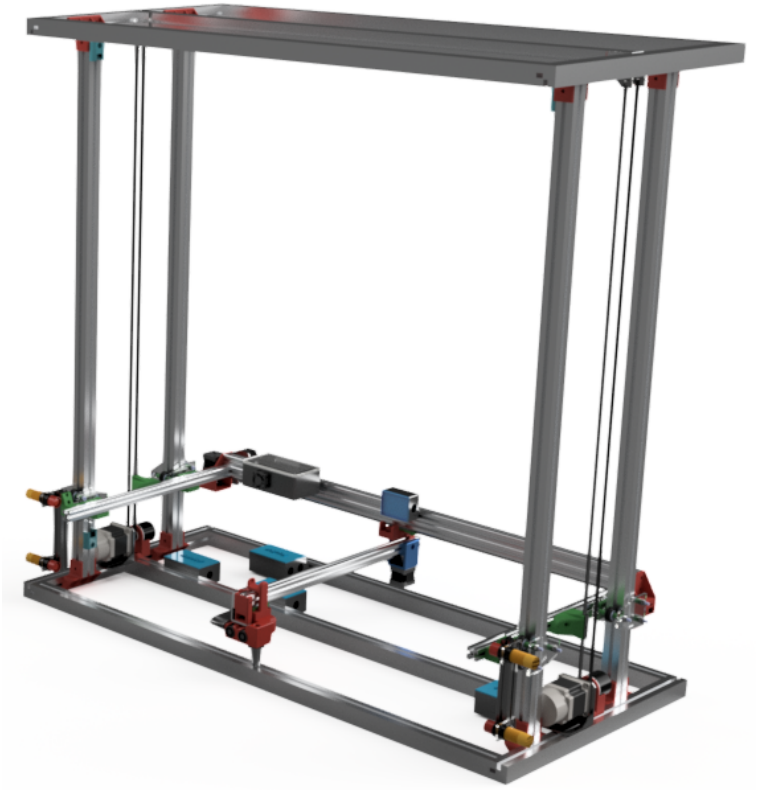


Figure 5.1: The RoeBot

This chapter explains how the most essential parts of the robot is designed and manufactured. All constructed parts are designed using Autodesk Fusion360 or Rhinoceros. Custom parts were manufactured in the FabLab at NTNU campus Aalesund. The FabLab has 3D-printers and a CNC laser cutter. For that reason we needed to design the components in a way suited for manufacturing manufactured in these machines.

5.1 Main frame

Figure 5.2: Main frame

The main frame is the robots bearing construction and the shell constraining the robots area of movement (figure 5.2). The material chosen for the main frame is aluminum profiles, as decided in chapter 4.11. The frame is constructed by 40x40 millimeter profiles, aluminum L-connectors and 3D-printed T-connectors made for the purpose of connecting the profiles at 90 degree angles. When designing the main frame, the rack systems size needed to be taken into account. The drawers are 900mm wide and can be extended 325mm from the rack (chapter 4.13.2). To be able to open the tray, the main frame needs to be wider than 900mm. Also it is desirable that all moving parts are kept within the frames circumference. As a result of these conditions the robots main frame is 1000mm wide and 500mm deep. For being able to work in the upper tray the robot needs to be taller than the rack system. Consequently the height is 1080mm. As a result of all moving parts being within the frames circumference, an encapsulation can be added for safety. This encapsulation can also omit unwanted light sources.

5.1.1 Designed parts

To avoid the frame from wobbling during operation, T-connectors were made (figure 5.3). It was discovered during assembling that the L-connectors used for joining the vertical and horizontal aluminum profiles, did not provide enough stability to the frame. The combination of the L- and T-connectors made the construction more stable, and made it possible to perform a precise operation of the robot in all directions.

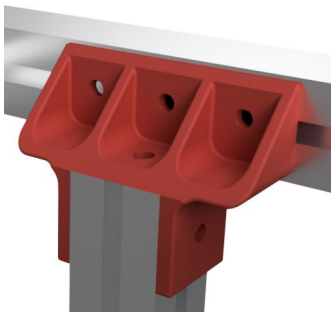


Figure 5.3: Frame T-connector

5.2 Horizontal system

Figure 5.4: Horizontal system

The horizontal system is designed to perform the x- and y-directional motion. Equipment used to provide linear motion, is two Nema 14 motors and two linear rails with sliding trolleys generating two prismatic joints. Sevens sensors are mounted on the system, where four works as end stops, two are safety beams, and the last is for detecting when the end-effector hits a tray. A custom end-effector is mounted at the end of the y-axis joint. The frame is assembled using 3D printed parts and aluminum profiles with a size of 20x20mm and 20x40mm. All the profiles are assembled using 5mm screws and t-nuts, the motors and both linear trolleys are assembled using 3mm Allan screws. For being able to mount the horizontal system inside the main frame, the aluminum profile at the back is cut to 900mm. The two profiles on the sides are 430mm and the profile moving along the x-axis is 470mm. Hence 30mm of space will be left to add a drag chain at the back for cables coming from electrical cabinet.

5.2.1 Designed parts

Motor bracket for x-axis motor is designed for a Nema14 sized motor. It is made to be assembled with three screws on to the horizontal systems frame. In addition to fastening the motor, holes are made for fastening the cable chains entering from the main frame and the cable chain passing to the y-axis trolley. In figure 5.5 the motor holder are shown in red. The two gray parts are the starts of the two cable chains.

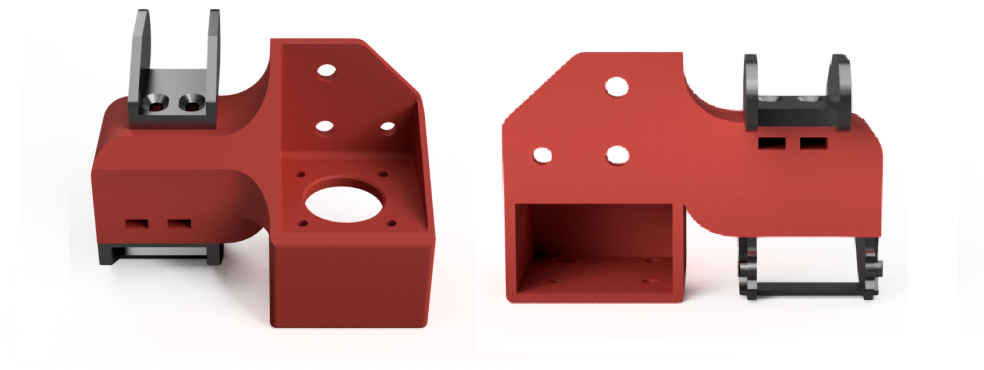


Figure 5.5: X motor holder

Motor bracket for y-axis motor is designed for a Nema14 sized motor. The screws for mounting the motor needs to be mounted inside the bracket and therefore it is hard to tighten them. This has been solved by adding holes large enough to fit an Allan key. The gap in the top of the bracket is made to fit the aluminum profile pointing in y direction. To provide the end-effector with maximum work area, the limit-switch is mounted on this bracket.

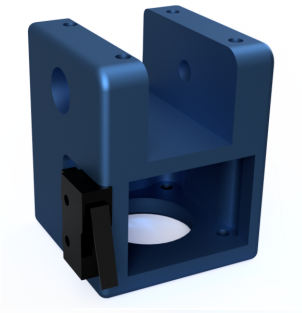


Figure 5.6: Y-axis motor bracket

Trolley bracket x-direction (figure 5.8) will be the link between the x-directional linear slider and the y-directional aluminum profile. A hole is made through the bracket making it possible to use and aluminum profile as bracing. To move the trolley in x-direction, a 6mm GT2 timing belt is used. For that reason we have made a pathway for the belt so it can move freely back and forth through the bracket. A trace is made with teeth so the belt can be assembled on to the bracket. To be sure that the belt is properly mounted and do not slip, the screws for assembling the x-directional linear trolley will expand the walls around and work as a clamp for the timing belt. As can be seen in figure 5.7, the holes has a slope making the expansion possible. The back and front side of the bracket is made for connecting the x- and y-directional cable chains.

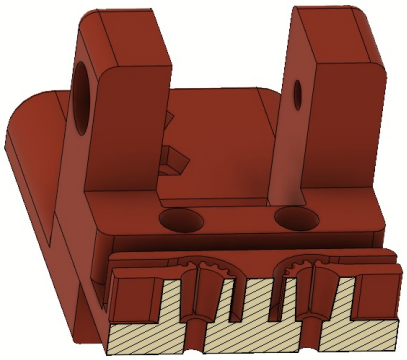


Figure 5.7: Timing belt lock

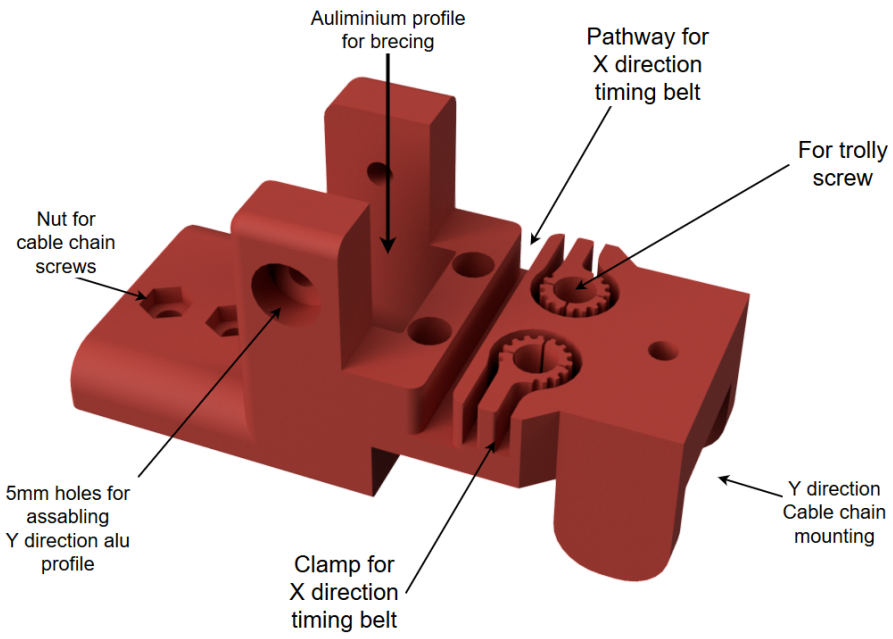


Figure 5.8: X direction trolley

End-effector is the part of the robot where all the roe handling equipment is mounted. It is developed to be assembled on the y-directional trolley so it can move along the y-axis. The tool bracket is represented in red in figure 5.9, which consists of two parts. The upper part (light red) combines the y trolley and the end tool with four M3 screws. A pathway and a locking mechanism for the timing belt is made in the same way as for the x-direction timing belt shown in figure 5.9. On the top of the part there is made a connection for the y directional drag chain. The lower part (dark red) is designed to hold the camera, an air hose for vacuum and a set of magnets for opening the tray. There is made a guiding for the hose which is easy to change so the length of the hose can be adapted. This makes it easy to regulate the depth of the TCP. Assembling the magnets is done with two 3x25mm screws coming in from the rear side of the bracket. In between the magnets and the bracket there are inserted springs. This gives the magnets more freedom to align itself when hitting the tray. In between of the magnet there id made a pathway for the cables going to the magnet end-stop.

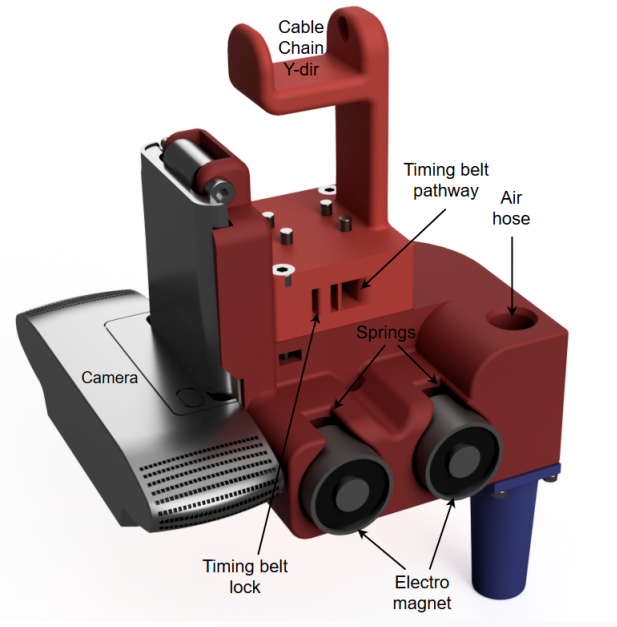


Figure 5.9: End-effector

5.2.2 Horizontal system electrical layout

electrical layout of the horizontal system is shown in figure 5.10. The component tags used in the figure is the same as can be found in the electrical schematics in appendix E.

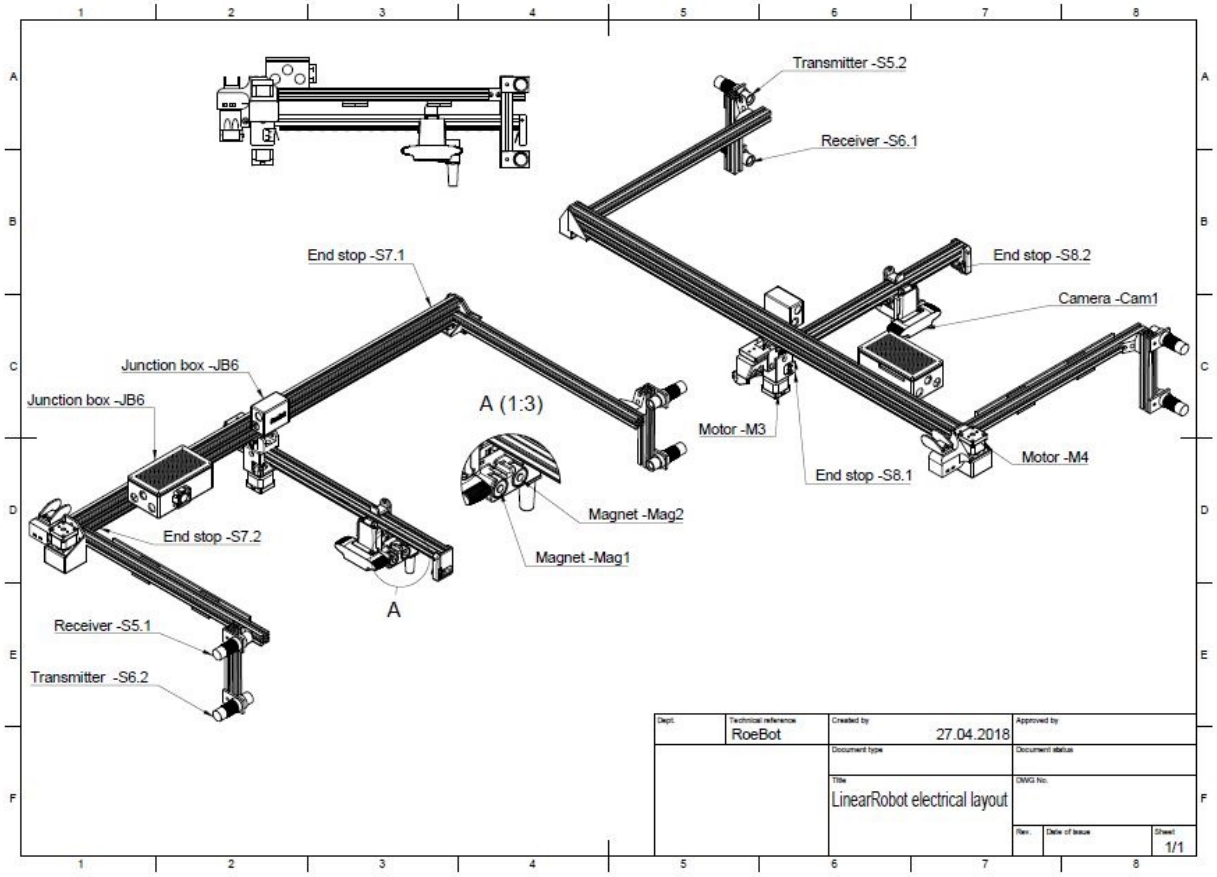


Figure 5.10: Horizontal system electrical layout

5.3 Vertical system

The vertical system provides movement along the z-axis and connects the main frame with the horizontal system. Two Nema 23 stepper motors are mounted on each side of the main frame and used to drive timing belts. The timing belts are connected to the horizontal system that is moving up and down inside the main frame. The timing belt connection with the horizontal system is built to provide movement in the z-axis and therefore it only has a stabilizing effect in the roll direction. Figure 5.11 displays the designed vertical system without the horizontal system mounted inside.

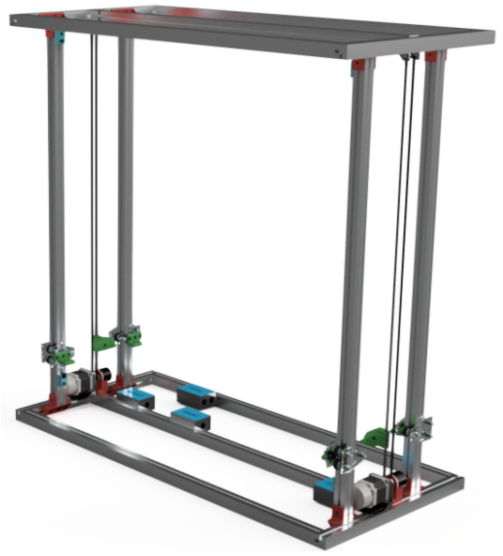


Figure 5.11: Vertical system design

5.3.1 Designed parts

Trolleys seen in figure 5.12, were created in order to make the horizontal system stable in pitch and yaw direction. Another feature of the trolleys are the flags on the inside, these are used for trigging the upper and lower optical sensors of the vertical system. The trolleys are built with two plexiglas, six 3D-printed spacers, three POM-wheels with bearings, three M5 bolts and nuts. The POM-wheels fit in the tracks of the vertically mounted aluminum profiles of the main frame, keeping the construction stable while operating and at rest. The trolleys are designed with a slot that makes it possible to tighten the wheels against the aluminum profile it operates on.

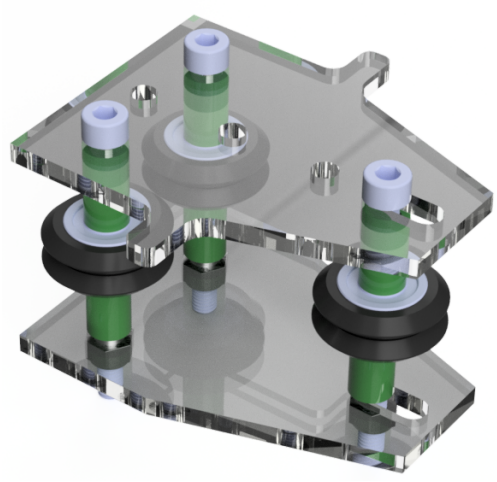


Figure 5.12: Trolley z-direction

Pulley holders are used for holding the upper pulleys that the timing belts run through. The pulley holders are adjustable making it possible to tighten the timing belts. M3 bolts are used with M3 nuts for adjusting the position of the piece sliding in the holders slot. Each side of the holders is composed of three layers of plexiglas. The first and inner layer has a pathway so the pulley shaft can be regulated up and down. Layer two is a guidance where a cubic shaft mounting can move up and down. Layer number three makes sure the shaft mounting do not fall out. The holders are shown in figure 5.13.

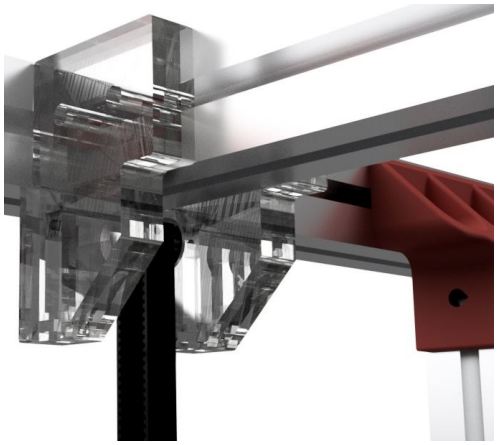


Figure 5.13: Vertical system pulley holder

Timing belt pulley with a connection in both ends, is created by 3D-printing. Because an HTD5M timing belt was decided to use in chapter 4.3, this pulley is designed by these measurements. The sole purpose of designing and 3D-printing these pulleys is the need for creating a connection between the motors and the encoders. M3 nuts and set screws are used for connecting the axles of the motors and encoders to the pulley. Figure 5.14 illustrates a pulley with timing belt mounted on the shafts of a motor and an encoder. This pulley has a tooth radius of 12.16mm. When using the HTD5M timing belt, the pitch radius is 12.74mm as illustrated in figure 5.15.

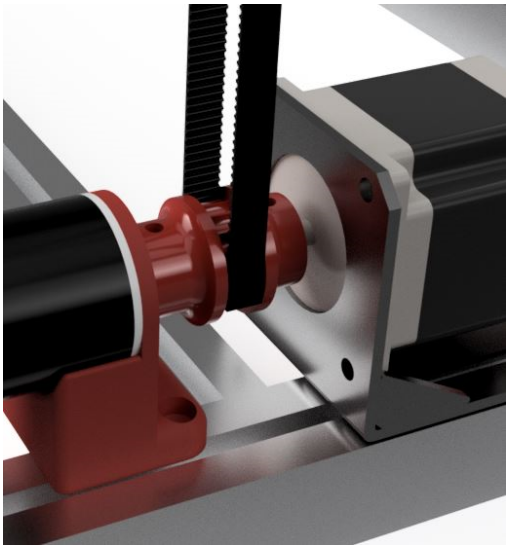


Figure 5.14: Vertical system timing pulley

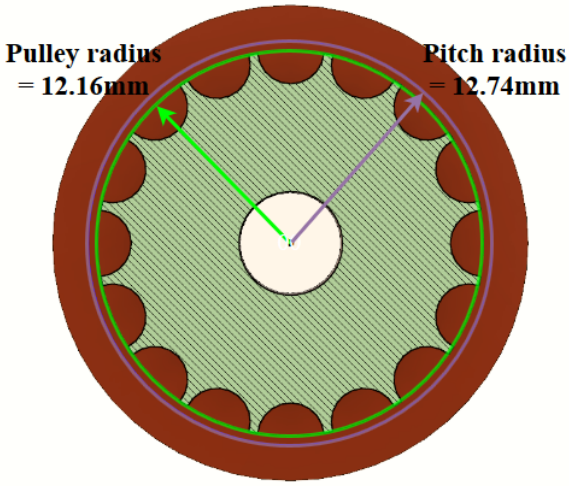


Figure 5.15: Vertical system pulley radius

Timing belt carriage, shown in figure 5.16 and 5.17, are designed and created by 3D-printing. These are the components connecting the horizontal system to the vertical systems timing belts, providing linear motion in the vertical axis. The timing belt is connected to the carriage by a clamp with the same pattern as the belt, M5 bolts and nuts are used for mounting the clamp and the horizontal system to the carriage.

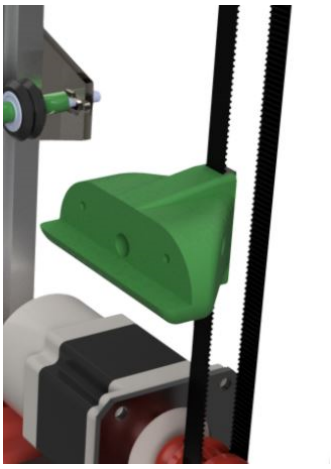


Figure 5.16: Timing belt carriage in place

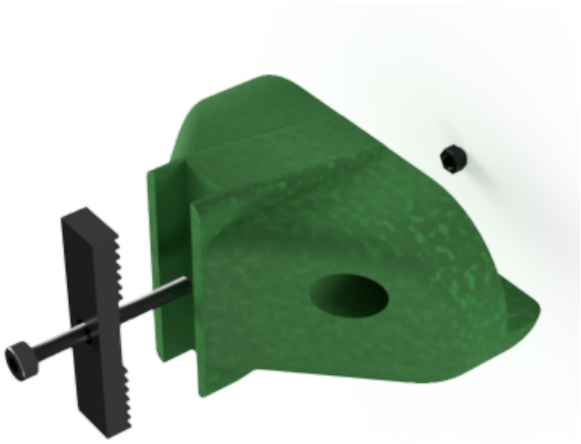


Figure 5.17: Timing belt carriage

Brackets for mounting optical sensors are created by 3D-printing. Optical sensors are mounted inside the brackets in a way that in case of a collision with a trolley, the sensors are not prone to wear. The brackets are mounted on the vertical aluminum profiles and are adjustable in the z-direction. In figure 5.18 a bracket with a sensor is displayed. Where it is mounted is shown in figure 5.19.

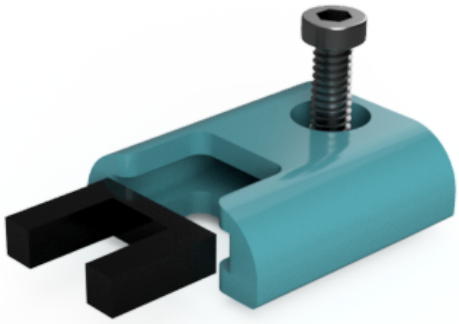


Figure 5.18: Bracket for optical sensor

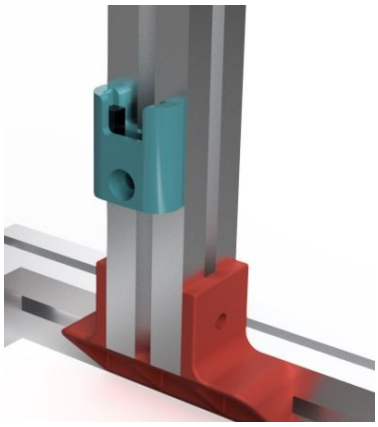


Figure 5.19: Bracket for optical sensor in position

5.3.2 Vertical system electrical layout

The electrical components associated with the vertical system is shown in figure 5.20. The component tags used in the figure is the same as can be found in the electrical schematics in appendix E.

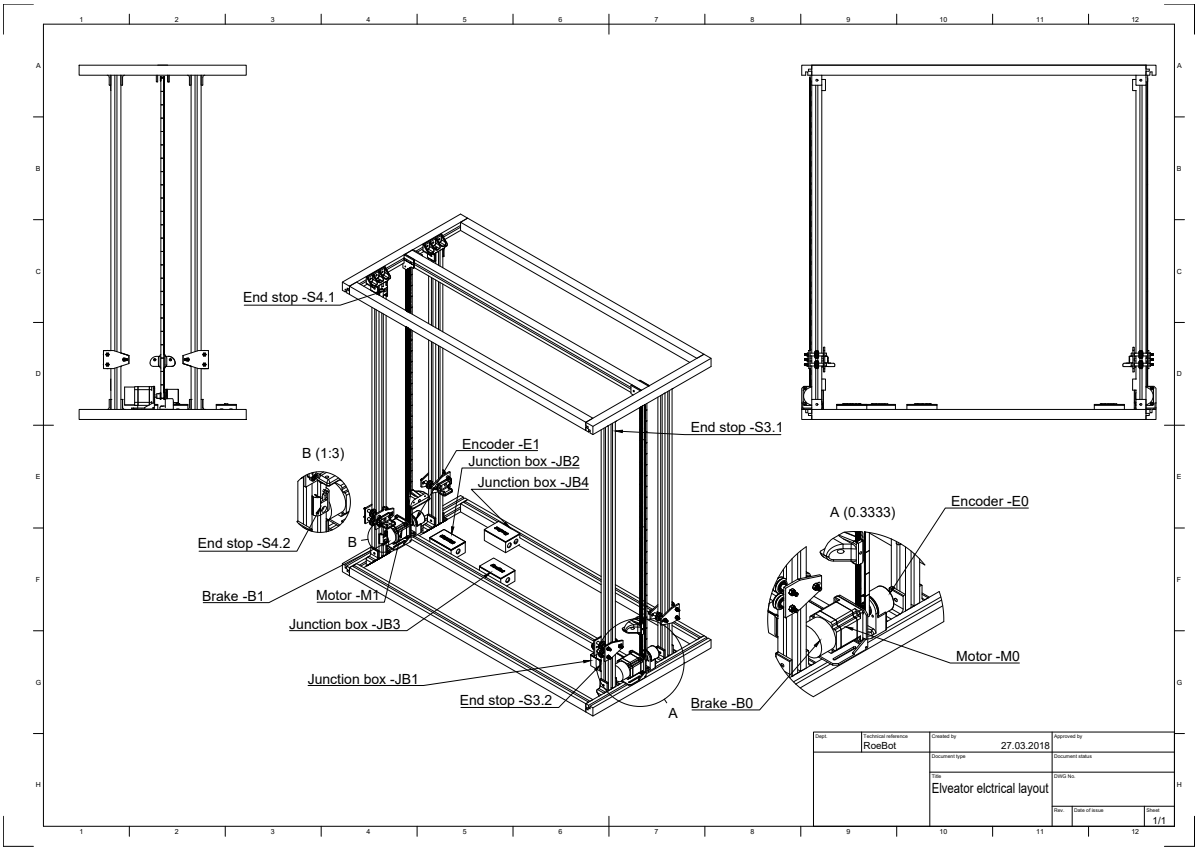


Figure 5.20: Vertical system electrical layout

5.4 Venturi vacuum generator

When designing the venturi chamber, the specifications mentioned in chapter 4.6 had to be considered. One of them being that the chamber needs to be big enough to carry the roe. The venturi vacuum generator is designed so it can process both air and water as motive fluid with minor adjustments. If air is used, the motive input port is designed with a threaded hole for a 3/8 fitting. In contrast, on the water version this hole is adjusted to 10.5mm without threads. Both the discharge and the vacuum port are made for 3/8 fitting, and therefore a variety of tubing sizes can be used. In between the motive fluid port and the venturi chamber, a nozzle with diameter of 2mm is implemented. From the nozzle and to the discharge, the path diameter increases with a five-degree slope. The test medium varies in size from 5-9mm (see chapter 4.13). Thus, the pathway from the vacuum port to the discharge is 9mm in diameter. To 3D-print this part without support, all slopes are less than 45 degrees when the vacuum and motive fluid port is faced down. The design of the vacuum generator is shown in figure 5.21.

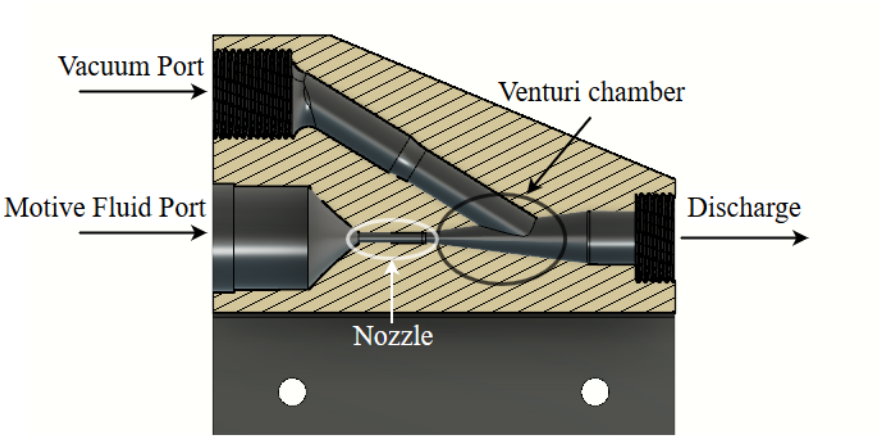


Figure 5.21: Venturi chamber design

5.5 Electrical cabinet

Figure 5.22 shows the inside of the electrical cabinet.

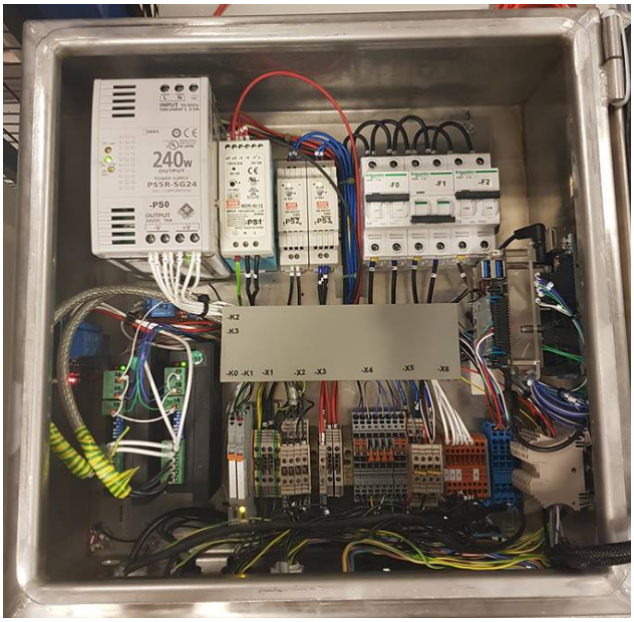


Figure 5.22: Electrical cabinet

5.5.1 Placement of controllers

Placing of controllers is critical as it defines the number of cables and wires needed to be assembled on the robot itself. For the first design, it was planned to only run power and communication wires from the main cabinet to the other parts of the construction. This means the controllers would be placed at the same location as the equipment it is supposed to control. This would make for fewer cables in the cabinet. The counter arguments for this was the extra space required for the boxes to

contain the controllers in, and having sensitive equipment close to working and operating hardware.

Regarding the Arduino Nano controller on the horizontal system, the latter is the biggest argument. As rattling and hazards might occur from being mounted on a moving part and close to other moving parts. It will also be closer to the trays and thereof susceptible to water. Because the Arduino Nano is mounted on top of the driver shield, all I/O is also connected to this shield. Everything needed from the cabinet is power and communication cables. The Arduino Nano was thereof chosen to be mounted on the horizontal system and put inside a box which was 3D-printed as seen in figure 5.23. The drivers can generate a lot of heat, therefore a fan is mounted on the box.

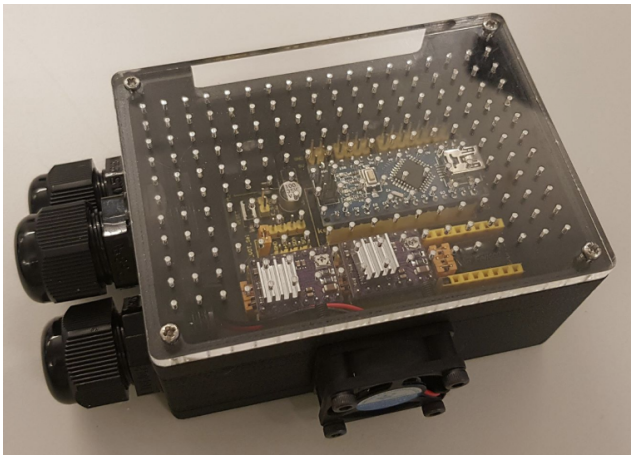


Figure 5.23: Arduino Nano with driver shield in ventilated box

The Arduino Mega(vertical system controller) however needs bigger drivers which are situated inside the cabinet and the I/O which is connected to this controller is placed all over the robot, and not situated in just one place. Thereof it is chosen to be mounted inside the electronic cabinet. However, placement inside the cabinet should be done while taking noise sensitivity into account. This means placing it away from noise radiating equipment such as power supplies and motor drivers to minimize the risk of any unwanted noise occurring. Precautions as grounding and placement is taken into account where the controller is mounted in the cabinet. The placement of the Arduino Mega can be seen in the right side wall of the electrical cabinet in figure 5.22.

Chapter 6

Implementation

This chapter explains the how the various systems of the robot works and how these are implemented in the finished system being the RoeBot.

6.1 System overview

Figure 6.1 introduces the system in its entirety. It consists of three main systems: HMI, Computer/processing and Robot.

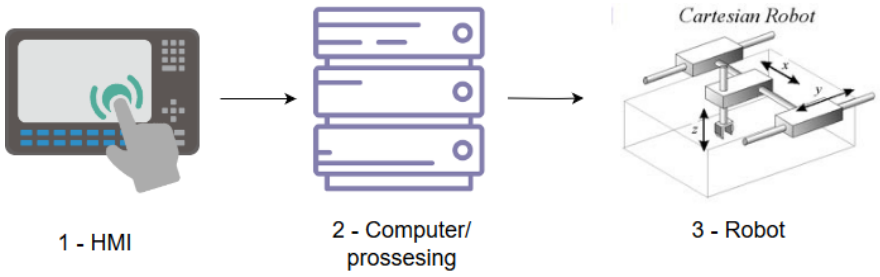


Figure 6.1: System overview
[71] [72] [13]

6.1.1 1 - Human machine interface (HMI)

Human machine interface (figure 6.2) is a platform that interacts with the process operator, also called the user. Consists of two sections, where the first section is the graphical user interface. It provides the user with information from the state of the machine, and in addition the opportunity to calibrate, start and stop the machine. The second section, which is buttons and lamps, are physical components. These can be used to control the machine. Three buttons are connected to the machine: start, stop and emergency stop. There are also two lamps, one for indicating system running, and one for indicating error.

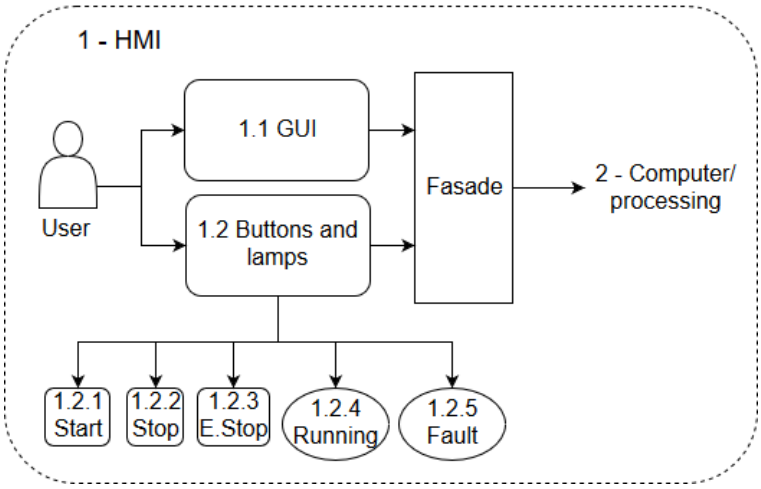


Figure 6.2: 1 - Human machine interface

6.1.2 2 - Computer / processing

The processing comprise the robots control logic. It includes a robot task manager keeping track of all available tasks and when to perform them. Connected to the task manager there are four modules. The first module is image processing, which analyses pictures and finds the positions of dead roe. Second module optimization. It finds the fastest pattern for removing the dead roe. Third module is statistics that hold information about the operation, such as the number of removed dead roe. The last module is communication. This handles all interactions with the robot, holding all available commands, and structure necessary for using a communication protocol. The computer and processing component also controls the camera taking pictures of the trays. The interactions are illustrated in figure 6.3.

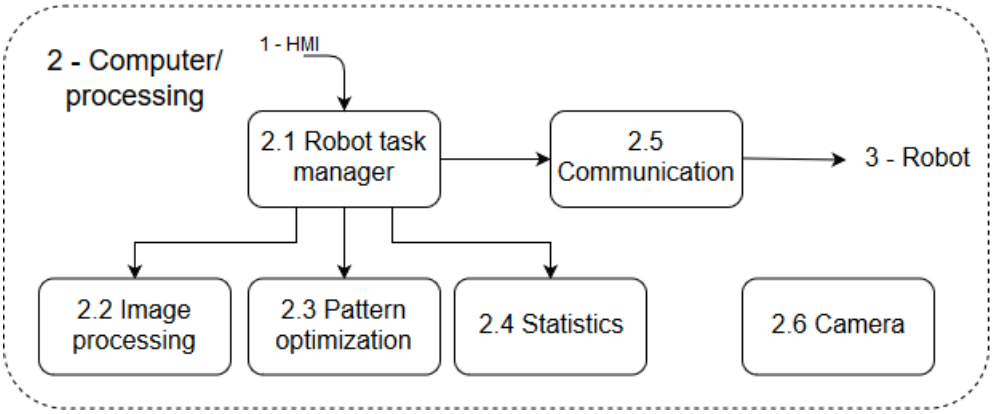


Figure 6.3: 2 - Computer / processing

6.1.3 3 - Robot

The last main component of the system represents the physical robot. It is divided into two modules, where each illustrates one system of the robot. Both contain a microcontroller which has a number of inputs and outputs connected to motor drivers and sensors. The controllers are programmed to transmit and receive commands and statuses. Figure 6.4 below, illustrates the two systems. The first module is the vertical system, and the second module is the horizontal system. An overview of the equipment connected to the controllers can be seen in figure. 6.5. By separating the Cartesian robot into two systems with specific tasks and responsibilities, the program and handling of tasks can be done individually on each controller.

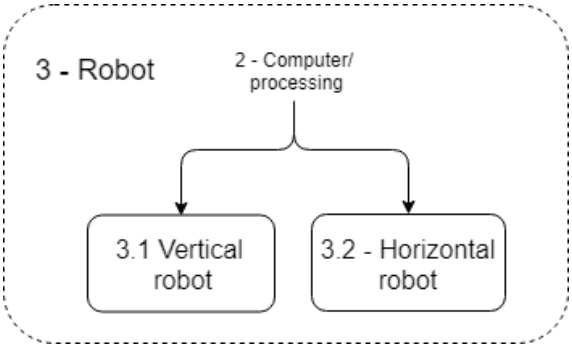


Figure 6.4: 3 - Robot

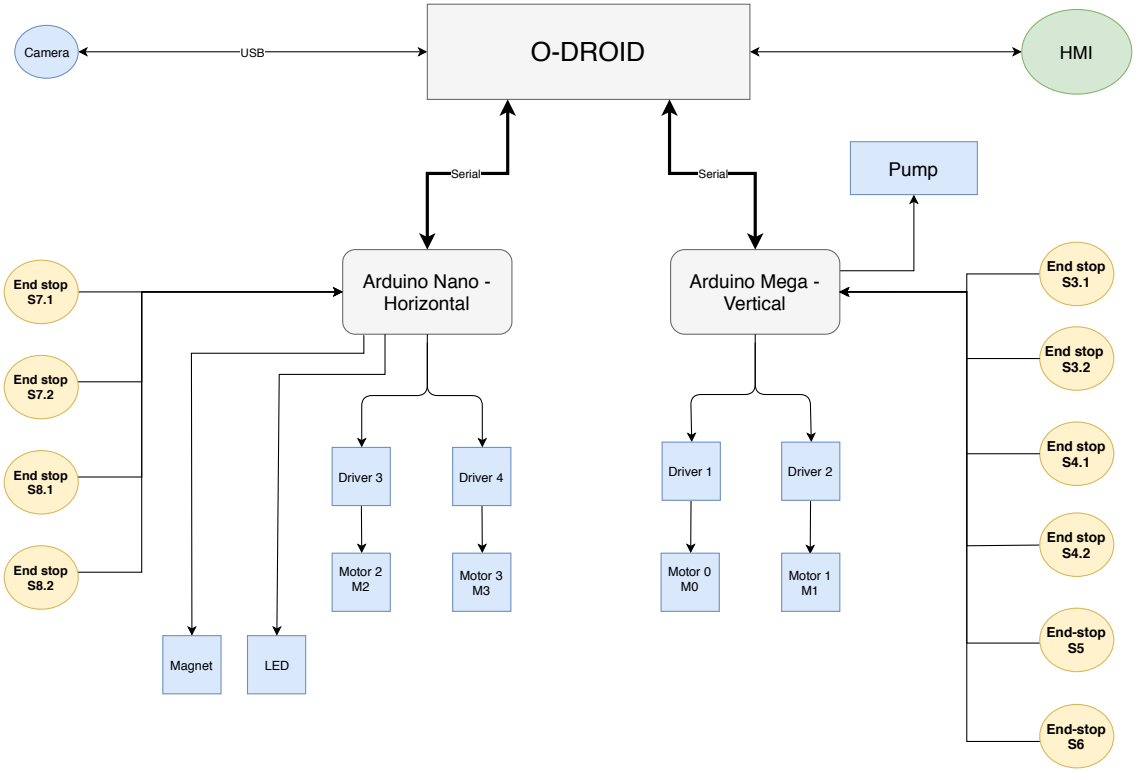


Figure 6.5: Overview of physical connections

6.1.4 3.1 - Vertical system

The vertical robot which controls motion along the z-axis, utilizes an Arduino Mega microcontroller (chapter 4.8.2). It controls two motor drivers and appurtenant motor brakes. Six optical sensors are used, where two of them are used as safety barriers and the other four as end stop sensors.

6.1.5 3.2 - Horizontal system

The horizontal system controls the motion along the x- and y-axis, and uses an Arduino Nano as controller (chapter 4.8.2). For a safe operation, mechanical switches are used as limit switches for each axis. It also controls the LEDs used for illumination.

6.2 Robot Program

This section elaborates how vital parts of the program and tasks were solved.

6.2.1 Program Structure

The java program that runs on the Odroid is developed with a high abstraction level. This means it will handle HMI interactions, image processing and among other tasks, calculate the routes of where to move. All low-level logic, such as checking sensors and driving motors, will be handled by the Arduinos. Interactions between the Odroid and the Arduinos are based on commands. The Arduinos are responsible for carrying out the commands, and answer in form of statuses to the Odroid. These statuses can relay information if that is requested. Statuses can be *ready*, *busy*, *faulty*, and more. The commands include tasks like *move*, *light regulation*, *suction*, and so on. Each Arduino has various components connected to it, and therefore each controller has a unique program. The way they handle the interactions with the Odroid is the same for both controllers. The abstraction level between Odroid and Arduinos results in fewer and only important messages being relayed by the communication, and the components are controlled internally by the controller they are connected to. Hence faster control of motors, lights and pump, and real-time update of sensors and other critical information is achieved.

6.2.2 Operation flow

The finished prototype operates in four states, see figure 6.6. First state is *Idle*, this is the state the system is in before it is started, and after it is stopped. Second state is *Calibrate*, the system is in this state when the robot is started for the first time by the user. In this state, the robot levels itself, finds its extremities and the height of the racks drawers. The third state is *Operate*, this is the state the robot is in when it is opening trays and removing the dead roe, a elaboration of this state can be read in chapter 6.2.3. The last state is *Error*, the robot is set into the *Error* state if a error occurs that needs the user to take action in order to continue operating.

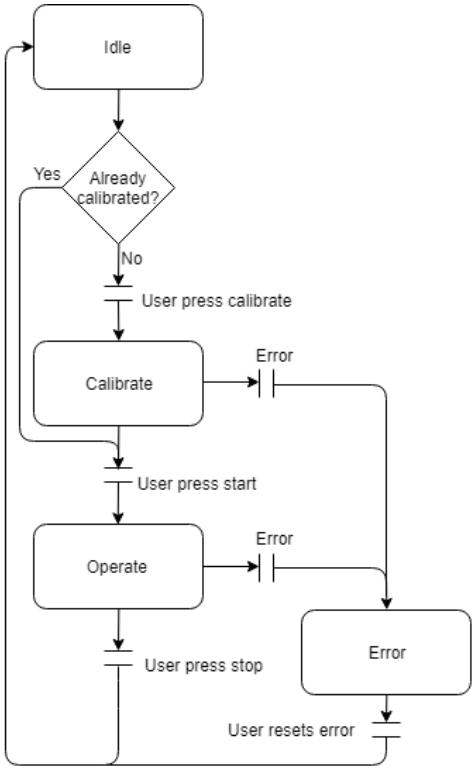


Figure 6.6: Main state flowchart

6.2.3 Operation state flow

Figure 6.7 shows what happens when the calibration is done and the user has pressed start and the main program enters the *Operate* state. The operate state comprise four internal states that control the tray handling, searching and removal of roe. It starts in the *Open tray* state where it opens a tray in the rack, the top tray is always the first to be opened. When a tray has been opened the operation state changes to *Searching*. Here the system captures images of the entire tray, then processes the images for locating the dead roe, and finally it finds the optimal route for removing the dead roe. When the searching is complete and dead roe was found, the state changes to *Removal*. In the *Removal* state, the system controls the removal of the dead roe in the order provided from the searching. The last state is *Close tray* where the open tray is being closed. This state is entered either when all dead roe is removed or if the searching finds none dead roe. When a sequence of the states has been performed for the first tray, the system repeats the sequence for the next tray. These steps are repeated for all trays until all have been checked. The system will then wait for a given time before it starts a new sequence, with the first tray and so on. If at any time an error occurs, the system is set to the *Error* state seen in figure 6.6.

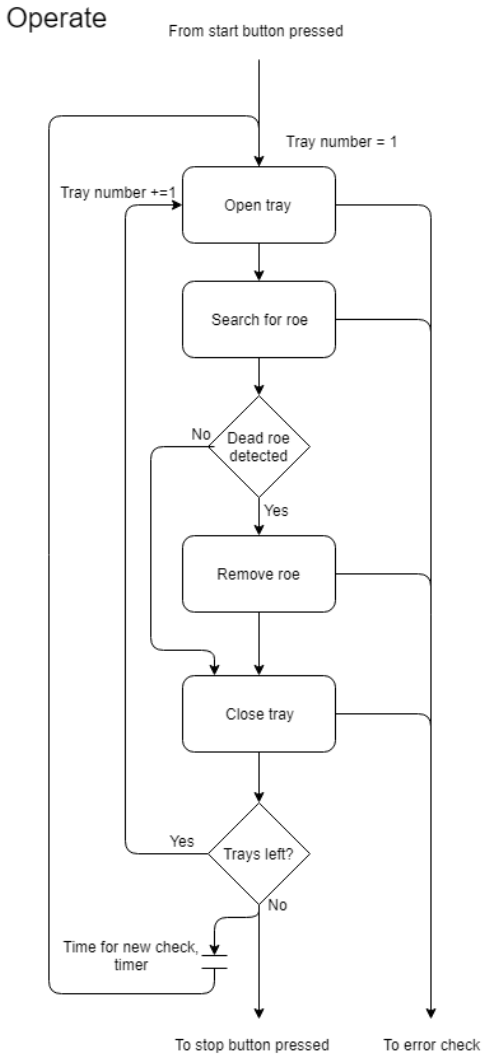


Figure 6.7: Operation state flowchart

6.2.4 Java classes

Figure 6.8 and 6.9 shows the interaction between the HMI and the *RoeRobotDevice* class, and from *RoeRobotDevice* to the Arduino.
The most important java classes which is developed for this program to function is listed below with an explanation of their main tasks.

- *GUI* gives a graphical interface where the user can control the system. From here the user can calibrate, start, stop and regulate the color of the light. It is also possible to view the state the robot is in.

- *GPIOHMI* represents physical buttons and lamps mounted on the cabinet. The buttons are connected to events, meaning the changes will be detected at any time.
- *RoeAnalyserFacade* is the link between the HMI and the *RoeAnalyser*. It contains the same methods as the *RoeAnalyser*. By using such facade a new HMI can be implemented without knowing how the rest of the system works.
- *RoeAnalyser* functions as the organizing class. This class holds the main loop where the robot can calibrate, run or stop. Methods in this class calls the methods in the *RoeAnalyserDevice* class. For instance, method for running will call on methods for opening tray, search for dead roe and remove dead roe one tray at the time.
- *RoeAnalyserDevice* operates as the robot when seen from other classes. It executes tasks called from the *RoeAnalyser* and contains functions as calibration, pick roe and capture an image.
- *RoeImage* is a class used for holding the images captured of the trays. It addition to the image, it holds the position of the roe located by the *ImageProcessing*.
- *ImageProcessing* receives a *RoeImage*, performs the processing of image for locating dead roe. The position of the dead roe located is added to the corresponding list in the *RoeImage* object. When the processing of a image is complete, the listeners are notified.
- *PatternOptimalization* optimizes a list of coordinates via a nearest neighbor algorithm or a genetic algorithm. Then it returns a new optimized list with the shortest found path for visiting all roe.
- *SerialCommunication* is the communication link for the java program. Via this class, the program can interact with controllers operating the robot. Other classes do not know about the controllers which this class interacts with. It handles the commandos it is given, and sends them to the appropriate controllers with correct payload. Incoming messages are parsed, and the listeners for the messages are notified.
- *Commando* is used as a superclass for all valid commands in the program. The under-classes of *Commando* tells which kind of commando they are. All commands holds their respective register address and payload. Table 6.2 lists the commands.
- *Status* class represents the superclass for answers from the microcontrollers. *SerialCommunication* class creates the status object with the corresponding incoming status address and attached payload. The status subclass tells what kind of status it is. Table 6.1 lists all the different statuses.
- *TrayRegister* representing a rack of trays.
- *Tray* serve as a tray in a rack. It holds the tray number, the amount of dead roe, and offsets in millimeter from the flag to other parts of the tray.

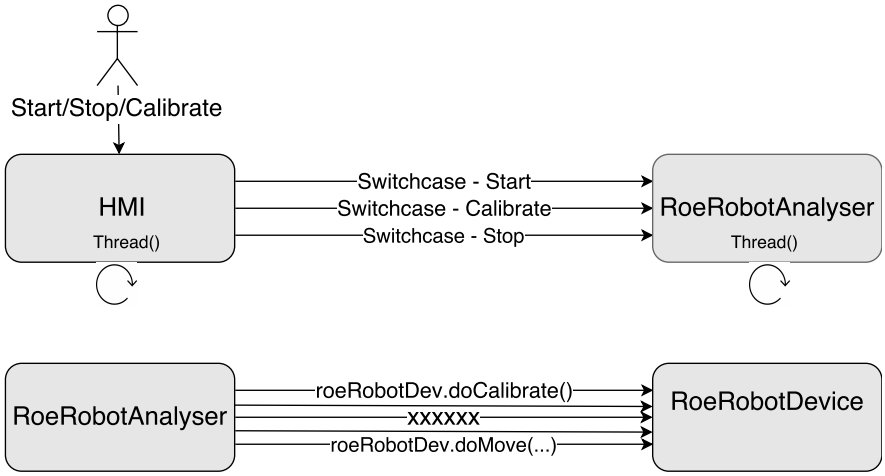


Figure 6.8: HMI class and main robot class interaction

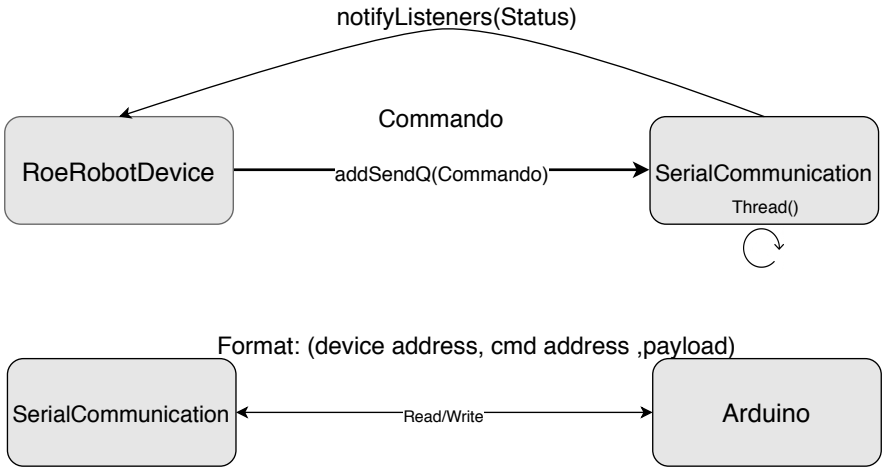


Figure 6.9: Main robot classes and Arduino controllers interaction

6.2.5 Arduino

The Arduinos controls all the low-level logic of the robot. This means some of its tasks is operating the stepper motors, constantly check end sensors and perform the commands received. They operate in switch case, which decides the state of the Arduinos and what tasks they shall perform. The state of the switch case is set according to the commands received from the Odroid. Calibrating, moving and fault are some of the cases it can reside in. The Arduinos main purpose is moving the stepper motors safely within the workspace as well as controlling the vacuum and the lights.

- *Safety barrier and end stops.* The Arduino Mega nor the Nano carries enough interrupt pins to connect all end stops and safety beams. It was thereof decided to check the state of all the sensors during each step of the stepper motors. Checking all sensors during each iteration requires more processing power, but it will not affect the processing time significantly. This

is also the only possible solution to secure a safe operation without using interrupt pins. The safety barrier sensors are mounted on the side-arms of the horizontal system in a position that makes them detect objects before the robot hits them. The top sensor triggers when there is an obstacle above the robot, and the bottom sensor is mounted beneath the end-effector, triggering if anything is beneath the robot.

- *Tray detection.* In the vertical axis the trays are detected by trigging the safety IR beams with flags mounted on the trays (chapter 4.7.2). These flags are mounted in the same position as the steel brackets on the tray, so the position of the steel brackets can be known. Steps for a flag is known by measuring the length of the flag in vertical direction, then calculating the number of steps the length correlates to. The steps moved in vertical direction are counted while the beam is broken. If the steps counted exceeds the calculated flag size plus a safety offset, the object blocking the beam is recognized as something else than a flag. The robot is then stopped and put in a faulty state to avoid it from crashing. When the robot moves in the opposite direction of the triggered sensor, it is no longer considered as an obstacle.

- *Find tray.* To find the trays, the Arduino Nano will move the end-effector in y-direction one step at a time, until the magnets endstop detect the metal plate on the tray. Then the magnets are activated and the tray is locked in place to the end-effector.

Name	**Address**	**Payload**	**Description**
Busy	0x50	-	Robot is busy
ReadyToRecieve	0x51	-	Ready for a new task
Stopped	0x52	-	Received stop command
EMC	0x60	-	Emergency stop activated
UpperSafetySwitch	0x61	-	Upper safety switch triggered
LowerSafetySwitch	0x62	-	Lower safety switch triggered
ElavatorLimit	0x63	-	Vertical system hit end stop
LinearbotLimSwitch	0x64	-	Horizontal system hit end stop
Failure	0x65	-	General failure message
Parameters	0x70	Parameters	Return calibration parameters
EncoderOutOfSync	0x80	-	Encoder out of sync
EncoderOutOfRange	0x81	-	Encoder out of range

Table 6.1: Status - Answer from the Arduino controllers

Name	Address	Payload	Description
Move	0x05	Bytes and coordinates	Move the TCP to a specific coordinate
Suction	0x02	-	Engage suction
Stop	0x03	-	Stop all action
Calibrate	0x10	-	Perform calibration
Light	0x11	On/Off	Turn On or Off lights
ChangeLedColor	0x12	[R, G, B]	Changes the colors on LEDs
FindTray	0x14	-	Find the tray
Velocity	0x20	Parameter	Updates velocity parameters
Acceleration	0x21	Parameter	Updates Acceleration parameters
MagnetON	0x22	-	Activate the magnet
MagnetOFF	0x23	-	Deactivate the magnet
StateRequest	0x30	-	Ask for current state
CalibParam	0x31	-	Ask for calibration parameters

Table 6.2: Commando - Tasks for the Arduino controllers to perform

6.3 Robot calibration

For the robot to recognize the working area and create a coordinate system for moving, it has to find the edges and save these as extremity points. This is done by moving each stepper motor inwards until they reach the limit switches which is considered as the home position. When these switches are reached the current position of the motor will be set as origin, in other words, to zero. They will then be incremented one step at a time until the outer limit switches are reached, this is considered as the outer position. The current values for each stepper will then set the maximum points for its respective axis. When the Arduinos has performed the calibration, they return a status containing the calibration parameters to the Odroid.

6.3.1 Vertical robot calibration

While doing the calibration, the vertical robot has an extra task, as the trays are recognized by the IR beams connected to the Arduino Mega (chapter 6.2.5). When it reached the top position, it moves down towards the home position. While doing this it will save the z-positions of the detected trays, these positions are also relayed to the Odroid.

6.4 Robot motion

This chapter describes how the robots axes is moved inside the coordinate system and how it orientates it self.

6.4.1 Distance transformation

To be able to move the robot is it important that all parts of the system has a common way of expressing distance. The camera taking the image of the roe operates in pixels, and the motor controllers operate with steps. As a consequence, converting to a common unit is necessary. In figure 6.10 an illustration of the total conversation and its steps are shown. By knowing the distance from the camera lens down to the tray and the field of view angle, pixels can be measured in mm. How this is done will be explained in chapter 6.6.2. When having the distances in millimeter, it can be converted to steps by dividing the distance in millimeter with the millimeter per step found in chapter 6.4.2 and 6.4.3.

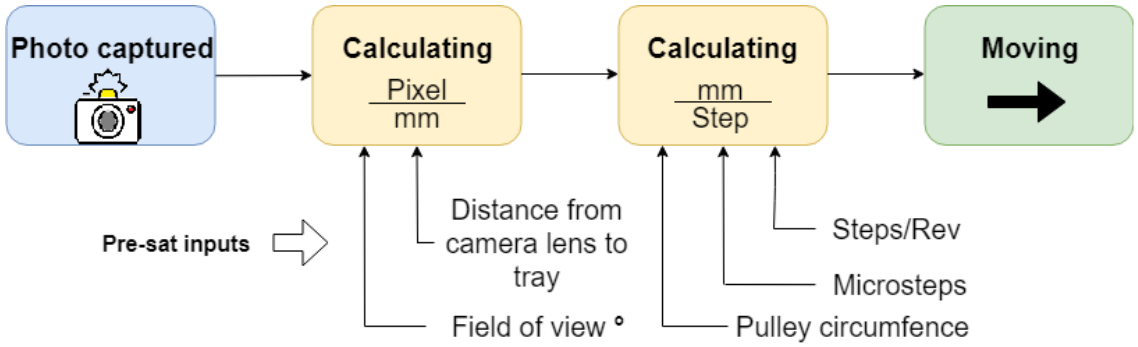


Figure 6.10: Movement of the system

6.4.2 Vertical motion

The vertical system moves one step on both motors at each iteration until it has reached the destination position. Because the horizontal robot is attached to the timing belts inside the vertical robot, one motor have to be inverted. Inverting one motor will make them rotate opposite of each other. This will make the vertical robot able to move the horizontal robot along the z-axis. As explained in chapter 3.2.2, the roe needs to be removed gently, and for that reason high accuracy is desirable while performing z-directional motion. From chapter 5.3.1, the timing pulley designed has a pitch radius of 12.74mm, and the motor drivers are configured with 16 micro steps. The length of one step can be found by dividing the pulley circumference by the total amount of steps for one full revolution. This leads to finding the accuracy of the vertical motion with equation (6.1).

$$acc = \frac{mm}{step} = \frac{D_p \times \pi}{S_{rev} \times f_m} \tag{6.1}$$

$$zAcc = \frac{\pi \times 2 \times 12.74}{200 \times 16} = 0.025 \frac{mm}{step}$$

6.4.3 Horizontal motion

When moving the horizontal system, the positions are counted in steps. A new position will be calculated from millimeter to steps. The coordinate system has maximum values in x- and y-direction which are saved from the calibration. The new position to move, is represented as x-position and y-position in steps. To move from the current positions to the new position as smooth as possible, the calculations are based on the Bresenham Line Algorithm (chapter 2.11). Its basic operation is used to calculate how many steps the x- and y-direction stepper motors should move in relation to each other. The delta x and delta y is calculated. If delta is zero the robot is in position and should not move. If delta is not zero, check which delta is the biggest and calculate the slope based on this. As the controller operate with integers, the slope is rounded and multiplied with the moving direction. Multiplication is done with the direction that has the biggest delta and by that move the most.

The theoretical accuracy has been calculated from the pulley pitch circumference, divided by the total steps per revolution of a motor. Both x- and y-axis motors have 200 steps per revolution (S_rev) and configured with 4 micro steps (f_m). This configuration leads to 800 steps per revolution. The pitch diameter provided by x-axis is 12.73mm and for the y-axis is 10.186mm. With these values and equation (6.1) we can find the length travelled for each step.
For the x-axis, mm per step will be:

$$xAcc = \frac{\pi \times 12.73}{200 \times 4} = 0.05 \frac{mm}{step}$$

For the y-axis, mm per step will be:

$$yAcc = \frac{\pi \times 10.186}{200 \times 4} = 0.04 \frac{mm}{step}$$

6.5 Tool center point

The straw on the end-effector is configured as the robots tool center point (TCP). This was chosen because this is the only point of the end-effector that will be in touch with the test medium(roe). As the robot operates in a Cartesian coordinate system, controlling the whereabouts of the TCP is relatively simple compared with other robots (chapter 2.2.1). A challenge that gives the controlling of the TCP another dimension of difficulty, is that it needs to be controlled relative to signals from the IR-beams, camera and placement of the tray opening mechanism. These components are all mounted at different heights with offsets from the TCP.

6.5.1 IR-beam offset

The IR-beams are used for avoiding conflict between the moving robot and objects inside the working area of the robot. Thus the z-axis offset between the beams and other components need to be at least the length of what is defined as an obstacle. Because the lower IR-Beam is used for detecting positions of the trays. This makes it necessary to know the z-distance between the TCP and the lower IR-beam for calculating the height that the robot needs to have for opening and operating on

trays. The distance between the TCP and trigging point of the lower IR-beam was measured by operating the robot downwards in z-direction until the lower IR-beam was interfered by an obstacle (tray). Then measured the distance between the top of the tray and the TCP with a ruler.
This offset is illustrated in figure 6.11.

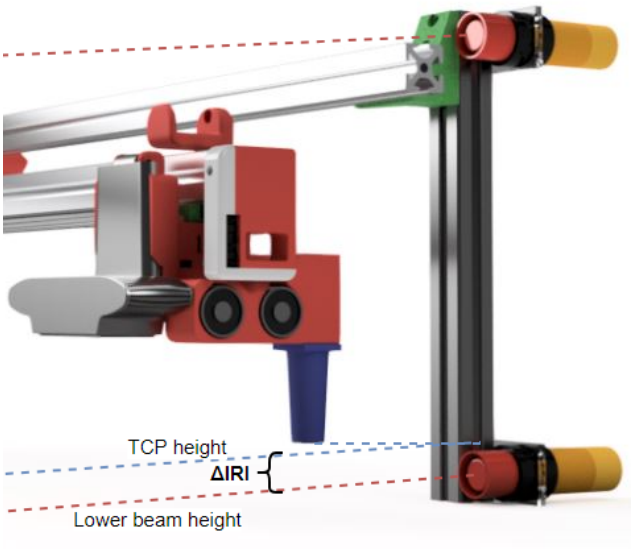


Figure 6.11: Offset from IR-beams to TCP

$\Delta IRl =$ Distance between lower IR-beam and TCP

6.5.2 Magnet offset

Placement of the tray opening mechanism is also necessary to know relative to the TCP. The fact that the distance between the magnets and TCP is fixed makes it simpler to operate. The magnet offset is illustrated in figure 6.12.

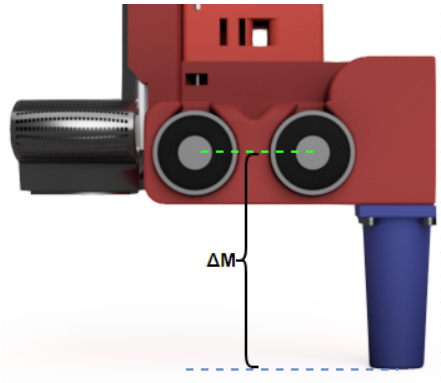


Figure 6.12: Offset from magnets to TCP

$\Delta M =$ Distance between magnet center and TCP

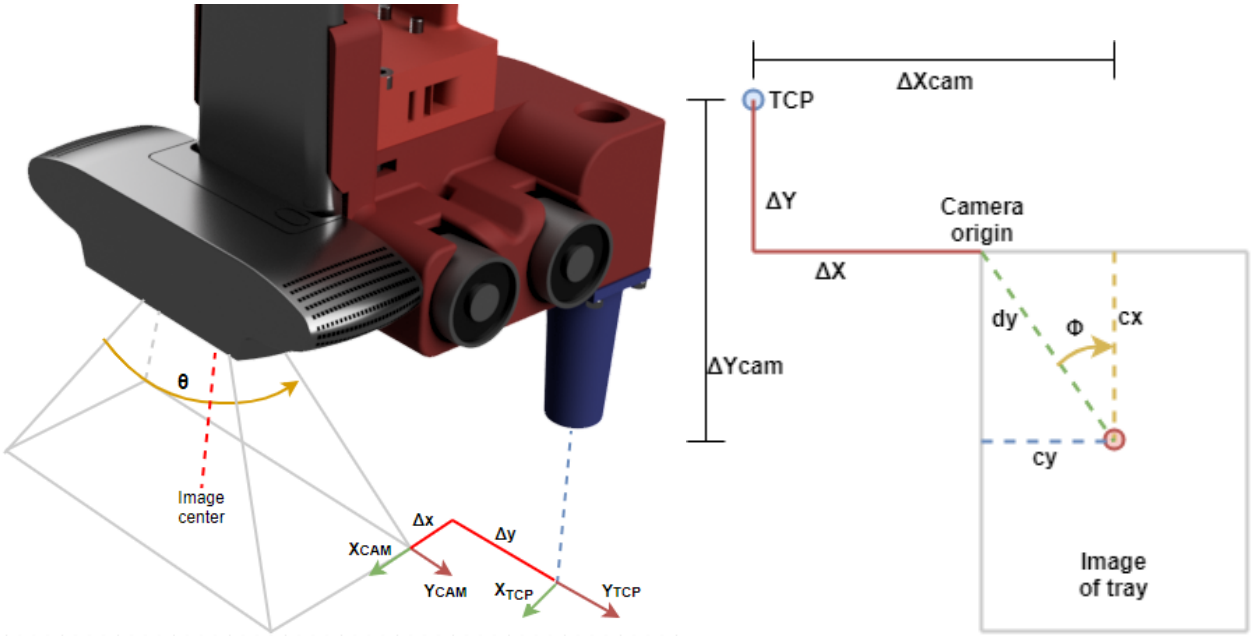
6.5.3 Camera offset

The offset that needs most calculation for being solved relative to the TCP, is the coordinates being provided from the image processing. These coordinates are calculated relative to the image origin. The offset between the image origin and the TCP in x- and y-direction changes linearly when the distance between the camera lens and surface of the tray changes. This distance needs to be calculated for providing coordinates of the dead roe that is calculated to the TCP coordinate system.

(a) Camera offset x- and y-direction

(b) Camera offset z-direction

Figure 6.13: Camera TCP offsets x-, y- and z-direction



(a) Image offset x- and y-direction

(b) Horizontal plane illustrated

Figure 6.14: Image origin to TCP offsets x- and y-direction

As illustrated in figure 6.14a and 6.14b, the distances that needs to be calculated is Δx and Δy. This calculation can be done by using the formulas from chapter 2.8.1 and the offsets between camera and TCP, illustrated in figure 6.13a and 6.13b.

The first calculation that needs to be done is finding the diagonal length from the center of the image to the camera origin (dy). This is done by using equation 2.6.

$$dy = dx \times \tan\frac{\theta}{2}$$

Then we calculate the angle ϕ between dy and cx with the equation 2.7.

$$\phi = \arctan\Delta = \arctan cy/cx$$

Next step is calculating the distances in cx and cy from the image center to the image origin, shown in equation 2.8 and 2.9.

$$cx = \cos\phi \times dy$$

$$cy = \sin\phi \times dy$$

As the distance between the image center and TCP in x- and y-direction is fixed and known, we can use these for calculating the position of the origin relative to the TCP.

$$\Delta x = \Delta Xcam - cy\ mm$$
$$\Delta y = -(\Delta Ycam - cx)\ mm$$

$\Delta Xcam =$ Offset from TCP to lens center in x-direction (78.7 mm)
$\Delta Ycam =$ Offset from TCP to lens center in y-direction (34.6 mm)
$\Delta Zcam =$ Offset from TCP to lens center in z-direction (43.2 mm)
$\Delta x =$ Offset from TCP to image origin in x-direction
$\Delta y =$ Offset from TCP to image origin in y-direction
$\Delta z =$ Distance between camera lens and surface of tray

When the end-effector of the robot shall remove a roe that has been detected by the image processing it is necessary to add the Δx- and Δy-offsets to the x- and y-coordinate of the roe.

6.6 Machine vision

Machine vision is implemented in the project as means to locate the dead roe in the hatcherys trays. This implementation includes mounting a camera on the robot, mounting lights used for illuminating the surface to be captured, and performing image processing for finding dead roe in the trays.

6.6.1 Camera

Placing the camera on the end-effector was chosen as this provides the ability of moving the camera in all the degrees of freedom provided by the robot. Thus it is possible to position the camera in various positions that are desired for each picture to be captured. How the camera is mounted can be seen in figure 6.15.

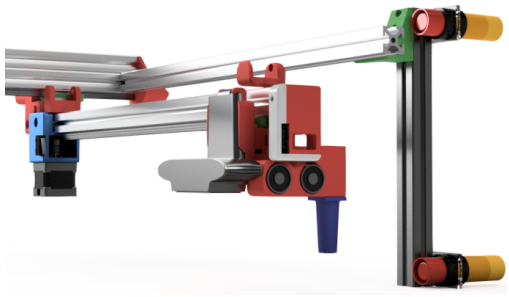


Figure 6.15: Camera placement

6.6.2 Environment

Figure 6.16 shows the robot with the mounted led strips emitting light. These are addressable RGB light strips, and are controlled by the Arduino Nano. With the red, green and blue light spectrums, we can control each individually. With the Nano controller it is possible to adjust each color in 256 steps, where 0 is off, and 255 is fully on.

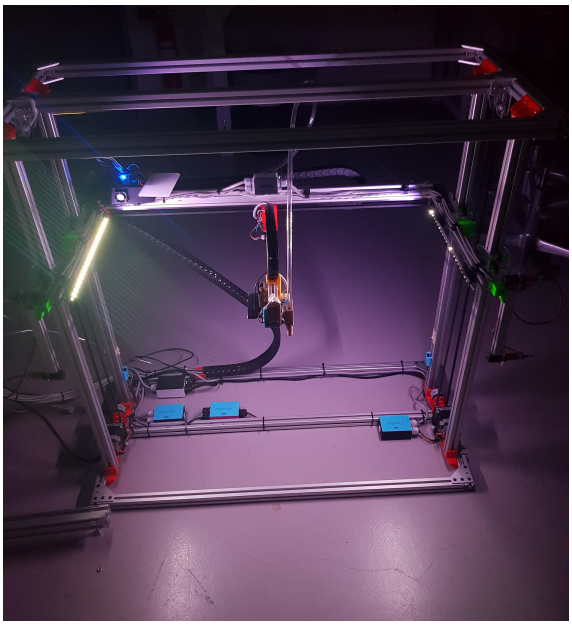


Figure 6.16: Robot with lights

The robots height limits the ability to take overview pictures of the trays, the distance between camera lens and surface can at most be 80 mm. Below is a calculation of the cameras FOV.

Calculate FOV: Calculate the diagonal length from the center of the image to the upper corner. This is done by using equation (2.6).

$$dy = 80mm \times \tan\frac{78^\circ}{2} = 64.78\ mm$$

Then calculate the angle ϕ between the diagonal and the baseline with equation (2.7).

$$\phi = \arctan\frac{1080}{1920} = 29.36^\circ$$

Next step is calculating the total width and height of the image being captured. This is done with equation (2.8) and (2.9). In this calculation the image will be relative to the tray. Therefor, the height will be $Cy = 2 \times cx$ and the width will be $Cx = 2 \times cy$.

$$Cy = 2 \times cx = 2 \times (\cos 29.36^\circ \times 64.78) = 112.92\ mm$$

$$Cx = 2 \times cy = 2 \times (\sin 29.36^\circ \times 64.78) = 63.52\ mm$$

As found in the calculation done above, the size of the captured area will be $112.92mm \times 63.52mm$. By knowing that the tray size where the roe is stored ($650mm \times 200mm$, see chapter 4.13) is larger than the area captured in a picture, it is necessary to capture several images for covering the entire tray.

Image layout is configured in order to capture images from the entire surface of the trays where the roe is stored. As illustrated in figure 6.17, the images are captured ”sideways”, as the camera is mounted on the robots end-effector in this direction. Since one image covers $112.92 \times 63.52mm$ of the trays surface, several more images are needed to cover the trays surface being 650 x 200 mm, as explained 4.13. It is desirable to capture neighbour images with a bit overlap preventing roe from being neglected by the image processing if they are only partially visible in the image. The size of a roe varies from 5 mm to 9 mm, thus the overlap between images should be more than 9 mm.

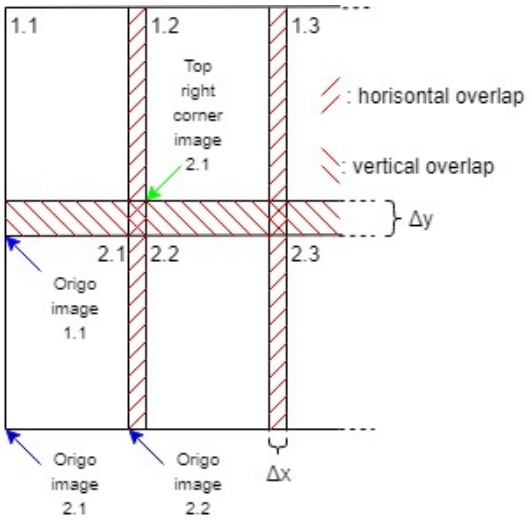


Figure 6.17: Image layout

Calculate number of images in x- and y-direction:
We start by defining the minimum size of a overlap, (size of the largest roe).

$$\Delta x \geq sizeofroe = \Delta x \geq 9\ mm$$

Then we calculate the number of images needed with the minimum overlap size, must be rounded up to closest integer for covering entire tray.

$$NrImgsX = \frac{Tx}{Cx - sizeofroe} = \frac{650}{63.52 - 9} = 11.92 \approx 12$$

Next step is calculating the overlap that we get with the number of images found in the equation above.

$$\Delta x = \frac{Cx \times (NrImgsX) - Tx}{NrImgsX - 1} = \frac{63.52 \times 12 - 650}{13 - 1} = 10.2\ mm$$

Last step is verifying that the overlap calculated is equal to or greater than the minimum required overlap.

$$\Delta x = 10.2mm \geq 9\ mm \quad \rightarrow \quad OK$$

Go through the same procedure with y-direction as for x-direction. We start by defining the minimum size of a overlap (size of the largest roe).

$$\Delta y \geq sizeofroe = \Delta y \geq 9\ mm$$

Then calculate the number of images needed with the minimum overlap size, must be rounded up to closest integer for covering entire tray.

$$NrImgsY = \frac{Ty}{Cy - sizeofroe} = \frac{200}{112.92 - 9} = 1.92 \approx 2$$

Next step is calculating the overlap that we get with the number of images found in the equation above.

$$\Delta y = \frac{Cy \times (NrImgsY) - Ty}{NrImgsY - 1} = \frac{112.92 \times 2 - 200}{2 - 1} = 25.84\ mm$$

Last step is verifying that the overlap calculated above.

$$\Delta y = 25.84mm \geq 9\ mm \quad \rightarrow \quad OK$$

$\Delta x =$ Overlap x-direction
$\Delta y =$ Overlap y-direction
$Tx =$ Tray size x-direction (650 mm)
$Ty =$ Tray size y-direction (200 mm)
$Cx =$ Image size x-direction (63.52 mm)(rotated)
$Cy =$ Image size y-direction (112.92 mm)(rotated)

The result of the calculations above, shows that in order to cover the entire area where the roe is stored it is necessary to capture 12 images in x-direction and 2 in y-direction. This makes for a total of 24 images. With the number of images being 12 x 2, the overlap in x-direction becomes 10.2 mm and 25.84 mm in y-direction. As both of these distances are greater than the maximal size of a imitated roe (9mm) they are suited for the application.

6.7 Image processing development

During the project there has been developed two image processing techniques. The first technique is meant for locating dead roe in images captured of roe in a hatchery. The second image processing is developed as a means to test the robot we are developing. This image processing technique tries to locate the white plastic bullets amongst the orange water balls that is our test medium (chapter 4.13.1).

6.7.1 Roe

The following image processing is developed in MatLab with the DIPimage toolbox (see chapter 4.14.2).

Images captured at a Marine Harvest hatchery, with a Canon EOS 1000 SLR camera without flash. The ceiling lights were dimmed and capturing images with glare from the water surface was carefully avoided, as this will block vision beneath the surface. Figure 6.18 displays a section of a tray in the hatchery filled with roe. The two distinct white-yellow roe in the middle of the picture are dead, the red and transparent-red ones are alive. With this information, we can assume that we can distinguish the dead from the living by analyzing their color.

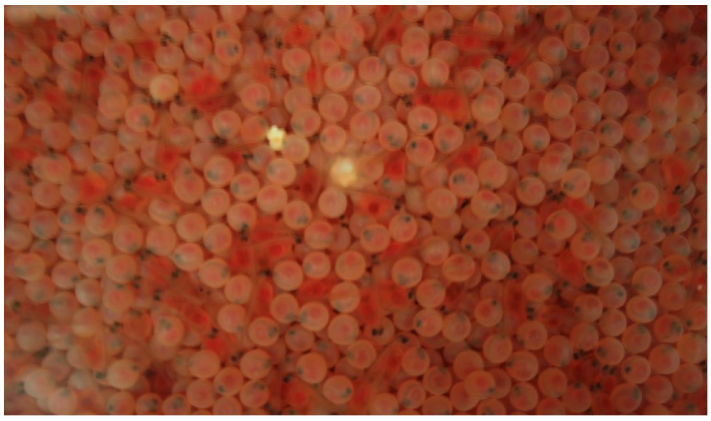


Figure 6.18: Original RGB image

Gray-scale transformation of RGB image. In figure 6.19 one can see that the dead roe are still standing out with a different gray level than the rest.

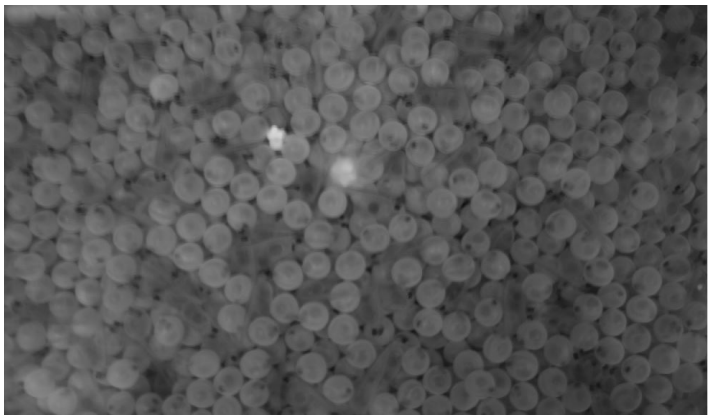


Figure 6.19: Gray image

Binarizing with adaptive technique finds the pixels that stands out in value in local sections of the image, these pixels are set to binary high(1), the rest are set to binary low (0). The binary image 6.20 displays two large binary objects, by comparing the binary image to the gray image we see that these objects are the dead roe. By looking closely, you can see a small binary object in the middle of the right edge of the image. By comparing the location of this object to the same location on the RGB image, we see that this is glare from light that reflects on the surface of a living roe. As we are not interested in removing living roe we need to remove this artifact from the image.

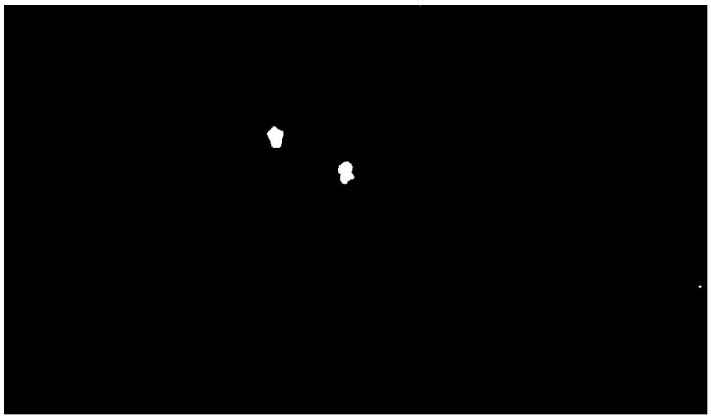


Figure 6.20: Binary image

Erode the binary image with a circular structure element. In figure 6.21 we can see that the binary objects from the dead roe has decreased in size and the smaller binary object from the glare has disappeared. This is the desired effect of erosion (see section 2.9.2).

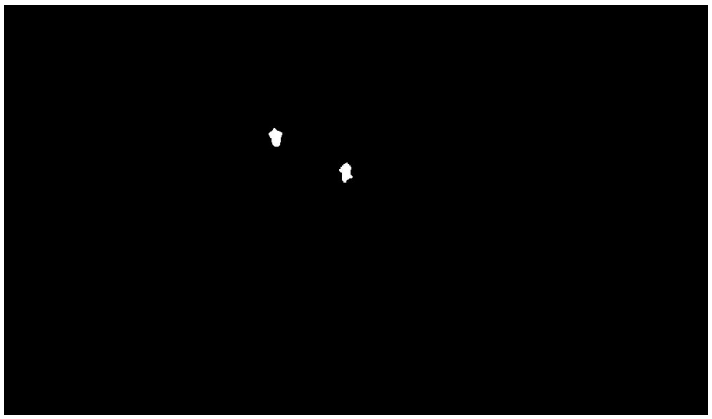


Figure 6.21: Eroded binary image

Dilate the eroded image for restoring the size of the binary objects from the roe. The structure element used for dilating is the same as the one used for erosion, this makes for the objects being restored to a shape as equal to the original as possible, seen in figure 6.22.

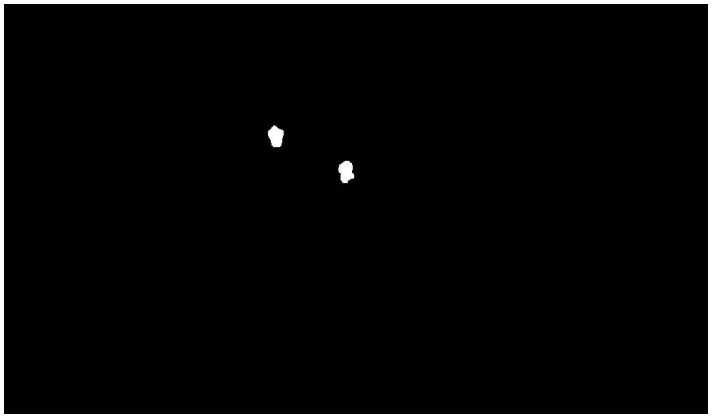


Figure 6.22: Dilated

Finding center of the dead roe position in the picture is done by labeling the dilated image making each of the binary objects into a dedicated object with properties such as area, circumference, orientation, center of gravity and more. Properties are extracted consisting of the x- and y-coordinates of the centroids, see figure 6.23. These coordinates are with respect to origin of image being in the top left corner. The operation that labels the blob's starts at the left side of the image and goes to the right. Thus, the order of the coordinates provided starts with the leftmost and so in the order they are to the right.

```
centroids =

  569.4124  276.9908
  715.8964  350.0609
```

Figure 6.23: Table of centroids from image processing

6.7.2 Roe imitation

The following image processing is developed in MatLab with the DIPimage toolbox and then implemented in Java with the open CV library. The following images are from the image processing done in Java.

Captured image with the camera mounted on the robots tool, image displayed in figure 6.24. The illumination of the surface is from the led strips on the robot with parameters decided from the test, described in chapter 7.3.

Figure 6.24: Original RGB image

From figure 6.25 one can see that there are five distinct white circles. With a closer look one can see three circles in the upper left corner. These three are dead roe that lays beneath the living and are therefore hard to spot. The white objects in the lower left corner is glare from the light breaking the water surface.

Gray-scale transformation of RGB image.

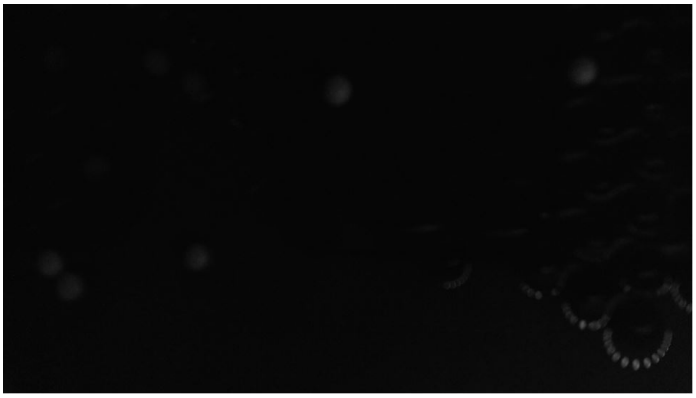


Figure 6.25: Gray image

Binarizing of gray-scale image. Adaptive and manual thresholding techniques were tested for seeing which technique yielded the best result.

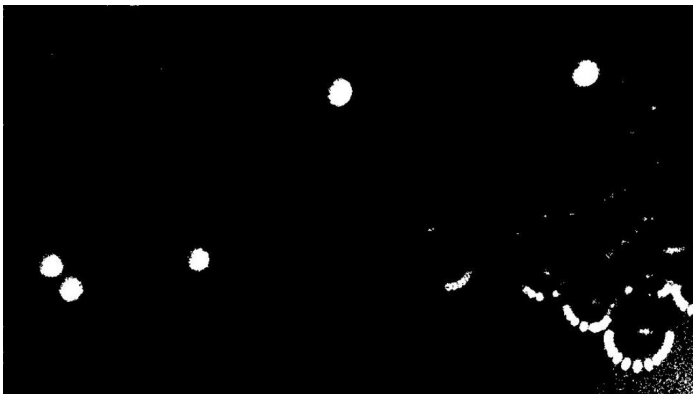


Figure 6.26: Manual binarizing

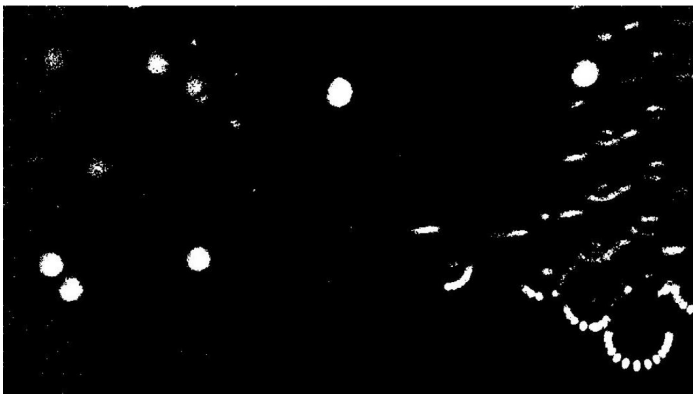


Figure 6.27: Adaptive binarizing

The images above shows the different results from using manual and adaptive thresholding methods (chapter 2.9.1. The manual thresholding (figure 6.26) yields a result where all pixels from the gray image with a value above a preset limit are set as logical high. This results in finding the most distinctive white circles, and some of the glare from the light. The result of the adaptive thresholding (figure 6.27) reveals the same objects as the manual thresholding and in addition we can see that the tree circles in top left corner are discovered as well. A drawback of the adaptive thresholding is the discovery of the additional light glare.

We choose the adaptive binarized image for further use in the image processing as this finds the most of the roe in the picture. It is likely that the glare from this image is easier to remove as the glare objects are smaller than the glare discovered in the manually binarized image.

Erode binary image for removing undesired artifacts from the image. A circular structure element was used for eroding. Figure 6.28 displays the eroded binary image.

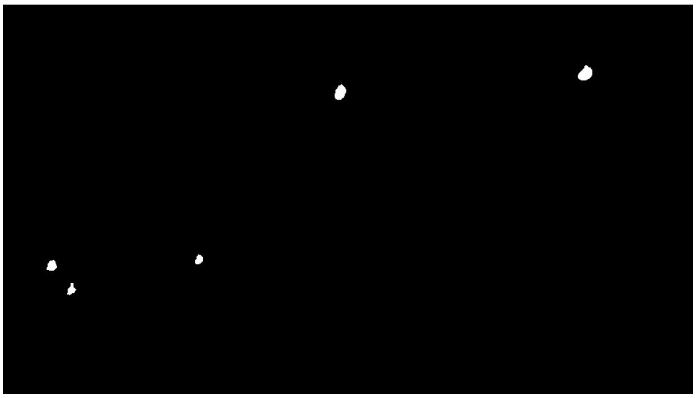


Figure 6.28: Eroded image

Dilate the eroded image for restoring the size of the objects in the image, see figure 6.29. The same structure element as used for eroding was applied for achieving a result where the objects are as equal to those in the original image.

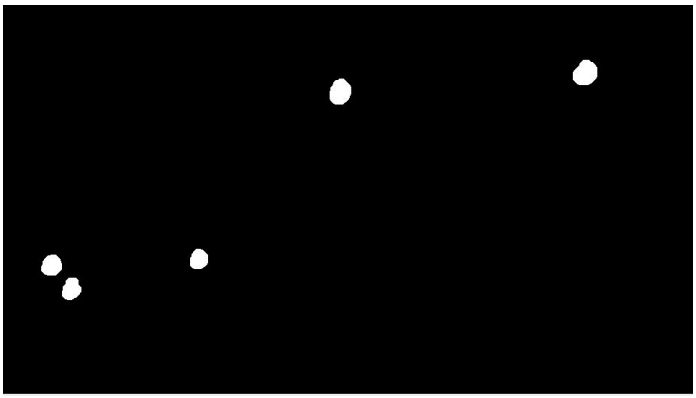


Figure 6.29: Dilated image

Label blobs in the image for making it possible to get the properties of each blob.

Extract properties from the labeled image. A list with x- and y-coordinates of the blobs centroids is returned.

6.8 Path optimization

Optimization was implemented to reduce the time used on removing dead roe. From the image processing (chapter 6.7), a coordinate list of dead roe is returned. This list is arranged from the first to the last dead roe located. Two algorithms which sorts the destinations to reduce cost are made. Our cost is the time used for horizontal movement and the time used for processing. Both algorithms are implemented and the one which yields the lowest cost will be used.

6.8.1 Nearest neighbor

A nearest neighbor algorithm were made for decreasing the cost. The basic idea of the algorithm is to chose a start destination, and from this point find the nearest destination. The start coordinate in this system is the first destination in the list. The application creates a new list that contains all destinations and makes a clone of the original list. First we add the start coordinate to the new list and remove it from the cloned list. From the start coordinate, we calculate the distance to all coordinates in the cloned list, and adds the closest one to the new list. The added coordinate is set as start coordinate and are removed from the cloned list. This process is repeated until the new list contains all destinations. Then the list can be returned.

6.8.2 Genetic algorithm

We have implemented a genetic algorithm (chapter 2.10.3) that rearranges the coordinate list in a way they can be removed as fast as possible. The new list will be given by our fittest chromosome. Because our robot will be in the position where it took the last picture, we will use this position as the start coordinate. This start coordinate will always be the first coordinate in our rearranged list, and the total distance of a pattern will be calculated from this point. First we made a random pattern optimizer. This will create a population for a GA to evolve. But sometimes it can be used as a stand-alone optimizer as well. It generates a population with N numbers of chromosomes holding a list of genes represented as coordinates. Each chromosome represents one possible solution that is random generated and keeps the start coordinate as its first. After all chromosomes are generated, the fittest pattern can be returned. Figure 6.30 illustrates the sequence.

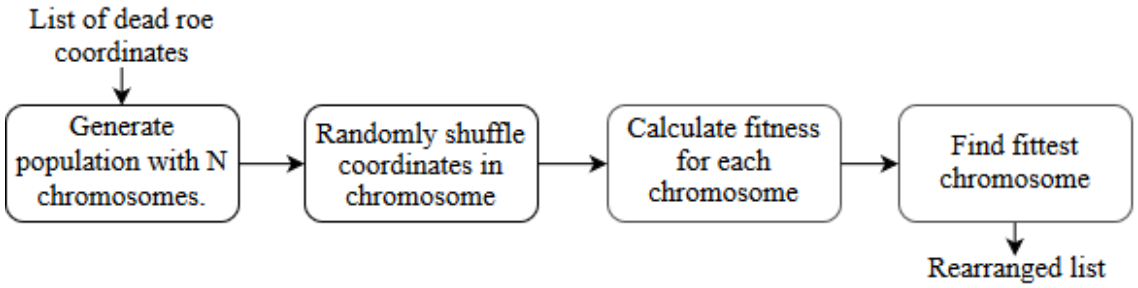


Figure 6.30: Random pattern optimizer flow chart

The genetic algorithm starts by creating a new empty population. Then we select a set of parent chromosomes for mating using a method called tournament. These two parents uses an arranged crossover method for breeding two children. First the start coordinate is added to the child. Secondly two random numbers representing the start and end crossover positions are made. All the coordinates within this range in parent one will be added to the child at the same positions. Then the missing coordinates are assigned from parent two. Crossover is repeated to create child number two as well. When two children are created they are added to the new population. The mating process is repeated until the size off the new populations is the same as the old population. To make sure all possible combinations can be achieved, all chromosomes in the new population can be mutated with the possibility of P_m. Mutation is done by flipping the position of two random coordinates in a chromosome. Training stops after termination criterion is satisfied. The criterion is for K number of generations or if no improvement have been achieved for ten generations. A sequence diagram for the genetic algorithm is shown in figure 6.31

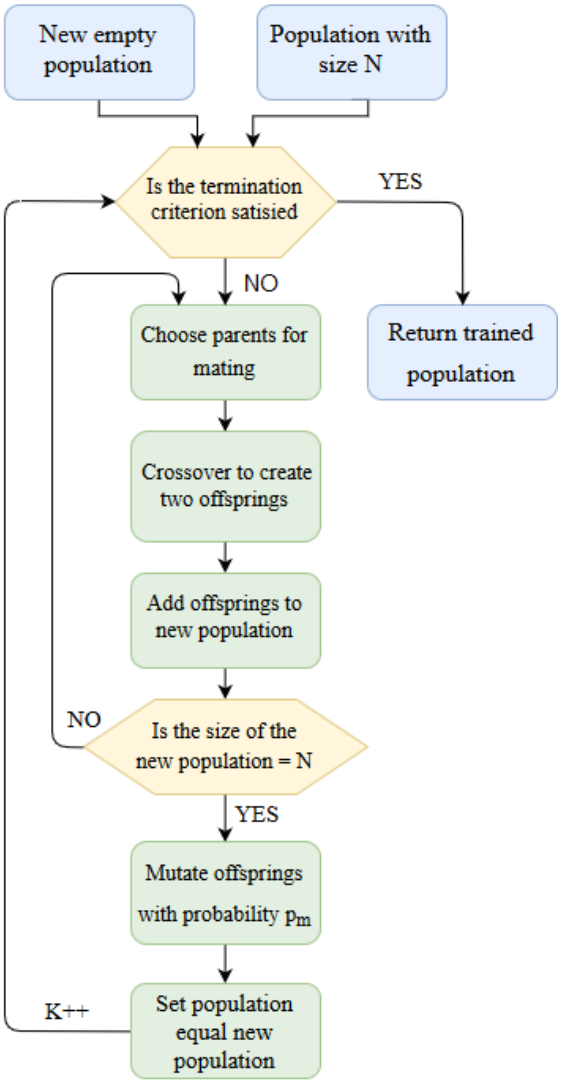


Figure 6.31: Genetic algorithm flow chart

For increasing the result of the GA, elitism can be used. Elitism makes it possible to retain the two best chromosome from a generations and add them to the new populations without performing crossover and mutation.

6.9 Tray handling mechanism

The horizontal system handles the opening and closing of trays by pulling it out in y-direction. Electromagnets mounted at the end-effector are used to attach itself to the metal plate on the tray(figure 6.32). The robot will connect the magnets to the desired tray and gently pull it out from standby (in) to working (out) position. The tray is released by deactivating the magnets when working position is reached.
For detecting the metal plates it hit, the magnets housings are used as a limit switch. The two housings are connected to an electrical pull-up circuit illustrated in figure 6.33. When the two magnets hits the metal plate at the same time, they will function as a closed switch and activate the controller input. This will tell the robot that a tray is located, thus eliminating the procedure of having to move the end-effector with an offset for opening a tray.

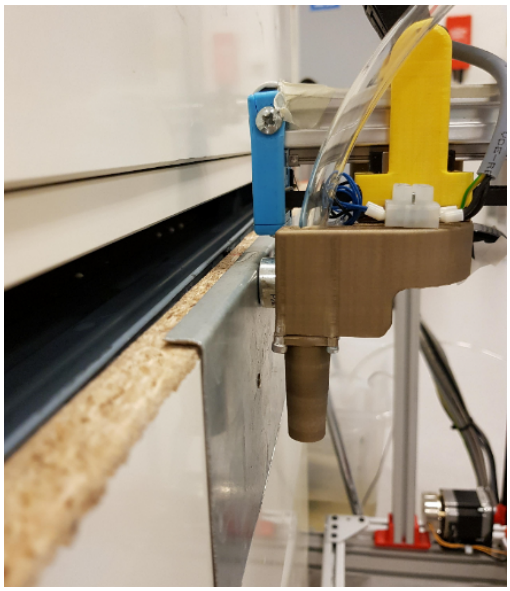


Figure 6.32: Tray handling mechanism

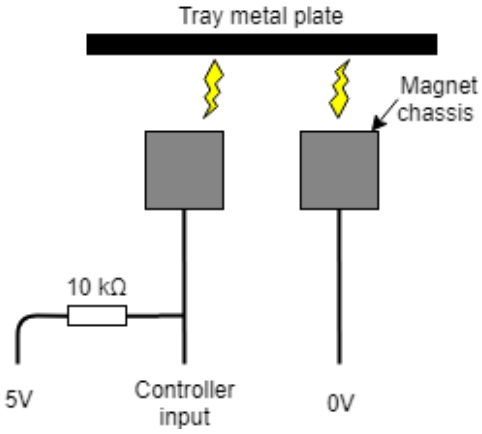


Figure 6.33: Magnet switch

6.10 Roe removal

The chosen method for removing the roe is by suction. For creating suction we chose to generate vacuum with a venturi chamber (chapter 2.6). By the test described in chapter 7.1, we found that the medium best suited for generating the vacuum for this application, is water. For generating vacuum we use an electrical pump that provides the venturi chamber with motive fluid. The pump and venturi chamber are both submerged in a box with water, a PVC tube is connected to the chambers vacuum port as illustrated in figure 6.35. The other end of the PVC tube connected to the vacuum port is fixed in the straw on the robots end-effector, as shown in figure 6.34. Controlling the suction is done by turning on and off the supply to the electrical pump.

Figure 6.34: Vacuum tube mounted to TCP

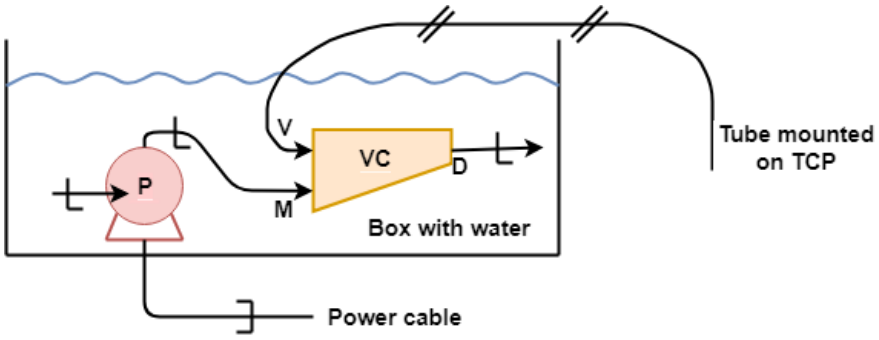


Figure 6.35: Suction process diagram

$P =$ Pump
$VC =$ Venturi Chamber
$V =$ Vacuum port
$M =$ Motive port
$D =$ Discharge port

6.11 Relays

Relays are used for controlling equipment connected to the 12V and 24V supply with the 5V I/O on the Arduino Mega. They work both in output for controlling, and as input by connecting the sensors to control the relays. The relays needs to have the specifications as desired voltage and amount of contact sets for their intended purpose.

6.12 Emergency stop

A rule of thumb regarding emergency stop is that it should be as little software related as possible, as emergency stop should not relay on software to do its job. The emergency stop controls relays which in turn are connected to the following components:

- Supply to the Arduino Nano shield(-SC0, horizontal system). → Horizontal system stops immediately.
- Supply to the drivers for the vertical system(-DM0, -DM1) motors → Vertical system stops immediately.
- Supply to the brakes of vertical motors → Activates brakes, prevents horizontal system from falling to the ground.
- Signal to the Odroid → Alerts the software system.

6.13 Graphical user interface

A graphical user interface (GUI) has been developed. This interface will provide the user with option for controlling the robot. The GUI was created with Swing, which is an embedded library in java. When creating a GUI, the priorities was functionality and that it was easy to use.

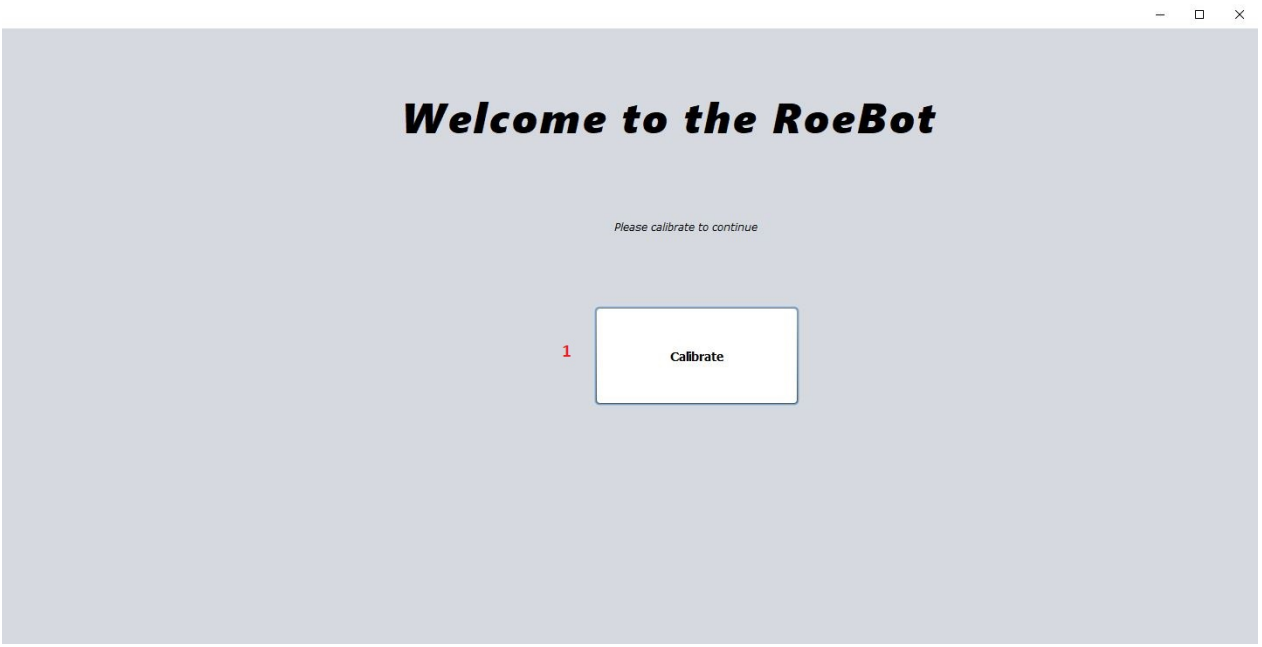


Figure 6.36: Graphical user interface - calibration

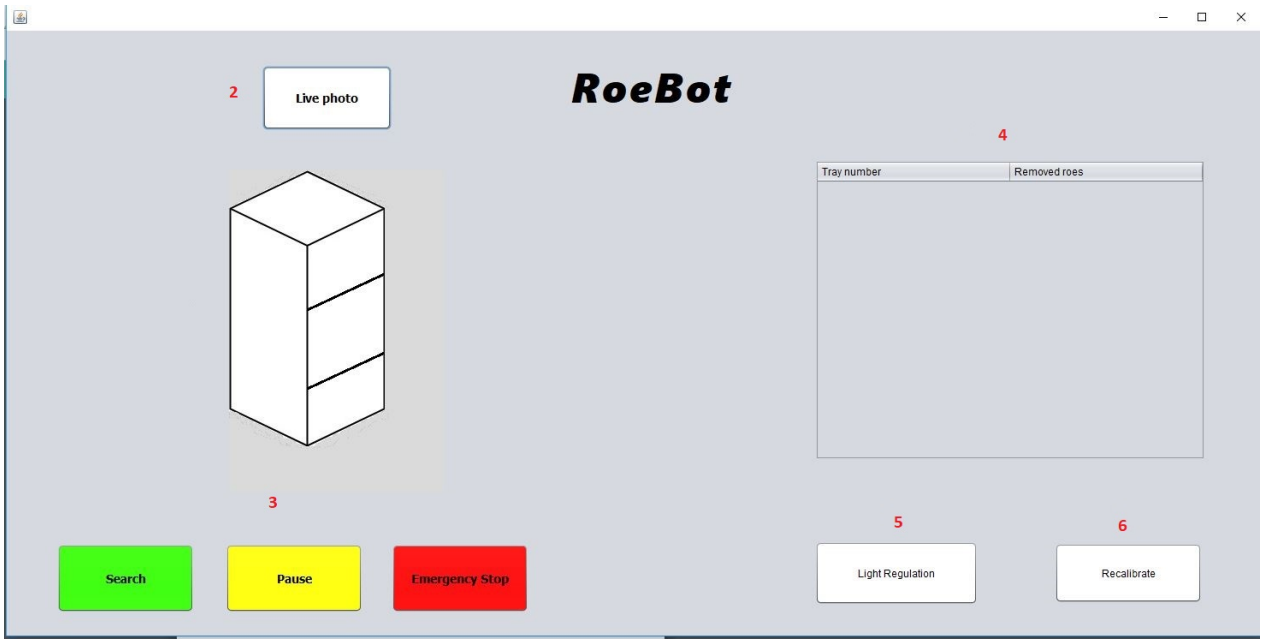


Figure 6.37: Graphical user interface - main frame

The GUI is composed mainly of two frames. The two frames are shown in figure 6.36 and 6.37. The first frame will occur when the program starts. Because the robot always needs to be calibrated before starting, this frame only possesses the opportunity to calibrate the robot. After the robot has finished the calibration, the GUI will change frame to the main frame. In this frame, the user will be able to control the robot and read statuses. In addition, this frame provides an illustration of which tray the robot currently are open. The different components int there frame are listed beneath with corresponding numbers as in the figures.

1. *Calibrate*. This will make the robot find its extremities in each axis, and locate position of trays.

2. *Live photo*. Changes the panel below between tray illustration and pictures from the camera.

3. *Search, pause and stop buttons*. Search button starts the robot, making it locate and remove dead roe. The pause button stops the robot in the current position. After it is paused the description will change to continue and can be used to start the robot where it left of. Stop button stops the robot and if it is working in a tray, it closes the tray before moving to home position.

4. *Table* contains number of removed roe in each tray. Since this robot do not verify if a roe is removed, this numbers will be the number of performed suctions in a tray.

5. Opens *light regulation* that independently controls the colors of the lights mounted on the robot. By dragging the spherical objects up or down, the light levels will either increase or decrease (figure 6.38).

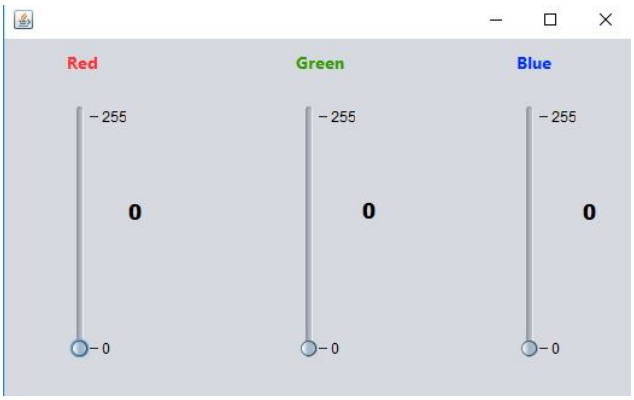


Figure 6.38: Light regulation panel

6. *Recalibrating*. Pressing this button will make the robot recalibrate itself.

Chapter 7

Reviews

This chapter elaborates the tests and reviews performed during the project, and how they affected the final result.

7.1 Venturi generator testing

As mentioned in chapter 4.6 the venturi generator is compatible with various motive mediums. Thus we performed a test, revealing what medium is the best suited for this application. The mediums that were tested, are compressed air and water.

Air requires smaller chambers and more pressure to create the vacuum than *water*. While testing, we found that a lot more pressure were needed when we had longer a hose to get the vacuum needed on the other end of the hose. We found that the delay from turning off the pressure on the venturi chamber, to the vacuum disappearing were almost a whole second. This is caused by the air being compressed on the input and thereof continues to expand after the compressor output(air flow on the input) is turned off. This was unfortunate, as the vacuum should be short and precise.

Water as medium to create vacuum requires a different design on the venturi chamber than the *air* design. Bigger pathways for the water is needed but the input pathway on the vacuum needs to the same. To flow water through the chamber water pump was used. The whole chamber and pump is enclosed in water. We found our pump made a bit greater vacuum then our air compressor solution. The delay from which the pump was turned off and the vacuum disappearing in the tube was as far as we could measure, instantly. Water performed with minimal delay and responsive control.

7.2 Timing belt stretch test

A stretch test of the 6mm and 9mm timing belts was done to check how much weight the belts can carry without being stretched or snap. If a belt is stretched, it might cause the belt to slip on the timing pulley. This can make the robot disoriented or in worst case damage the robot and its surroundings. The stretch test was performed with a Galdabini QUASAR 200 stretch test machine [73].

Figure 7.1: Timing belt stretch test

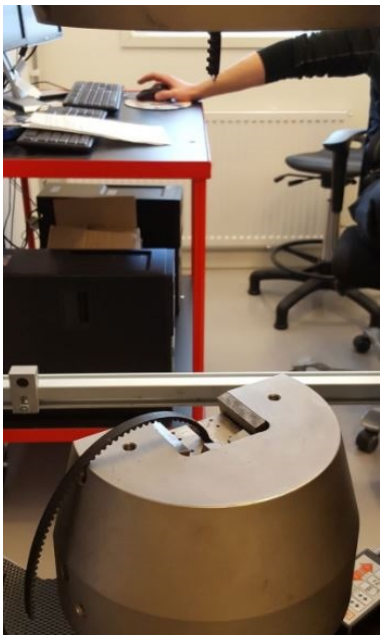


Figure 7.2: Timing belt stretch test snapped

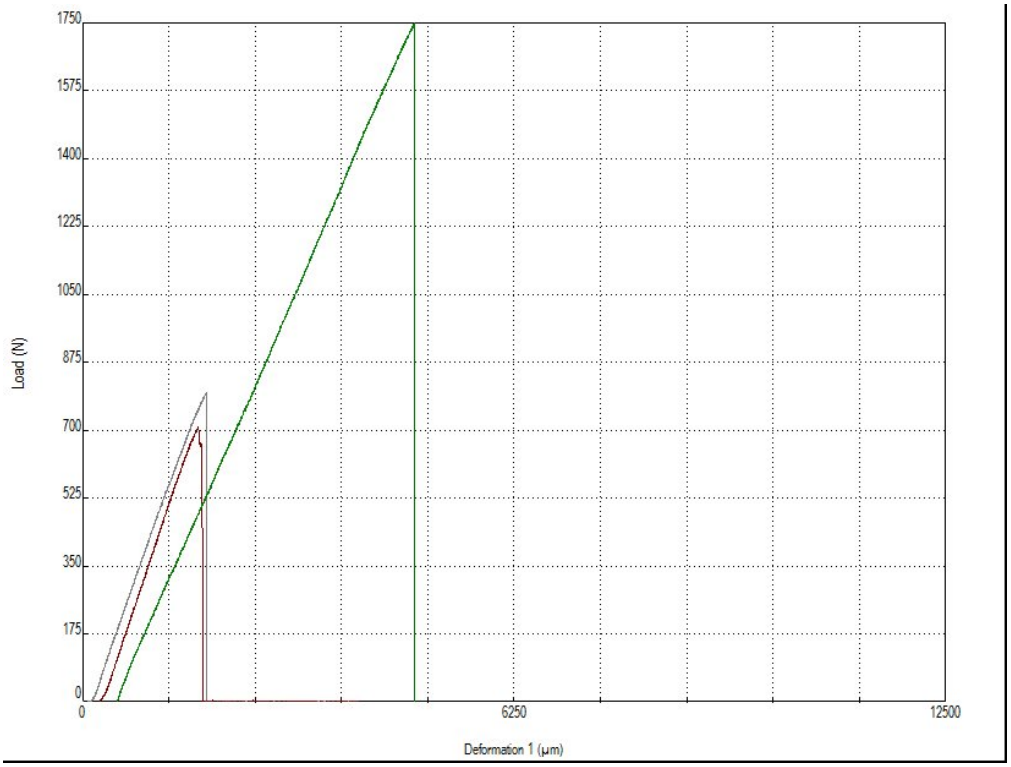


Figure 7.3: Timing belt stretch test results

Figure 7.3 shows the stretch test of the two different timing belts that are used on the robot. The 6 mm GT2 belt was tested two times as it in the first test snapped in the jaw of the test jig, the first test is the red graph, the second test is the gray graph and the green graph is the result of stretch testing the 9mm HTDM 5 band. From the test, the amount of supplied force needed before the belts ruptured can be found. Using Newtons second law for calculating the snapping force from Newton to kilograms.

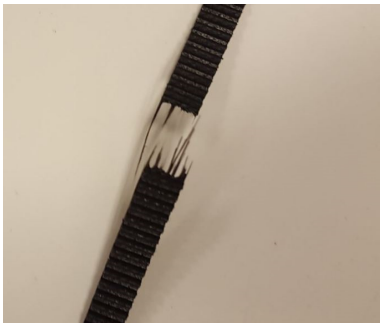


Figure 7.4: 9mm HTDM5 belt snapped

Newtons 2. law.

$$F = m \times a$$
$$m = \frac{F}{a} \tag{7.1}$$

$F =$ Force (Newton N)
$m =$ Mass (kilograms kg)
$a =$ Acceleration $9.81 \frac{m}{s^2}$

Table 7.1 holds the values read from figure 7.3 and the values calculated with equation (7.1). As can be seen from the table, the 6mm timing belt can withstand at least 71.4 kg, and the 9mm band is able of withstanding 176.9 kg.

Belt dimension (mm)	**Elongation (mm)**	**Snapping force (N)**	**Snapping force (kg)**
6 (red)	Approx. 1.72	Approx. 700	Approx. 71.4
6 (gray)	Approx. 1.80	Approx. 784	Approx. 79.9
9 (green)	Approx. 4.77	Approx. 1736	Approx. 176.9

Table 7.1: Timing belt stretch test

7.3 Illumination test

Illumination is needed for being able to detect anything from the images in the processing. On the other hand, the roe and fry gets stressed when being exposed of light (chapter 2.1.2). Thus, it is necessary to keep the illumination as low as possible, and still be able to find the dead roe in the trays. The test for finding the best level of illumination was performed in a dark room, with the lights mounted on the robot, as shown in chapter 6.6.2. The colors being manipulated with for detecting the white plastic bullets, was green and blue, as the red color would make the red roe stand out as well. Several combinations with the lower bounds of the green and blue specters was tested. As expected, when the light got darker, it was harder to see the plastic bullet in the images (figure 7.5). What was not expected, was that when the light levels got close to zero, it made the

bullet easier to find, as there was almost no reflection from other surfaces (figure 7.6). Thereof we decided for using [0, 2, 4] as illumination level for the application. The color combinations lower than [0, 2, 4] made it impossible to locate the plastic bullets in the middle of the tray, as these are furthest from the light source.

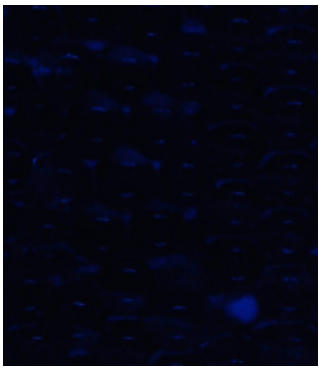


Figure 7.5: Illumination [0, 3, 4]

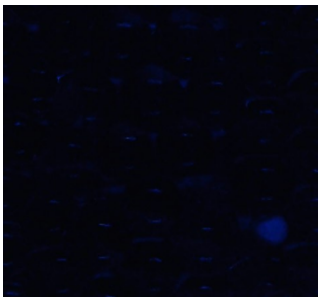


Figure 7.6: Illumination [0, 2, 4]

7.4 Electrical cabinet testing

Before testing the robot, the wiring of the electrical cabinet and the components on the robot needed to be controlled. This ensured that the wiring was done correctly and that there were no short circuits that would damage the electrical components. The testing was done with a multimeter measuring the resistance between different points in the circuits. After the wiring was checked we applied the voltage supply and measured that all of the different voltage levels was correct.

7.5 Vertical system testing

Tests of the vertical system (table 7.2) was done by first testing each component by itself without the horizontal system mounted. This was done because all of the vertical moving components are independent from each other and could damage the vertical system if moving in opposite directions.

Function	**Result**
Brake on/off	Both works
Test motors	Both works
Test encoders	Not tested
Test optical end switches	All works

Table 7.2: Vertical system test

7.6 Horizontal system testing

Before assembling the belts to the horizontal system, a set of test were done. These tests controlled that all the end stops were working and configured correctly in the software. We also checked that the controller were able to operate the stepper motors. Table 7.3 shows the results of the test.

Function	**Result**
Test stepper motors	Both works
Test mechanical end switches	All four works
Test electromagnet	Both works
Test tray detection on magnets	Works

Table 7.3: Horizontal system test

7.7 I2C as communication protocol

The first chosen protocol between the controllers was I2C protocol. This because it fits the required bandwidth and all the controllers can communicate in this protocol as its supported by all the languages. I2C is also implemented by the libraries already in use on all the controllers. The I2C protocol is master-slave and thereof the Odroid will be configured as the master and the Arduinos as slaves. The standard bandwidth for I2C is 100 kbit/s (12500 bytes) and as our program sends at most ten bytes in one go and a polling which sends one byte with intervals of hundred millisecond and answers being normally around three bytes. This means in theory we would have lots of bandwidth to go on. If not, adjusting the bandwidth higher is possible. The I2C serial protocol was thereof chosen as our desired communication protocol between the controllers [32].

7.7.1 I2C communication via Arduino

To be able to read and send data via I2C on Arduino, the wire library is used. The address for the Arduino slave device is set, when the device recognizes this address sent on the bus it triggers an interrupt. This interrupt is either *onRecieve* or *onRequest* based on write or read command from the master. The interrupt functions holds a handle to a specified function. The function handle is set during the setup. The ability to read the incoming data is given by using the command *read()* from the wire library and the same with sending data via the *write()* command.

7.7.2 I2C communication via Odroid

Using the Pi4J library(1.2-snapshot version) made the ODROID XU4 compatible with I2C. The library is granted access to the I2C bus on the Odroid and is thereof able to send messages on the bus. I2C devices are created with addresses and connected to the specified bus. By using these devices read and write methods, the correct sequence according to the protocol is carried out. Buffering of both incoming and sending multiple bytes is possible which makes it more general and lightens the ease of use.

7.7.3 Voltage-level translator

The Arduino runs its I2C I/O pins at 5V and the Odroid on 1.8 volts [55] [57] [64], voltage shifting is therefore needed to connect the two different voltage signals. The voltage translator also has 1kΩ pull-up resistor for the clock and data line which is needed for the I2C to work, and a drain capacitor on 0.1uF to remove some of the high frequency noise from the lines [74].

7.7.4 I2C communication on robot construction

When mounting the controllers on the robot with all the electrical parts of the robot being connected and running, we ran into trouble regarding the communication between the controllers. The communication worked perfectly under ideal conditions with only the controllers, but not when installing them on the robot. The troubleshooting and steps taken to try and fix the communication is described in this section.

- *Odroid connected to Arduino Mega, with only 5V and 12V supply powered on*
  This is the bare minimum setup, as the 12V supply powers the Arduino Mega and the 5V supply powers the Odroid. The 24V supply was turned and no stepper motors were driven during this test phase.

  - With only voltage level translator connected between the controller, it worked for an random amount of time. As described in chapter 7.7.3 the translator board has pull-up resistors and a capacitor which suites in general most circuits. We tried to modified these to get a better result.
  - Disabled internal pull-up resistors in Arduino and Odroid, this made no changes.
  - Added 10kΩ pull-up resistors to the SCL and SDA line. For a total of $\frac{1}{(1/1.7)+(1/10)} = 1.45k\Omega$. This did not have any obvious effects.
  - To make sure the voltage on the communication pins were correct, and voltage drop was not the cause of these problems the voltage was measured. The measured voltage between ground and the SCL and SDA when they were not hold low by any controller was about 5V, although some voltage drop to 4.8V did occur when measuring. This voltage drop is not significant and should not affect the controller to misinterpret the signal. A voltage from miniumum 0.7*VCC on the input is regarded as high, this can be found in the datasheet for the Arduino Mega chip the Atmega2560 [75].
  - The 10kΩ pull-up resistors were changed with 4.7kΩ pull-up resistors, and the communication works indefinitely without any problems. Total resistance value from the SCL and SDA data line to each voltage ref is now $\frac{1}{(1/1.7)+(1/4.7)} = 1.28k\Omega$.

- *Odroid connected to Arduino Mega, with 5V, 12V and 24v supply and steppers running.*
  Turning the stepper motors on and supplying the drivers introduces EMC noise from the stepper drivers which is situated in the cabinet. The controllers and stepper drivers are mounted on different side of the cabinet to reduce the possible EMC noise to affect the controllers.
  - When turning on the 24V supply all communication halts.
  - Added 0.22uF caps for decoupling of voltage references to ground on each side of the voltage translator. With the decoupling caps the communication works with 24V supply turned on. When the motors were driven the communication halted after a random amount of time,

and did not last any longer then halfway through the calibration process. This is as good as we got it to work.

- *Figuring out why running the stepper motors would stop the communication*
  The first thought and most logical reason would be noise radiated from the stepper drivers. The task is then to reduce the noise and localize where it came from and why the communication halted. When installing the components the drivers were mounted on the opposite side of the cabinet as the controllers, this was done to remove the controllers from possible noise sources. The possible sources now can be that it is carried through the cables from the drivers, come from the power supply when it supplies loads or EMC noise from the stepper drivers.
  - Cables running to the motors were switched with shielded cables and the shielding terminated to ground. This would be the most obvious step from shielding the rest of the equipment from the noise, but it did not seem to help.
  - Next step was changing the power supply to an external supply. No changes here either. This concluded it was not the power supply.
  - To remove the interference from the drivers, the drivers were moved outside the cabinet. The only cables which went into the cabinet from the drivers were now the signal cable and the power cables from the supply. We still had the same problem with the communication stopping after an random amount of time.

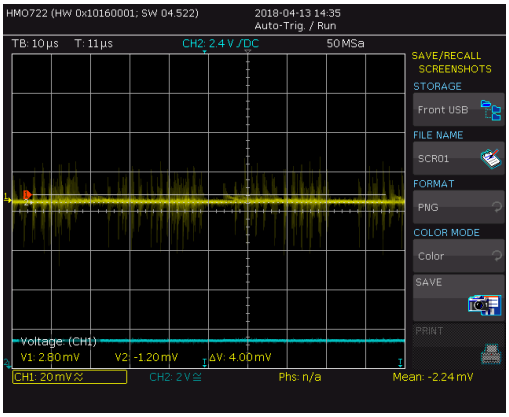


Figure 7.7: SDA line with drivers running

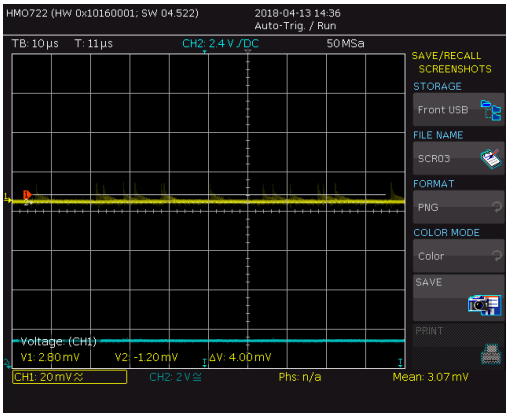


Figure 7.8: SDA line without noise

- By connecting an oscilloscope to the SDA and SCL line on the Arduino side of the voltage translator, we were able to confirm that the noise was around 75mV as can be seen in figure 7.7, which is not high enough to affect the system. As mentioned earlier in section 7.7.4, the threshold for HIGH signal is $0.7 \times VCC = 0.7 \times 5V = 3.5V$, and with 75mV noise the lowest possible voltage will be $4.8V - 0.075V = 4.725V$. Which is way above the 3.5V threshold. This is with shielded cable, the drivers mounted inside the cabinet and by using internal supply.
  The lines without driver noise is shown in figure 7.8.

- *Odroid connected to Arduino Mega and Nano*
  However the noise did not matter, because as soon as we connected the Nano to the bus, there was no communication between the Odroid and the other controllers. Even with only just the controllers powered. Connecting the Nano introduced a much longer cable run on the bus, as the cable run from the Odroid to the Nano is about three meters, this adds some capacitance to the line.
  - Because of the added capacitance and that the voltage translator adds some delay to the circuit, as mentioned in their data sheet. We tried some different setups with the speeds to see if it could solve our problem. Reducing it to 100kHz from 400kHz on all controller, also tried setting the master at 400kHz and setting the slaves to 100kHz, but to no avail.
  - After connecting the Arduino Nano we also tried reverting all the changes mentioned above and running a clean line with external supply and drivers mounted outside with shielded cables. Still no valid messages across the bus.
  - The conclusion was that the I2C communication does not work at all when the Nano is connected to it. Therefore to be able for the Arduino Nano to communicate with the other controllers some other other major steps has to be performed to the system.

- *What happens when the communication halts*
  - By measuring with an oscilloscope we can see that the SDA line is held low, and the Odroid(master) is not able to regain access to the data line until the Arduino(slave) is restarted. There can be a lot of reasons to this, there are some bugs reported on the official github of the Arduino software regarding the Arduino wire library using it with I2C, the most important one being how the Arduino freezes while using PWM output at the same time as initiating I2C communication or just freezing. Other reason may be the Arduino missing a clock pulse. However, it is not possible to verify that this was the source of our problems. What we do know is that the bus is stuck until the Arduino slave releases the bus by restarting. As can be seen in the figure 7.8.1. The yellow line is the SDA line with a delta voltage of 200mV and the blue line represents the SCL line with a delta voltage of 5V. It can clearly be seen how the master with the SCL clock tries to initiate a connection, but is not able to because of the SDA line being held low. And therefore not getting the correct start sequence for a message.

Figure 7.9: Master trying to initiate transfer with slaves

We learned by this that the I2C protocol should only be used on smaller ranges and when there is no possible noise present. As noise and added capacitance by the length of the cables can make for a lot of issues regarding this protocol including the bad error handling by the wire library. Better research should have been done on these issues when choosing the protocol the first time over. Since the problem with the I2C-bus were so severe when connecting the Nano it was decided to go for plan B and change the protocol instead.

7.7.5 Changing the communication protocol to Serial

Since reliable and stable communication between the controllers could not be obtained with the I2C protocol, the decision was to change the protocol to get the robot working. Because we are not able to test the complete robot in action without the communication working we need to get the communication working as fast as possibly. This means we need to choose the easiest, the one that seems to require the least amount of work and that most certainly will work. We chose a basic Serial setup as explained in section 4.9.

- *Re-programming java and Arduino*
  - Changing the protocol means reprogramming the I2C communication class to one that uses Serial instead. Because the software holds such high cohesion it is only necessary to change the one class which the communication protocol is defined in and coded. The Arduino controllers and its software is also developed in this regard but because of the limits of the programming language the code is more intertwined and needed more rework to change from interrupt based I2C protocol to a bi-directional serial protocol.

7.8 Design reviews

In the process of assembling the robot it was discovered a need for modification of several components. Most of the changes that had to be made were due to unforeseen interactions with other components or the restrictions of the robots area of movement. The changes were mostly implemented with the 3D-printed parts as these are the easiest manipulated and cheapest parts to change.

7.8.1 First iteration

The first design review was done after test-assembling the entire robot for the first time. At this point we discovered some flaws in the design causing interaction between components.

Y-axis motor-housing was centered on the y-axis linear-rail, this in combination with the diameter of the timing pulley made the timing belt rub against an edge on the y-axis trolley. Changed the motors placement in the motor-housing, centering the timing belt in its slot of the trolley.

Y-axis bracing was needed for keeping the y-axis beam horizontal, without bracing the beam tilted down from its own weight. The bracing was designed by using a POM-wheel pressing against the x-axis beam, forcing the y-axis beam to stay leveled. The bracing is displayed in figure 7.10.

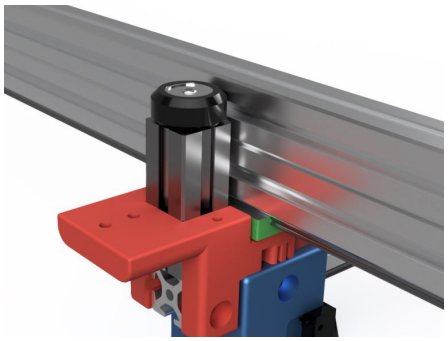


Figure 7.10: Bracing of y-axis

Y-axis cable chain was added for keeping the wires for the tool gathered in a dedicated area.

X-axis motor-bracket was modified with adding a house surrounding the timing pulley for avoiding objects from being caught between the belt and the pulley.

Fan on linear driver box has been added for cooling the stepper drivers for the x- and y-axis.

7.8.2 Second iteration

The second iteration was done while doing the second and last assembling of the robot, changes in the design was done for practical reasons such as easing the mounting and creating components which provide fewer sources of error during operation.

Vertical system end stop flags needed redesigning as their original design made them unnecessary hard to adjust for trigging the end stop sensors. If the tool of the robot were redesigned it would have been necessary to redesign the end stop flags also. With the new end stop flags being a part of the trolleys at the vertical systems, they will not need rebuilding in case of a tool change.

Vertical system end stop sensor holders were redesigned because of the same reason as the *vertical system end stop* flags were redesigned. The new holders makes it possible to adjust the height

of the end stop sensors easy as they are mounted by one bolt and are adjustable, as can be seen in figure 7.11.

Handling tray mechanism For locking the horizontal system to the tray some sort of mechanism is needed. Two main designs were made and discussed. One of them consists of using a servo to extract a tool and locking it by grabbing a handle mounted on the front side of the tray. This solution means a servo is needed, a tool mounted on the servo, and a handle to grab the tray. All of this has to be properly mounted and situated to fit perfectly. The end-effector has to be moved in position, with the offset of the handle taken into account. The other design involves using electromagnets for locking the trays to the end-effector. This requires a bit more power than using a servo, and instead of mounting a handle on the trays, a steel bracket can be used.

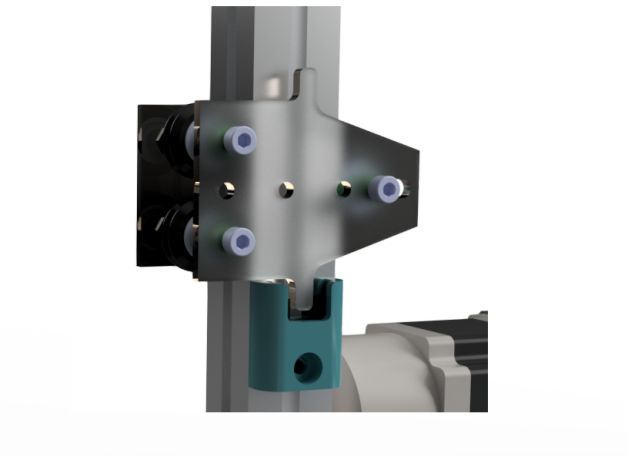


Figure 7.11: Optical sensor holder

7.8.3 Third iteration

The third and last iteration was done after using the robot for a while. At this point we could see which designs that did not perform well during continuous operation.

Y-direction pulley holder needed redesigning, as the first design (see figure 7.12) did not keep its vertical position relative to the y-axis beam over time. The tension from the y-axis timing belt made it tilt inwards, this resulted in a lower tension of the band and thus inaccurate positioning in y-direction. The pulley holder was reinforced with an extra fastening point on top of the y-axis beam, as shown in figure 7.13.

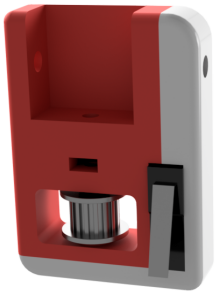


Figure 7.12: Old y-axis pulley holder

Figure 7.13: New y-axis pulley holder.

7.8.4 Vacuum generator review

We tried a few design before finding the one that functioned optimally and as desired. One of the earlier designs is shown in figure 7.14. This version did not create enough vacuum, but the other

requirements mentioned in chapter 4.6 were met. The conclusion was that the flow from the fluid port into the venturi chamber, did not achieve the speed needed to generate enough vacuum. Also the motive fluid was angled in to the chamber, thus making the fluid turbulent.

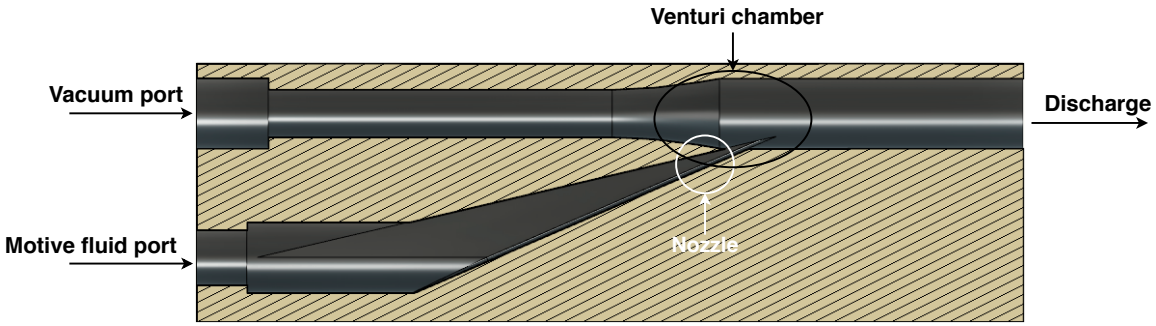


Figure 7.14: Venturi chamber design early version

7.9 Software reviews

Throughout the software progression and development of structures a variety of different options has been considered, where some of them would work fine and as expected. By reviewing our code logic and structure and seeking guidance, different and better versions with looser coupling and a more professional finish has been developed. Some solutions as using commandos to execute tasks between the controllers has always been a part of the software solution.

7.9.1 First iteration

The first iteration of the program structure focused on creating objects in the java program such as sensor objects and hold all the information related to this part. Which is critical to controlling the robot. This was mainly proposed due to thinking of the Arduinos as an I/O module where the Odroid would function as the one controller. The Odroid would then hold and update every information required for controlling the robot. This approach results in the java-program running low level logic, which means changes and integration or changes to equipment being harder and requires more work on the main java program which controls the main robot logic. Some of these tasks would be updating every sensor in real-time and decide what happens when one of the sensors are triggered. The result of this would be a lot of critical information constantly travelling over the i2c bus, as well as tasks which could be handled internally in the Arduino. Slower responses may occur and a lot of extra work for the Odroid.

7.9.2 Second iteration

After seeking some guidance and getting answers to questions regarding how to approach the programming, a more abstract approach was taken. The program structure would focus around keeping real-time sensor updates and low level logic control in the Arduino. Dividing the java program into communication and Roe robot logic, where *communication* handles the required communication to the Arduinos(controllers), and *robot logic* handles everything related to controlling the robot

including tasks as sending commands to the communication to analyzing the pictures with help of other classes and libraries.

7.9.3 Third iteration

The third iteration splits the roe robot into more classes, where one would do the roe analyzing and another handle all the tasks required by the program to operate the robot as wanted. This means sending correct commands and relaying incoming information. It implements I2C bus in the following class I2Ccommunication which handles the communication from the Odroid to the Arduino controllers via the bus. Two queues holds outgoing messages(commando) one which are expecting information back, and another which only writes to the devices. This is done because it is necessary to know when the slave is supposed to answer with information(read bit in I2C protocol set high, triggering *onRequest* function in Arduino). The I2C-Communication receives object with the super-class Commando. The objects handled by this class are always created with an under-class which extends the commando, this under-class tells the communication class how it should handle this specific commando. From every incoming message(read messages from the slave devices) an object with Status as super-class is created, as long as the register address relayed in the payload is recognized as a valid status.

7.9.4 Fourth iteration

The last and final software iteration works mainly as the third one did, but instead of implementing an I2C bus it uses Serial with one USB-port connected to each controller. It interacts with the other classes the same way as the I2C class did. As described more in depth at 6.2.

Chapter 8

Final results

This chapter elaborates on results of the various systems implemented in the robot and the robot in its entirety. Figure 8.1 is a picture of the final system with the RoeBot and the hatchery imitation.

Figure 8.1: The RoeBot

8.1 How the robot works

The robot identifies trays in the rack during calibration and continues by moving to and open the first tray by utilizing the tray opening mechanism. When the tray is open, the robot then captures a serie of images of the tray and processes them for locating dead roe. A list of coordinates for the dead roe is returned from the image processing, and is then processed by the path optimizer for finding the fastest route altogether. Then by a sequence of movements the dead roe are removed by suction. At last the robot closes the tray and moves on to the next. How the various operations explained above is developed can be read in chapter 5 and 6.

8.2 Workspace of the robot

Several factors limits the size of the robots workspace. It is important that the robot can freely move without crashing into obstacles. Some obstacles decreases the workspace, on the horizontal system these are mainly the flags mounted on the trays and the vacuum tube. For the vertical workspace, the limits are provided by the junction boxes at the base and the casing for the horizontal controller at the top. From the robot calibration in 6.3, the limitations can be found in millimeter. The workspace is 713mm for x-, 351mm for y- and 735mm in z-direction. These parameters generates the total workspace illustrated in figure 8.2 seen from the front and 8.3 is the horizontal system seen from the top.

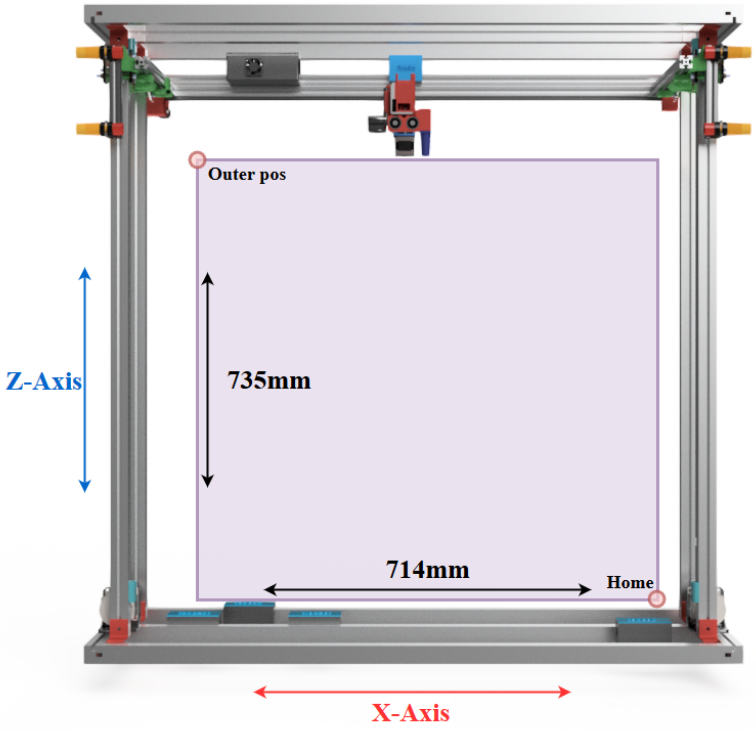


Figure 8.2: Robot workspace dimensions x and z plane

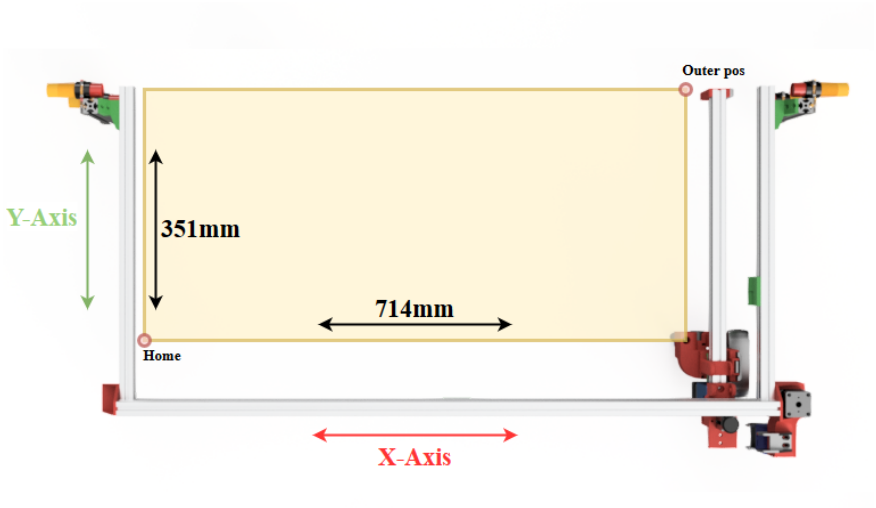


Figure 8.3: Robot workspace dimensions x and y plane

8.3 Vertical system motion

The vertical system performs movements in z-direction. The test is performed in order to verify the precision and repeatability of the motion. Testing of the movement was done by measuring the software position relative to the actual position of the system. Movement testing of the vertical system was done as following:

- To check the accuracy of the motion, after the calibration the robot was moved to position (10). A reference point was marked on this position.
- Measured and marked a new point on the z-axis 600mm above the robots current position. Then moved the robot to position (610), which means a movement of 600mm in vertical direction from position (10).
- The robot hit the marked reference point exactly, it was not possible to measure any error.
- To check the repeatability the same points were measured after performing random movements for ten times in a row, a second time after moving twenty times.
- All the measurements read perfectly, thus it was still accurate after performing random movements.

8.4 Horizontal system motion

The testing of precision and moving pattern for the horizontal system is done with a set of predefined tests. The test were performed by checking software position versus real life position. An illustration of how the horizontal system moved during the test is shown in figure 8.4. As well as random movements are shown in figure 8.5. All data is collected from the controller of the horizontal during motion.

8.4.1 Horizontal system movement

Testing the general movement of the horizontal system verifies that it is able to move freely inside the workspace. The extremity values from the calibration is controlled against the distances between the end-stop sensors. The end-effector was moved diagonally between all corners of the horizontal workspace. These movements combines all four combination of increasing and decreasing the x- and y-axis. Figure 8.4 illustrates the moving pattern of the end-effector.

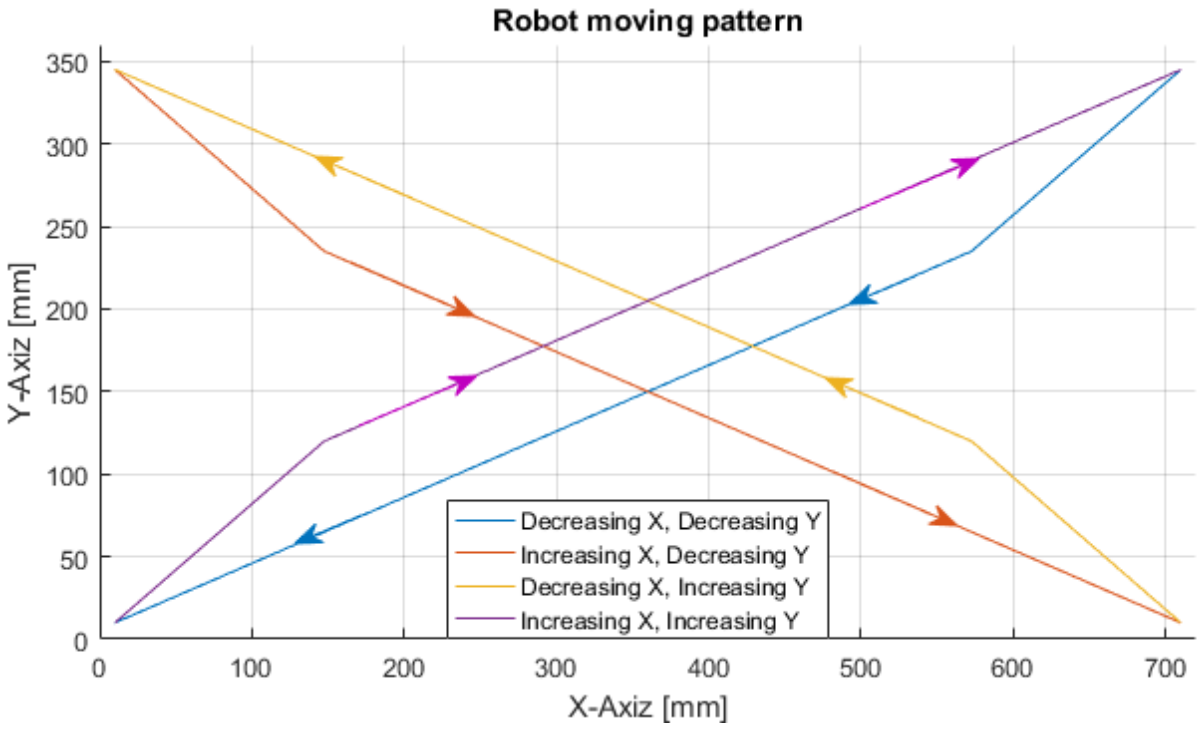


Figure 8.4: How the horizontal system moves

8.4.2 Horizontal system accuracy

An accuracy of less than 1mm should be achieved for the horizontal system to position itself for removing a roe. The first test is for the x-axis, and is done by marking a reference point at position (10,10). Then a position 600mm from the reference point is marked. Further on, the robot is moved to position (610,10), which is a movement of 600mm in x-direction. The robot hit the reference point on the x-axis exactly. The same test was done for the y-axis. Here with a movement of 300mm due to the workspace being smaller than the x-axis. As far as we were able to measure, the robot hit the mark. The software indicated an error of 0.04mm, but we were not able to verify an error margin this low.

8.4.3 Horizontal system repeatability

After the precision was verified the repeatability of the horizontal system also had to be checked. Firstly, a set of ten random movements were performed, and afterwards the reference points (10,10)

and (610,310) were measured. We were not able to measure any errors, the movements were as accurate now as when first performed. The software indicated an error of 0.02mm and 0.04mm. Secondly, a set of twenty movements were performed with the same results when the reference points were measured.

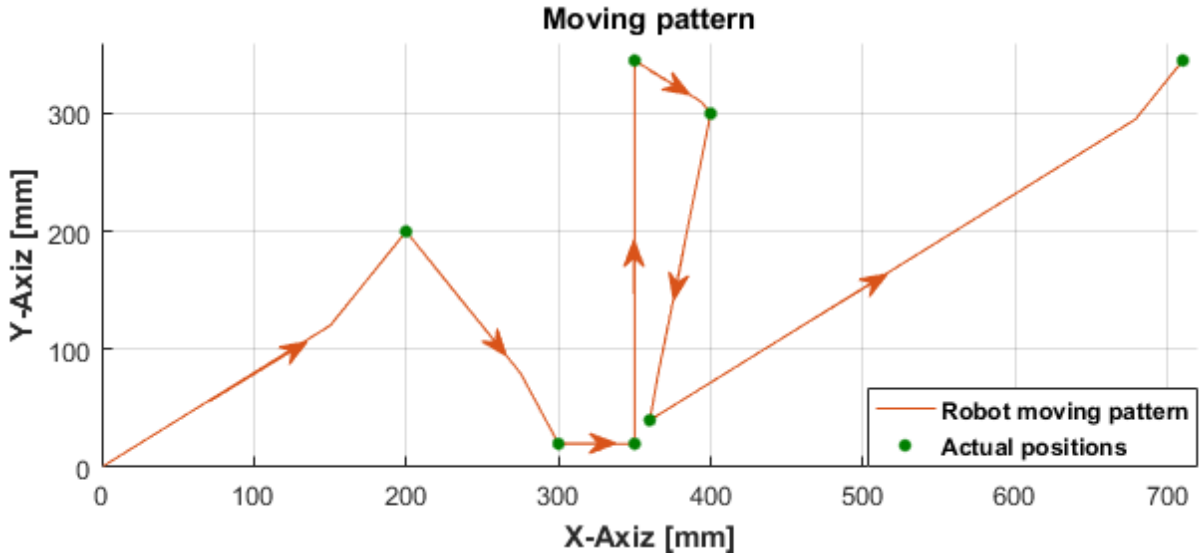


Figure 8.5: Horizontal system accuracy

8.5 Tray handling

Tray handling includes *identifying*, *opening*, and *closing* trays. For the *opening* and *closing* to be possible, a system handling the offsets for a tray needed to be in place. By using the safety beams as explained in chapter 6.2.5 the robot was able to identify trays and open the trays. Closing the tray was done by moving the end-effector in place and pushing the tray until the end stop was reached. All off the tray handling tasks were successful.

8.6 Machine vision

The image processing techniques developed were tested in order to verify their efficiency. Each technique was tested with a number of sampled images. By knowing the amount of dead roe in each image, and estimating the total number of roe, we were able to calculate a confusion matrix for the efficiency.

8.6.1 Image processing result of real roe

The verification of the image processing of roe, was done with sixteen images captured at a Marine Harvest hatchery. The pictures displays approximately the same amount of roe, with a mix of living

and dead, one of the images contains only living roe. Table 8.1 shows information of the images that were used for testing the image processing and the result of the image processing.

Image nr	**Dead roe in image**	**Dead roe found**	**Living roe miss-classified as dead**
1	6	6	0
2	1	1	0
3	2	2	0
4	2	2	0
5	2	2	1
6	4	3	0
7	2	2	1
8	4	4	0
9	4	4	1
10	3	3	0
11	2	1	0
12	3	3	0
13	1	1	0
14	0	0	0
15	2	2	0
16	1	1	3
Total	**39**	**37**	**6**

Table 8.1: Image processing result, real roe

Image from test number nine:
As can be seen from the result of the ninth processing in table 8.1, there were four dead roe in the image, and all of these were found with an additional miss-classification of a living roe. By comparing the image pre-processing (figure 8.6) with the image post-processing (figure 8.7), we saw that the object found in the lower right corner was actually glare that were miss-classified as a dead roe.

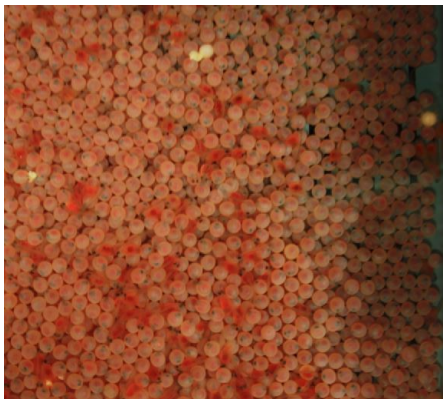


Figure 8.6: Pre processing real roe

Figure 8.7: Post processing real roe

Looking at figure 8.6, we approximated the number of roe in a image to be about 1500. By using sixteen images for testing, we got a total of circa 24000 roe in the test. Thereof we were able to calculate a confusion matrix for the processing accuracy.

Confusion matrix:

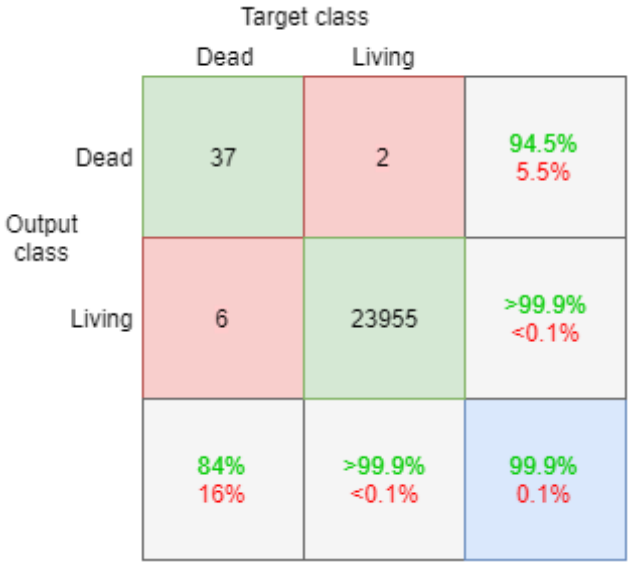


Figure 8.8: Image processing real roe confusion matrix

The confusion matrix (figure 8.8), reveals the accuracy of the image processing. As the confusion matrix shows, 94.5% of the dead roe were found, leaving a total of 5.5% of the dead roe in the tray. Another side to the result was the six living roe that were miss-classified as dead, this lead to the removal of $<0.1\%$ of the healthy roe. The total result of this image processing yield a result where 99.9% of the roe were correctly classified.

8.6.2 Image processing of imitated roe

The verification of the image processing of the roe imitation, was done with sixteen images captured by the robots camera. The images displays the imitating roe submerged in water, the number of dead roe varies.

Image nr	Dead roe in image	Dead roe found	Living roe miss-classified as dead
1	5	5	0
2	6	6	1
3	1	1	0
4	4	4	1
5	4	4	0
6	5	5	1
7	0	0	1
8	6	6	1
9	2	2	0
10	3	3	2
11	4	3	1
12	1	1	0
13	3	3	0
14	3	3	1
15	4	3	1
16	3	3	1
Total	**59**	**52**	**11**

Table 8.2: Image processing result, roe imitation

Image from test number three

Figure 8.9 and 8.10 shows the image from test number three in table 8.2 pre- and post-processing. This image was captured at one of the four locations closest to the lights mounted on the robot, hence one of the images that were most prone to light glare. The image can be seen in figure 8.9, and has one dead roe in the left side of the image. This roe is the same object located in figure 8.10.

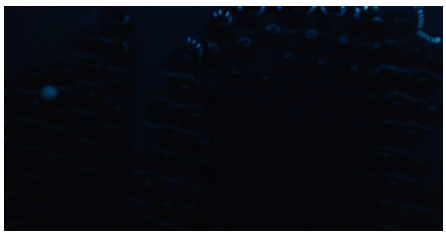


Figure 8.9: Pre processing imitated roe

Figure 8.10: Post processing imitated roe

Looking at figure 8.9, the approximate number of imitated roe in a image is about 240. By using 16 images for testing, we got a total of approximately 3800 roe in the test. From there the confusion matrix is calculate for the efficiency.

Confusion matrix

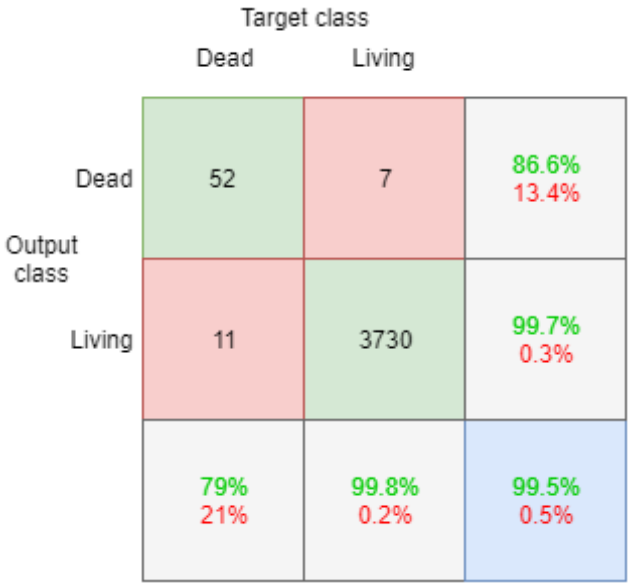


Figure 8.11: Image processing imitation roe confusion matrix

Figure 8.11 shows the confusion matrix calculated from the result of the image processing. It reveals that out of the 59 dead roe, 86.6% was found, leaving 13.5% in the tray after removal. Out of the approximated 3741 living roe, 11 (0.3%) was miss-classified as dead. The test in its total reveals the image processing to be 99.5% correct.

8.7 Pattern optimizer

In this chapter we will test the two different types of pattern optimization heuristics. Two data sets are made to evaluate result. Both sets are made within the range of $200mm \times 650mm$ and all destination are randomly spread. The order of the destinations are ranged in the same way as if they had been found by the image processing (chapter 6.7.1). The first set holds 25 destinations implanted mostly in the center of the camera series (chapter 6.6.2). The second contains 47 destinations. We have tested the nearest neighbor algorithm and genetic algorithm against the original path the dead roe was detected. The best solution will provide the lowest cost. The cost, is the time used for horizontal movement, plus the time used to improve the solution.

8.7.1 Result with 25 destinations data set

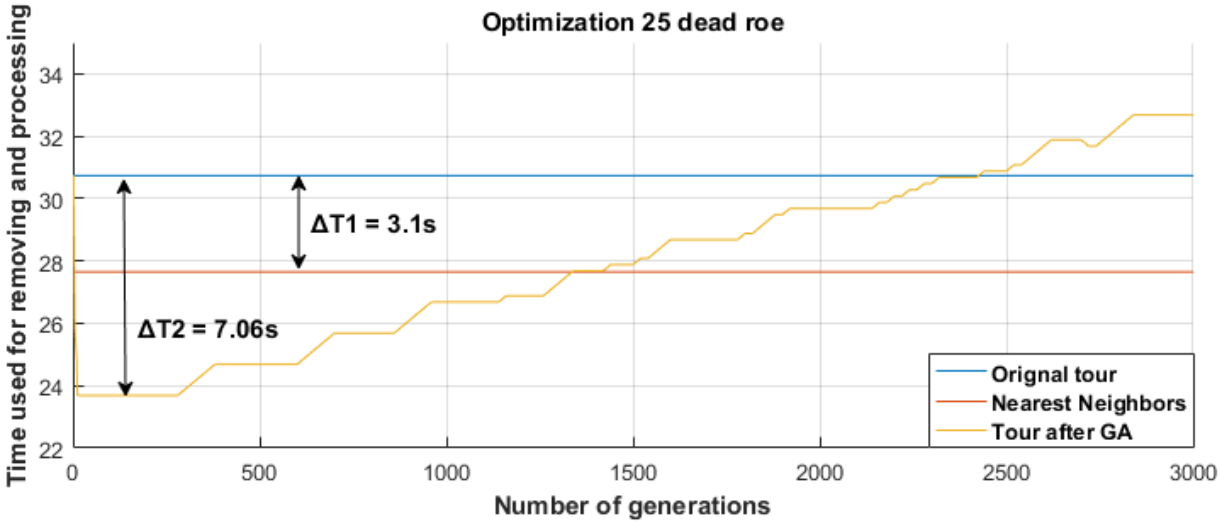


Figure 8.12: Result of GA

We started by testing the pattern optimizer with the 25 destination sets. An estimate of the cost for the three methods are illustrated in figure 8.12. This problem represents 25 dead roe that needs to be removed. As seen from the graphs, the original time is 30.74s. By using a nearest neighbor optimization algorithm (chapter 2.10.2), processing time was two milliseconds, and cost decreased to 27.65 seconds. Compared to the original tour, this was an improvement of 10%, and reduces the cost by $\Delta T1 = 3.1s$. We also tried to save cost by applying a genetic algorithm 2.10.3. The algorithm has been tested for 0 to 3000 generations with a populations size of 300 chromosomes. We see after the first 10 generations, the cost already have dropped to 23.7 seconds. Reducing the cost with 22.9% saves $\Delta T2 = 7.06s$. This result are maintained for approximately 300 generations. After these 300 generations, the cost expands. When the GA has run over 1400 generations, the cost is the same as for the nearest neighbor method. Above 2400 generations the cost is higher than the original cost. Paths generated from the original tour, nearest neighbor and our genetic algorithm is shown in figure 8.13, 8.14 and 8.15. Start destination are colored in green.

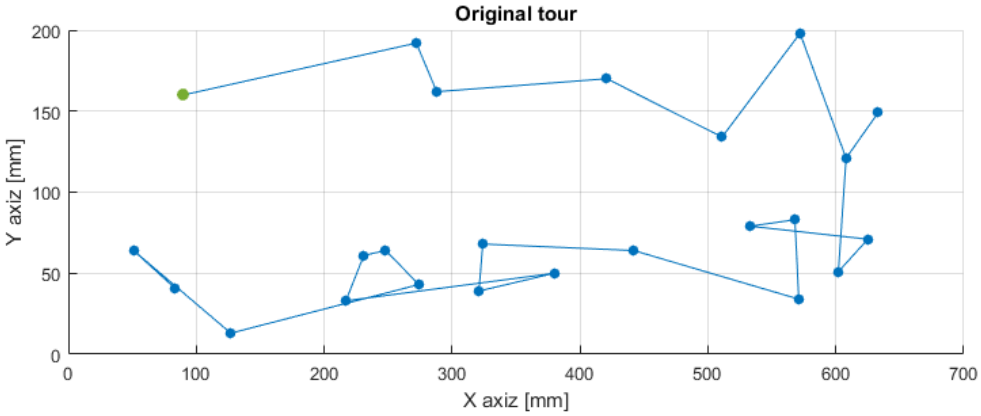


Figure 8.13: Original tour size 25

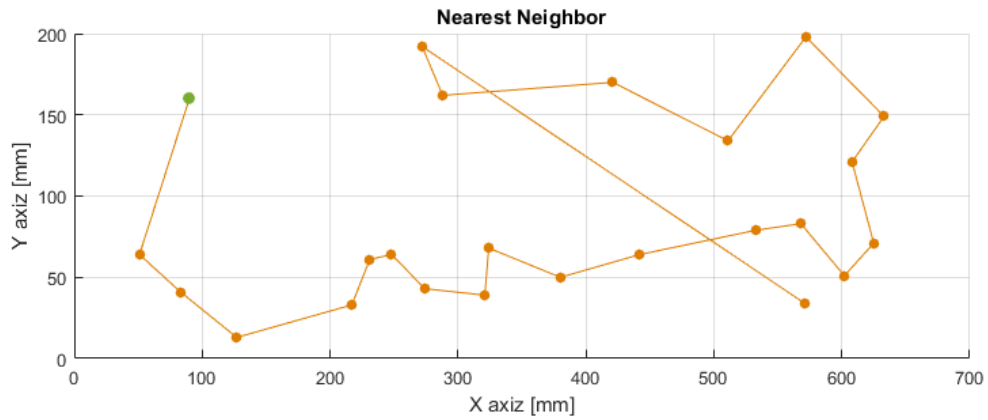


Figure 8.14: Tour optimized with nearest neighbor. Size 25

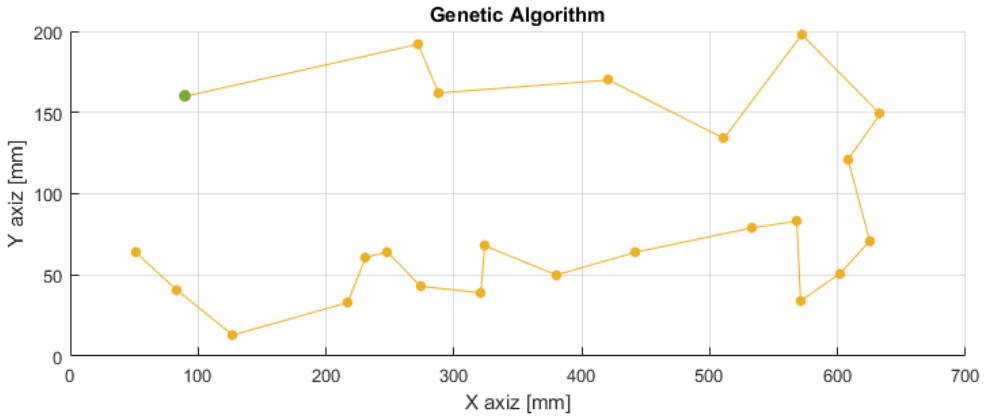


Figure 8.15: Tour optimized with GA. Size 25

8.7.2 Result with 47 destinations data set

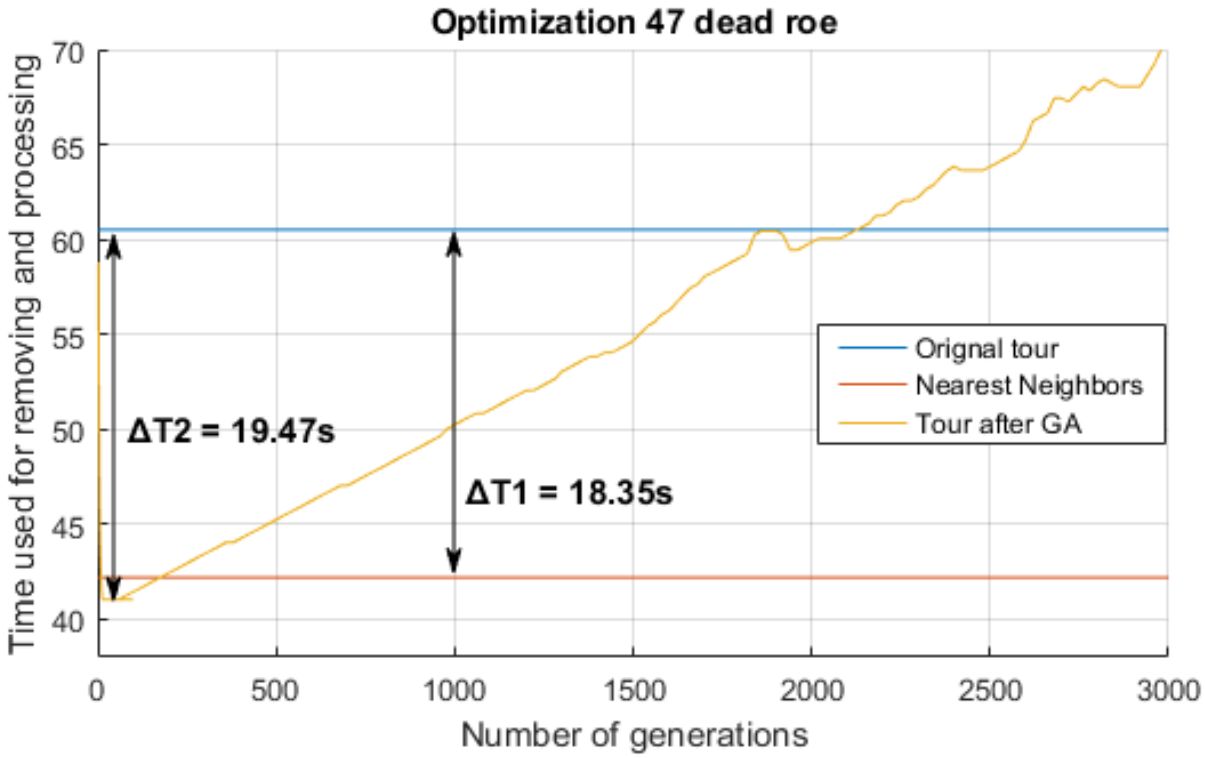


Figure 8.16: Result of GA

The second test was done by using the 47 destinations data set. In graph 8.16 the cost is represented along the y-axis and number of generations of the GA along the x-axis. In graph 8.16 we have improved the original tour by using both optimization heuristics. The blue line is the original tour with cost 60.51 sec. After the nearest neighbor algorithm has been used, the cost saving is $\Delta T1 = 18.35sec$, this is a reduction of 30.3%. Using the GA for improvement, the best result occurs in the range from 10 to 100 generations. Compared to the original tour, this saves us $\Delta T2 = 19.47s$ which is 32.2%. If we run our GA for over 210 generations, it has the same improvement as the nearest neighbor and steadily closes in on the original tour. When passing 2200 generations, the cost is higher than with the original tour. The paths for each method is illustrated in figure 8.17, 8.18 and 8.19. The start destinations are colored in green.

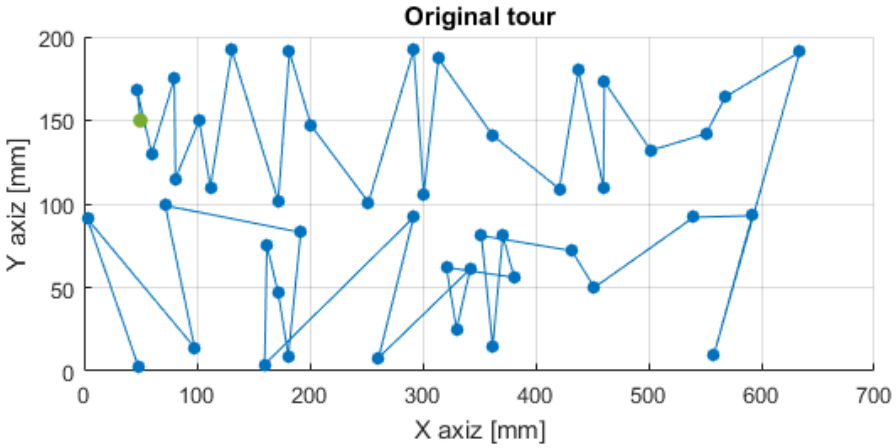


Figure 8.17: Original tour. Size 25

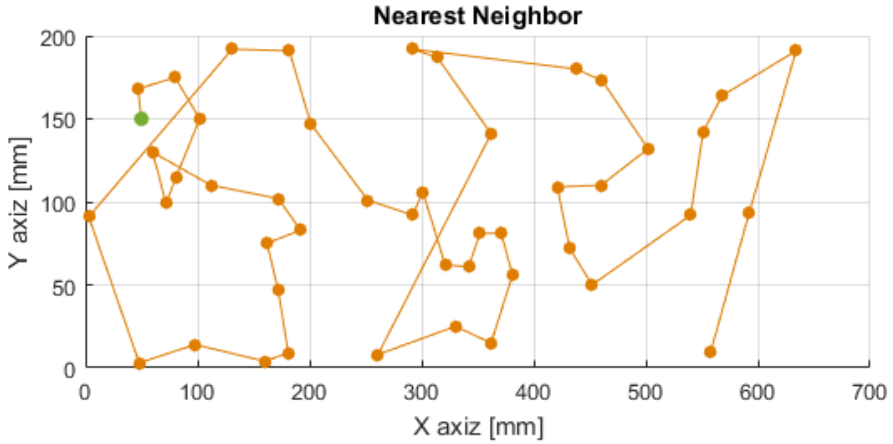


Figure 8.18: Tour optimized with nearest neighbor. Size 25

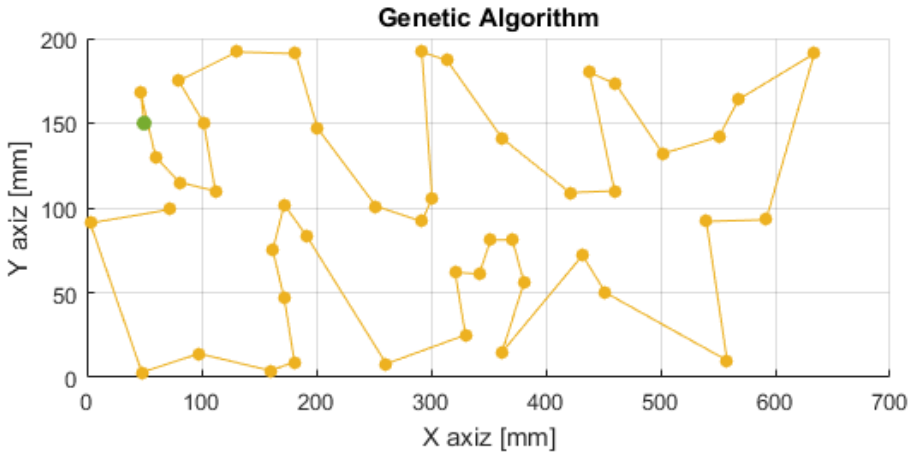


Figure 8.19: Tour optimized with GA. Size 25

8.8 Removing roe

One of the main tasks of the robot is to remove the dead roe. This is done by using the suction as explained in 6.10 and machine vision in 6.6. Since the test medium has a different composition than real life roe, the suction and support functions around is more important for a proof of concept than actually removing the roe. The robot removes the test medium with varying results. Figure 8.20 shows a roe being removed from a tray during testing.

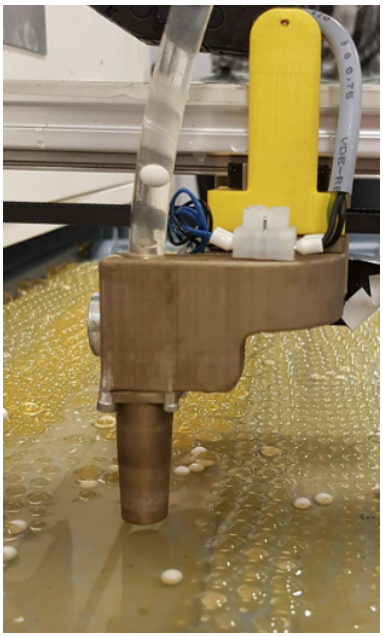


Figure 8.20: Removing roe

Chapter 9

Discussion

In this chapter we discuss the results obtained in the previous chapter and the reliability of the data collected. Also our experiences during the project and hypothetical profitability with the RoeBot is presented.

9.1 Results from testing

Here we discusses the results obtained from the previous chapter.

9.1.1 Motion

By looking at the specifications mentioned in chapter 3.2.1 one of the main points is accuracy for moving to one specific position. After doing the tests for horizontal motion in chapter 8.4 and vertical motion in chapter 8.3, the accuracy and repeatability could be pinpointed and verified as good enough to move within 0.04 millimeters. As a result, the movements are precise and consistent within our specifications. For smoother movements, deceleration and acceleration could be added. Because a simplified and modified version of the Bresenhams Algorithm is used, the movements are not completely straight. This could be perfected by implementing the proper version of the algorithm. However, this does not create any problems for our prototype, as it moves consistently, accurately and smoothly.

9.1.2 Rack system and tray handling

The robot can detect, close and open the trays in a rack system (chapter 8.5). But with the opening of trays, there are improvements that can be done. With the current design, the robot does not reach the metal plate on the trays when they are in closed position. One solution to this is to cut the side-arms on the horizontal system shorter and placing the IR-sensors closer to the home position. This will make the magnets able to reach the metal plate on the tray. In this configuration the calibration of the horizontal system can only be done while the vertical system is at the top or bottom position. Otherwise there would be a risk for the end-effector to hit the metal plates during calibration. At the same time, one would need to look at how this would affect the workspace of the robot.

9.1.3 Image processing

The result of the image processing of roe in chapter 8.6.1 reveals that we located 94.5% of the dead roe for removal. At the same time, <0.1% of the healthy roe would be removed. The total result for the image processing resulted in 99.9% accuracy. Due to the low percentage of healthy roe being removed, and the low number of dead roe that is relative to the total number of roe in a batch. We are not interested in the total result in the lower right corner of the confusion matrix (figure 8.8). The interesting result is the percentage of dead roe not located. This should be as low as possible, preferably zero. With the result being 5.5% of the dead roe not being located, we think this number is too high, thus the image processing should be improved. Another aspect of this image processing, is the images used in the test does not represent the images captured by a commercial machine. A commercial machine must be able to perform the roe removal operation with less illumination than the images used for developing and testing this image processing.

The image processing for the roe imitation yielded a worse result than the processing of real roe (chapter 8.6.2). This revealed a decreased number of located dead roe. The result of the test showed that 86.6% of the dead roe were located. This leaved 13.4% of the dead roe undiscovered, which was not a good result at all. On the other hand, the image processing made it possible to test the robot, as the robot autonomously tried to remove the roe in the positions returned from the image processing. The development of the image processing for the imitation-roe was undoubtedly harder to configure than the processing of real roe. This was caused by the low illumination in the images and the glare in the areas closest to the light sources.

9.1.4 Pattern optimizer

The result of the two pattern optimizations shows a cost reduction compared to the original tour. Depending on the size and the complexity of the data set, the difference between these two algorithms varies. The test in chapter 8.7.1 uses a data set with 25 dead roe, and the genetic algorithm resulted in a 12.9% better solution than the nearest neighbor algorithm. In the test from chapter 8.7.2, a larger and more complex data set gives a more equal cost for both algorithms. It is hard to tell which one will provide the best solution without performing more tests on real data sets. If we compare our algorithms to the original tour, we can see the benefit by implementing such algorithms. By using algorithms, we reduce the removing time with 10 to 32%, which will provide less exposure for the roe.

9.2 Software

Regarding the software, the GUI has not been prioritized. This means it has bugs and lack some features. After the biggest bugs were fixed, the program functioned as it was supposed to. All the big parts of the program was developed separately and then put together as a whole, with the *RoeAnalyser* state machine as its main logic. This made it possible to develop the image processing, path optimization, GUI, *RoeAnalyserDevice* state machine and the communication class individually.

However, the serial communication between the Java program and the Arduinos seem to function a bit clunky. The messages are always conveyed, but it sometimes looks a bit slow. The Writer and Reader uses the same Thread, as well as the same port. When the Java program reads a incoming message, the *Thread.sleep* function is invoked for the entire message to arrive before the parsing starts. Because a *BufferedReader* is used to buffer the message and then parse it, the serial port has to be set in *READ_SEMI_BLOCKING* mode to not time out while the buffer reads and parses the message. All this seem to add a small amount of delay when the Java gets a *readyToRecieve* status from the Arduinos before it sends a command and this command is performed by the arduino. This can be fixed by making a parser which do not use the *InputStream* on the serial port to buffer and parse the message. Thereof, the *READ_SEMI_BLOCKING* mode can also be removed.

On the Arduino side, the encoders on the vertical system is not implemented. This was mainly caused by lack of time. Fortunately, there has not been experienced a instance where the stepper motors has skipped any steps. Other than some bug fixes regarding failure statuses, the Arduinos and the java program does what they are supposed to do and as far as we have experienced, does so reliably.

9.3 Physical structure and design

In the beginning of the project, an extensive amount of time was used for designing the robot in 3D and finding suited components. This paid off when we were assembling the robot. Most of the parts and components were installed without any problems and no major changes were needed. The finished prototype is sturdy with both the horizontal and vertical systems being stable during operation. With none of the group members having background in design and modeling, we are very pleased with the design result.

9.4 Improvements to further the prototype

The following points are suggestions for further development of the RoeBot.

- Implement distance measurement from the end-effector downwards to the trays. With this, suction can be applied when the TCP hits the roe. It also can be used as detector for obstacles - If the RoeBot is not removing roe, the TCP should not be in contact with anything.
- Adding more sensors on the axes to verify where the robot is moving. This would add feedback if something in the movement slips and the end-effectors position is no longer accurate.
- Timing belts on the horizontal system should have mechanical tightening.
- Add a bigger and adjustable water pump with pressure measurement to regulate the suction. This will make it more adaptive to various applications.
- Test other light sources for the machine vision.

- Verify that all the dead roe is removed by doing more iterations of pictures and image processing. As mentioned in the introduction, the dead roe will rot and affect the surrounding roe. Thereof, a verification of the dead roe removal should be implemented. This can be done by taking pictures after the suction is done.

9.5 Separations from the prototype to a functional solution

The prototype is not suited for the potential work environment as it is designed today. Consequently some changes needs to be made before it can be sold as a commercial product. First of all, the components in a commercial system should be suitable to withstand environmental challenges in a hatchery. Therefore, all motors needs to be waterproof and aluminum parts have to withstand salt water. Secondly, it is supposed to work along with food production, so the machine needs components approved by the The Norwegian Food Safety Authority, as these parts will not pollute the roe by debris. Also the RoeBot has to sustain guidelines from the Machinery Directive. Finally, the RoeBot has to be designed to fit along a real rack system which is bigger than our test rack. In addition the RoeBot has to be able to operate on more than one rack, meaning it has to be able to move from one rack to another, or be built into the rack as a complete system.

9.6 Experiences

9.6.1 Planning

During the start of the project, a lot of time was used for planning the development of the prototype. This was done in form of a pre-project (appendix A.1). Especially the construction design was extensively 3D-modelled before the building and implementation started. This payed off during assembling of the construction, as the needed changes was minor.

9.6.2 Division of labor

All group members have different job and field experience. The different backgrounds have been used in the work distribution to take advantage of each members expertise. But during the start of the project, a lot of 3D-modelling had to be done. None of the group members had professional experience regarding designing or modelling, which led to self-learning. In a multidisciplinary project, as this turned out to be, with planning, constructing, and building the robot, a person with more designing experience than us would have saved us some time.

9.7 Data collection

We have interviewed a handful of people and by that we used a qualitative researching methods, when quantitative methods probably would have lead to a more solid and reliable result. The fish farm we visited has a small scale hatchery compared to other. Yet, the problems they face with the roe handling and removal should be the same as both larger and smaller facilities along the coastline, as the process is the same. In addition to interviewing the people working at fish farms,

we have obtained information from a company engaged in aquacultural research. Even though the research could have been done more extensive, the results have helped shape our project to a more realistic approach.

9.8 Hypothetical profitability

Full time employees in the fish industry had in 2014 an average yearly salary of 504.000NOK [76]. At large hatcheries the roe picking job is a time consuming job and usually requires a part time employed [77]. We assume that half a man-year is used for the roe picking job.

In our prototype, we have used some of the cheapest products available at the market, which in an industrial context often is related to quality. If the system is going to be sold as a commercial product, the points 9.4 and 9.5 has been taken into account, and then the price would probably end up at approximately 500.000 to 1.000.000NOK.

However, we expect that this robot will decrease the percentage of roe dying as a direct consequence from pollution. The potential number of surviving roe that can grow into adult salmon are around 355.000. Of the 52 hatcheries that was considered in the mortality statistics seen in figure 2.1.3, the possible yearly profitability per hatchery is estimated to be around 2.500.000NOK, considered that no roe dies [78].

Chapter 10

Conclusion

The project aims to develop a robot prototype as proof of concept for streamlining the removal of dead roe in salmon hatcheries. By introducing automated solutions to the aquaculture industry, cost of human labour can be reduced. Requirements for the prototype was to hold the accuracy and repeatability needed for gentle handling of roe. Dead roe must be located and removed in a way that reduces the exposure time of the product.

Our prototype has accomplished the required accuracy and repeatability. Utilizing machine vision and artificial intelligence, the robot manages to locate a significant amount of dead roe, and a time efficient route for removal. The prototype that is designed and built during this project, is not intended to be the final product. Yet, the solutions and ideas developed for the prototype has proven to be effective and can be implemented to build a commercial available roe picking robot. Looking at the profitability it is reasonable to believe that the RoeBot has a potential in the aquaculture industry.

Even though it has a lot of possible improvements, and everything does not work as flawlessly as intended, the group considers the project as a success. We set out to build a robot from scratch, and prove that the tasks we set to implement for this robot were possible but also beneficial for the aquaculture industry. As none of the group members have ever taken part in a project at this size, it gave us a lot of new and valuable experience regarding group and project management, but also experiences with programming complex systems, 3D-modelling and constructing. The project seemed at times quite overwhelming, but with teamwork and planning we managed to pull through with a result we deem as successful.

Bibliography

[1] *Most traded food*. Food processing. 2014. URL: https://www.foodprocessing-technology.com/features/featurethe-10-most-traded-food-and-beverage-commodities-4181217/ (visited on 05/07/2018).

[2] *Fish market*. EU. 2016. URL: https://ec.europa.eu/fisheries/eu-fish-market-2016-edition_en (visited on 02/16/2018).

[3] *Delta V*. Teknisk Ukeblad. 2017. URL: https://www.tu.no/storylabs/annonse-best-practice-har-gitt-25-prosent-mer-effektiv-produksjon/397092 (visited on 02/16/2018).

[4] Ole Andreas Fatnes. *Rognrapporten*. SalmoBreed & StofnFiskur. A Benchmark Company, 2017. ISBN: -.

[5] *Salmon Life Cycle poster*. ladybugsteacherfiles. 2018. URL: http://www.ladybugsteacherfiles.com/2012/03/salmon-life-cycle-poster-blank.html (visited on 05/23/2018).

[6] *Fisken&havet*. Havforskningsinstuttet. 1973. URL: https://brage.bibsys.no/xmlui/bitstream/handle/11250/114144/fhb_1973_11.pdf?sequence=1 (visited on 04/05/2018).

[7] Sverre Lysen. *Vasking av rogn*. 2014. URL: http://www.lagens-framtid.no/post/vasking-av-rogn (visited on 05/24/2018).

[8] *Saumon*. Marine Harvest AS. 2018. URL: http://marineharvest-france.com/reponses-a-vos-questions/saumon-conventionnel (visited on 05/23/2018).

[9] *Stor rapport om havbruksnæringen*. kyst.no. Feb. 22, 2018. URL: https://www.kyst.no/article/stor-rapport-om-havbruksnaeringen/ (visited on 05/22/2018).

[10] Jan Sunde. Private Communication - *Talked about roe that dies of pollution from other dead roe.* Ålesund, Norway, 2018.

[11] *Glossary of Robotics Terms*. YASKAWA. URL: https://www.motoman.com/glossary (visited on 05/14/2018).

[12] *ISO 8373 Robots and robotic devices*. the International author for Standardization. 2012. URL: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en (visited on 05/02/2018).

[13] *circuit digest*. circuitdigest. URL: https://circuitdigest.com/article/what-are-industrial-manipulators (visited on 03/26/2018).

[14] *Packaging automation trends*. Denso International Australia. Oct. 1, 2009. URL: https://www.processonline.com.au/content/materials-handling-logistics/article/packaging-automation-trends-using-small-assembly-robots-in-upstream-packaging-processes-555381507 (visited on 05/11/2018).

[15] Xin-Jun Liu - Jinsong Wang - Günter Pritschowa. *A new family of spatial 3-DoF fully-parallel manipulators with high rotational capability*. Science Direct. Apr. 2005. URL: https://www.sciencedirect.com/science/article/pii/S0094114X04001697 (visited on 05/14/2018).

[16] Pandilov & Dukovski. "COMPARISON OF THE CHARACTERISTICS BETWEEN SERIAL AND PARALLEL ROBOTS". In: (2014). URL: https://pdfs.semanticscholar.org/8095/cd9443fdee303a7ed607f253550c1c3a314b.pdf (visited on 02/14/2018).

[17] Chen Zhou. "Robot Motion Analysis - Kinematics". In: (Nov. 1999). URL: https://www2.isye.gatech.edu/~czhou/MOTION.pdf (visited on 02/09/2018).

[18] Tony Kliber. *Picking the Right Linear Positioning Device*. Design Engineering. Aug. 2, 2016. URL: https://www.design-engineering.com/features/linear-positioning-device/ (visited on 05/14/2018).

[19] *Belt drive*. Elega. URL: https://www.elega.lt/en/linear-motion-system-timing-belt-module-80x80 (visited on 03/12/2018).

[20] *Ball screw*. TBI MOTION Technology Co. URL: http://www.tbimotion.com.tw/product/category/14/1/en (visited on 05/14/2018).

[21] *Engrenagens Cremalheiras*. Macerol. URL: http://macerol.com.br/site/produto/engrenagens-cremalheiras/ (visited on 05/14/2018).

[22] *Ball Screw versus Belt Driven Actuators*. Myostat Mottion Control Inc. URL: http://www.myostat.ca/ball-screw-versus-belt-driven-actuators (visited on 05/19/2018).

[23] *Pitch Diameter*. Pfeifer Industries. URL: http://www.pfeiferindustries.com/glossary.html (visited on 05/19/2018).

[24] Mike Anselmo. *Go long: The pros and cons of rack-and-pinion systems*. Machine Design. Feb. 1, 2010. URL: http://www.machinedesign.com/linear-motion/go-long-pros-and-cons-rack-and-pinion-systems (visited on 05/14/2018).

[25] Gordon R Slemon. *Electric Machines And Drives*. Design Engineering. Oct. 9, 2015. URL: http://ivanbrazov.com/372497-electric-machines-and-drives.pdf (visited on 05/14/2018).

[26] *Stepper and Servo motors*. NEMA. URL: http://www.nema.org/Products/Pages/Servo-and-Stepper-Motors.aspx (visited on 05/15/2018).

[27] *StepperCalculations*. Norwegian Creations. URL: https://www.norwegiancreations.com/2015/07/tutorial-calibrating-stepper-motor-machines-with-belts-and-pulleys/ (visited on 05/15/2018).

[28] *Stepper&Servo*. Machine Tool. URL: http://www.machinetoolhelp.com/Automation/systemdesign/stepper_dcservo.html (visited on 05/17/2018).

[29] *Motor driver*. Future Electronics. 2016. URL: http://www.futureelectronics.com/en/drivers/motor-driver.aspx (visited on 04/05/2018).

[30] Hank van Ormer. *Utilizing Venturi Vacuum Generators Efficiently*. URL: https://blowervacuumbest.com/system-assessments/vacuum-generation/utilizing-venturi-vacuum-generators-efficiently (visited on 05/19/2018).

[31] *Serial Communication*. SparkFun Electronics. URL: https://learn.sparkfun.com/tutorials/serial-communication (visited on 02/27/2018).

[32] *I2C Serial communication*. SparkFun Electronics. 2018. URL: https://learn.sparkfun.com/tutorials/i2c (visited on 02/01/2018).

[33] *What is Machine Vision*. Cognex. URL: https://www.cognex.com/what-is/machine-vision/what-is-machine-vision (visited on 05/02/2018).

[34] *Head Tracking With WebRTC*. Opera Software ASA. July 18, 2012. URL: https://dev.opera.com/articles/head-tracking-with-webrtc/ (visited on 05/23/2018).

[35] *A Practical Guide to Machine Vision Lighting*. National Instruments Corporation. Jan. 30, 2017. URL: http://www.ni.com/white-paper/6901/en/ (visited on 05/14/2018).

[36] *How to avoid interference*. Vision Doctor. URL: http://www.vision-doctor.com/en/avoid-interference.html#Dirt (visited on 05/18/2018).

[37] Scott E Umbaugh. *Digital image processing and analysis, second edition*. International series of monographs on physics. CRC Press, 2010. ISBN: 9781429802052.

[38] Christine L. Valenzuela and Antonia J. Jones. "Estimating the Held-Karp bound for the geometric TSP". In: (2018). URL: https://ac.els-cdn.com/S0377221796002147/1-s2.0-S0377221796002147-main.pdf?_tid=f1c1e%2048b-884f-45b4-a0b0-313975d4fb81%5C&acdnat=1526223990_f7eede07293452e03ca51e5ca27 (visited on 05/13/2018).

[39] Christian Nilsson. "Heuristics for the Traveling Salesman Problem". In: (2018). URL: https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf (visited on 05/13/2018).

[40] THOMAS R. WILLEMAIN. "Random Tours in the Traveling Salesman Problem: Analysis and Application". In: (July 1999). URL: https://link.springer.com/article/10.1023/A:1011263204536 (visited on 05/13/2018).

[41] Michael Negnevitsky. *Artificial intelligence, a guide to intelligent systems*. Third. Addison Wesley, 2011. ISBN: 978-1-4082-2574-5.

[42] Jinghui Zhong Xiaomin Hu Min Gu and Jun Zhang. "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms". In: (2005). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.3747&rep=rep1&type=pdf (visited on 05/14/2018).

[43] Kenneth I. Joy. *BRESHENHAMS ALGORITHM*. Computer Science Department, University of California, Davis. 1999. URL: http://graphics.idav.ucdavis.edu/education/GraphicsNotes/Bresenhams-Algorithm.pdf (visited on 05/14/2018).

[44] *HDT5M timing belt*. Stock Drive Products/ Sterling Instruments. URL: http://www.sdp-si.com/D265/PDF/D265P1027.pdf (visited on 05/23/2018).

[45] *GT2 timing belt*. Aliexpress, POWGE. URL: https://www.aliexpress.com/item/2Meters-5GT-Timing-belt-5GT-25-Wide-25mm-Pitch-5mm-5GT-Belt-pulley-Neoprenen-Rubber-Small/32607033497.html (visited on 05/23/2018).

[46] *Bearing Friction*. American Roller Bearing Company. URL: https://www.amroll.com/friction-frequency-factors.html (visited on 05/01/2018).

[47] *HYBRID STEPPER MOTOR 14HY0007*. RobotDigg. URL: http://www.robotdigg.com/upload/pdf/14HY0007-20BA.pdf (visited on 05/23/2018).

[48] *Nema 23 stepper motor with brake*. Stepperonline. URL: https://www.omc-stepperonline.com/stepper-motor-brake/nema-23-stepper-126nm178ozin-w-brake-friction-torque-20nm283ozin-23hs22-2804d-b200.html (visited on 03/28/2018).

[49] *Nema 23 motor wit brake*. Robotdigg. URL: https://www.robotdigg.com/product/558/Brake-stepper-motor-Nema17-or-Nema23 (visited on 05/23/2018).

[50] *Nema23 Stepper Motor with 24V Brake*. RobotDigg. URL: https://www.robotdigg.com/product/558/Brake-stepper-motor-Nema17-or-Nema23 (visited on 05/23/2018).

[51] *Driver shield for Arduino Nano*. Keyestudio. URL: http://www.keyestudio.com/shop/3d-printing/keyestudio-cnc-shield-v4-0-board-compatible-with-arduino-nano-free-shipping.html (visited on 03/28/2018).

[52] *Switch SS-5GL*. Robotdigg. URL: https://www.robotdigg.com/product/141/Endstop-Subminiature-Switch-SS-5GL (visited on 05/14/2018).

[53] *IR-Beam Sensor*. Ebay. URL: https://www.ebay.com/itm/2pcs-5M-IR-Photoelectric-Sensor-Switch-Photoswitch-Through-Beam-PNP-DC-12V-24V/142399982113?hash=item2127b34a21:g:960AAOSwLF1X4LMC (visited on 05/14/2018).

[54] *Photo Interrupter*. Sparkfun. URL: https://www.sparkfun.com/products/9299 (visited on 05/14/2018).

[55] *Arduino Mega 2560*. Arduino. 2018. URL: https://store.arduino.cc/arduino-mega-2560-rev3 (visited on 02/01/2018).

[56] *Arduino Uno*. Arduino. 2018. URL: https://store.arduino.cc/arduino-uno-rev3 (visited on 02/05/2018).

[57] *Arduino Nano*. Arduino. 2018. URL: https://store.arduino.cc/arduino-nano (visited on 02/01/2018).

[58] *Arduino Micro*. Arduino. 2018. URL: https://store.arduino.cc/arduino-micro (visited on 02/05/2018).

[59] *FTDI Cable 5V*. SparkFun Electronics. URL: https://www.sparkfun.com/products/9718 (visited on 05/10/2018).

[60] *Logitech HD Pro C920*. Komplett AS. URL: https://www.komplett.no/product/856419/datautstyr/periferiutstyr/streaming/kamera/logitech-hd-pro-c920-webkamera# (visited on 05/14/2018).

[61] *Nema 14 stepper motor*. Robotdigg. URL: http://www.robotdigg.com/upload/pdf/14HY0006-20A.pdf (visited on 05/01/2018).

[62] *Arduino Nano*. Arduino. URL: https://store.arduino.cc/arduino-nano (visited on 05/01/2018).

[63] Tarun Agarwal. *Optical Sensors and Applications*. ElProcus. 2018. URL: https://www.elprocus.com/optical-sensors-types-basics-and-applications/ (visited on 02/01/2018).

[64] *Odroid-XU4*. Hardkernel co., Ltd. URL: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825%5C&tab_idx=1 (visited on 02/26/2018).

[65] *OpenCV*. OpenCV Team. URL: https://opencv.org/about.html (visited on 05/28/2018).

[66] *The Pi4J Project*. Pi4J. URL: http://pi4j.com/ (visited on 05/28/2018).

[67] *OpenCV*. DipLib. URL: http://www.diplib.org/dipimage (visited on 05/28/2018).

[68] *jSerialComm*. fazecast. URL: http://fazecast.github.io/jSerialComm/ (visited on 05/28/2018).

[69] *Stepper Library*. Arduino. URL: https://www.arduino.cc/en/Reference/Stepper (visited on 05/28/2018).

[70] *FastLED Animation Library*. FastLED. URL: http://fastled.io/ (visited on 05/28/2018).

[71] *HMI*. nicontrols. URL: https://nicontrols.com/uk/hmirepairs (visited on 03/26/2018).

[72] *ComputerProcessing*. okclipart. URL: http://www.okclipart.com/author/admin/page/1274/ (visited on 03/26/2018).

[73] Holger Hartmann. *Quasar200*. URL: https://www.holgerhartmann.no/strekkproving/quasar-200-article527-890.html (visited on 05/07/2018).

[74] *SparkFun level Translator*. SparkFun Electronics. URL: https://www.sparkfun.com/products/11955 (visited on 02/27/2018).

[75] *Atmega2560 datasheet*. Microchip. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf (visited on 05/02/2018).

[76] *Lønn i oppdrettsnæringen*. SSB. 2015. URL: https://www.ssb.no/arbeid-og-lonn/statistikker/lonnfisko/aar/2015-02-13 (visited on 05/16/2018).

[77] Roe Picker at a hatchery. Private Communication - *Talked about roe the roe picking job.* Ålesund, Norway, 2018.

[78] *Laksepriser*. SSB. 2018. URL: https://www.ssb.no/laks (visited on 05/23/2018).

Appendix A

Project planning

A.1 Pre project

PRE-PROJECT REPORT

FOR BACHELOR THESIS

NTNU
Kunnskap for en bedre verden

TITLE:
Roe picking robot

CANDIDATE NUMBERS:
460020 460009 271779 460006

DATE:	COURSE CODE:	SUBJECT:	DOCUMENT ACCESS:
17.01.2018	IE303612	Bachelor thesis	- Open

STUDY:	PAGES/APPENDIX:	LIB.NUMBER:
AUTOMATION ENGINEERING	11/6	- NA -

SUPERVISOR(S):

- Ottar L. Osen - Associate Professor at NTNU
- Ibrahim A. Hameed - PHD, Associate Professor at NTNU

TASK/SUMMARY:

This pre-project report includes formalities about labour distribution, group rules and ambitions throughout a bachelor thesis in automation engineering. The assignment is given by NTNU who wants to develop machines that can automate processes in the aquaculture industry. The machine has to monitor roe during the hatching process and remove the ones that dies, thereby reducing the mortality.

The goal with the project is to develop a prototype that is able to perform the required task. This includes designing, programming and building the prototype with the usage of computer vision and image processing. A prototype will be built in order to achieve proof of concept.

Postadresse
Høgskolen i

N-6025
Norway

Besøksadresse
Larsgårdsvegen 2Larsgårdsvegen
7694 05 00636
Ålesund
www.hials.no

Telefon

EpostadresseEpostadresse
postmottak@hials.no

Telefax
70 16 12 0070 16 12 00 70 16 13 00

Bankkonto

Foretaksregisteret
NO 971 572 140

This assignment is an examination answer given by student(s) at NTNU campus Aalesund.

CONTENTS

CONTENTS **2**
1 INTRODUCTION **3**
2 TERM **3**
3 PROJECT ORGANIZATION **3**
3.1 PROJECT GROUP 3
3.1.1 Tasks for the project group – organization *3*
3.1.2 Responsibilities and tasks for project manager *4*
3.1.3 Responsibilities and tasks for secretary *4*
3.1.4 Responsibilities and tasks for other members. *4*
3.2 SUPERVISORS AND CONTACT PERSONS 5
4 APPOINTMENTS **5**
4.1 APPOINTMENT WITH EMPLOYER 5
4.2 WORKPLACE AND RESOURCES 5
4.3 GROUP NORMS – COOPERATION RULES – ATTITUDE 5
5 PROJECT DESCRIPTION **5**
5.1 PROBLEM DESCRIPTION 5
5.2 REQUIREMENTS FOR SOLUTION OR PROJECT RESULT - SPECIFICATION 6
5.3 PLANNED PROCEDURE(S) FOR DEVELOPMENT – METHOD(S) 6
5.4 GATHERING OF INFORMATION – GATHERED AND PLANNED 7
5.5 ASSESSMENT – ANALYSIS OF RISK 7
5.6 MAIN ACTIVITIES IN FURTHER WORK 7
5.7 PROGRESS PLAN – PROJECT MANAGEMENT 8
5.7.1 Master plan *8*
5.7.2 Management utilities *9*
5.7.3 Development utilities *9*
5.7.4 Internal control – evaluation *9*
5.8 DECISIONS – DECISION-MAKING PROCESS 9
6 DOCUMENTATION **10**
6.1 REPORTS AND TECHNICAL DOCUMENTS 10
7 PLANNED MEETINGS AND REPORTS **10**
7.1 MEETINGS 10
7.1.1 Meetings with supervisors *10*
7.1.2 Project meetings *10*
7.2 PERIODICAL REPORTS 10
7.2.1 Progress reports (incl. milestone) *10*
8 PLANNED DEVIATION MANAGEMENT **11**
9 EQUIPMENT/REQUIREMENTS FOR IMPLEMENTATION **11**
9.1 HARDWARE 11
10 REFERENCES **11**
APPENDIX **11**

1 INTRODUCTION

In the production of fish in aquaculture a critical part is the hatching process of roe. The roe is stored in trays with a constant flow of water for about 120-degree days (degree*days), before it hatches and becomes fish spawns. In this process there is a significant chance that the roe dies of series of different causes. Dead roe will start rotting and the rot will spread to the surrounding roe. Therefore, it is critical that the dead roes are removed as soon as possible.

Today the technique is humans inspecting the trays and manually removing the dead ones, which is a time-consuming and expensive process. In the project given by NTNU, the main goal for this project is to remove the human labor and intervention in the hatching process.

2 TERM

Roe – fish eggs

Mattilsynet – Food Safety Authority in Norway.

FSE - Regulations concerning safety at work in and operation of electrical installations

3 PROJECT ORGANIZATION

3.1 Project group

Student number(s)
460020
460009
271779
460006

3.1.1 Tasks for the project group – organization

Responsibility	**Group member responsible**
Project manager	460020
Secretary	271779

3.1.2 Responsibilities and tasks for project manager

Responsibilities:

- Gain an overview of the projects progression

Tasks:

- Make sure all group members are working with the project in a way that is productive and relevant to the desired solution.
- Coordinate with the group members to get an update of the projects progression

In addition, the project manager has the responsibilities and tasks listed in chapter 3.1.4.

3.1.3 Responsibilities and tasks for secretary

Responsibilities:

- Set up meetings
- Take notes during meetings

Tasks:

- Send meeting notice to everyone that should participate in the meeting
- Send an updated list of the projects progression before the meeting
- Take notes during the meeting
- Make sure that all project members write what they have done in the project

In addition, the secretary has the responsibilities and tasks listed in chapter 3.1.4.

3.1.4 Responsibilities and tasks for other members.

Responsibilities:

- The group members are responsible for the activities that are assigned to them in the project planning. In each task there will be a person responsible for the task solving and a person that will be helping in the solving of the task.
- The group members are responsible for writing what they have done for solving their tasks, what methods that were tried and why they decided on the chosen method in the end.

Tasks:

- Making sure that the activities assigned to each member is being worked on and the progression needed for finishing the activity in time is met.
- Writing project report through the entire project.

3.2 Supervisors and contact persons.

- Ottar L. Osen - Associate Professor at NTNU
- Ibrahim A. Hameed - PHD, Associate Professor at NTNU

4 APPOINTMENTS

4.1 Appointment with employer

A standard contract recommended by NTNU has been signed by both parties. The contract concerns rights of the company and the students. A more detailed overview of the contract can be seen in Attachment 2.

4.2 Workplace and resources

The group will do most work at the premises of NTNU campus Aalesund. There is an agreement that this facility is best suited, as both tools and useful knowledge are easy accessible if needed.
The supervisors at NTNU will be available and assist with programming and other challenges. NTNU has been willing to finance our project with a total budget of 5000 NOK. Currently they are the only financial resource.

4.3 Group norms – cooperation rules – attitude

- Core time of the work day will be Monday to Friday from 08.30 to 16.00. There will be no overtime pay or any form for allowance. Lunch break 11.30.
- Delays or sickness must be reported to group leader or other qualified group members.
- Group meeting 08.30 every Friday. Secretary takes notes from the meeting
- Group meeting every other Monday with supervisors, or et cetera.
- All group members must agree to a proposed solution before it is accepted.

5 PROJECT DESCRIPTION

5.1 Problem description

A challenge during the hatching process is roe dying before it is hatched. When this occurs, the roe can rot and infect its surroundings, and this increases the mortality among the roe.

Today human surveillance is used for identifying dead roe, which then are removed manually. This process is time-consuming and expensive

Another challenge in the fish farming industry is to determine whether a roe is possible to fertilize, it is possible to determine in the first stage of the life cycle. An infertile roe will increase the mortality among the roe population, in the same way as a dead roe. Today humans use magnifying glass for determining the possibility of fertilization. This is also an ineffective process.

5.2 Requirements for solution or project result - specification

The machine will be a prototype made by easy accessible and low-cost components for achieving proof of concept. The prototype must be able to pull one tray from a rack-system. When the tray is pulled, the machine must scan the tray and find the dead roe. If, and only if, the machine finds a dead roe it will be removed from the tray and then the search for more dead roe will continue. When the machine has controlled the entire tray, it will push the tray back into the rack and move onto the next tray. The machine needs a user interface that is easily operated by instructed personnel. The machine will keep statistics over the roe population in each of the trays and the amount of removed roe. A finished product needs to meet the requirements from Mattilsynet and FSE.

Machine cycle:

1. Pull a tray from the rack.
2. Scan the tray for dead roe.
3. Remove dead roe if found.
4. Push the tray back into the rack.
5. Move the scanner to the next tray in the rack.
6. Do point 1 to 5 until all eight trays have been scanned.
7. Do 1-6 cycles a specified amount of times in a given timeframe

5.3 Planned procedure(s) for development – method(s)

We are not familiar with the development method for machines similar with the one we are planning to construct. We see a resemblance with our construction some types of linear robots and 3D-printers. We want to use these machines in order to get information and ideas for our solution.

During the project, the group plan to use much time on planning and drawing of the machine. Extensive planning will lead to less trouble when constructing the machine and hopefully avoiding large software and hardware problems.

5.4 Gathering of information – gathered and planned

Gathered:

The project is solved in cooperation with NTNU and Moereforsking. Moereforsking holds extensive knowledge about aquaculture and are very interested in following the progress of the project.

Planned:

In order to plan and construct a prototype that meets the requirements of the tasks to be solved we are in need of information:

- About which components that will be best suited for our project, we need to consider both cost and quality.
- On different technologies that we can use for detecting the dead roe
- About the guidelines from Mattilsynet and Miljoedirektoratet.

5.5 Assessment – Analysis of risk

Frequency			
High			
Medium	2.	1	2 5
Low	4.	1. 3. 4	3 5
Consequence	Low	Medium	High

Risk	Taken measures - Reduces risk
1 Sickness	1. Eat your vitamins
2 Delay in shipping or ordering part	2. Plan ahead. Work with substitute parts or work on something else while waiting
3 Computer destroyed or crashed	3. Cloud saving
4 Reworking a solution	4. Good planning. Working out a possible solution before starting the work.
5 Damaging components	5. Quality of components. Proper usage and testing.

5.6 Main activities in further work

Nr	Main activity	Main responsible	Activity responsible	Time
A1	Bachelor report	KH	Everyone	135 days
A2	Planning	PEA	Everyone has subtasks	18 days
A3	Drawing	YB	Everyone has subtasks	23 days
A4	Programming	PEA	Everyone has subtasks	35 days
A5	Construction	KH	Everyone has subtasks	15 days

A6	Testing	KAL	Everyone has subtasks	35 days
A7	HSE	KAL	Everyone	135 days

The scheduled time for each of the tasks above is set to overlap. We use overlapping assignments because during the execution one can be prevented from completing a task. For example, during construction you may have to wait for parts, this time can be used on tasks that are scheduled to be done later on. A more detailed plan of execution can be seen in Attachment 3.

5.7 Progress plan – project management

5.7.1 Master plan

A1 Bachelor report

Everybody in the project will be writing the report throughout the entire project in order to ensure that procedures and methods are documented while they are still fresh in memory.

A2 Planning

Making a detailed plan will reduce the faults later in the project. There will be made concept drawings for all the main parts of the project. The concept drawings shall contain all the equipment and measurements needed to complete each part of the project. During the planning there will be made an equipment list and parts will be ordered.

A3 Drawing

Make detailed drawings for each part of the project. There will also be made mechanical, electrical, pneumatic and system -drawings.

A4 Programming

Programming will play a central role in this project. With computer vision and robot control being two of the biggest tasks regarding programming. There will be used open source libraries and different coding languages on the respective systems. Communication between the systems will be coded using a specified and known protocol. Human-machine interface is also one of the tasks needed to be solved.

A5 Construction

Constructing is a time-consuming part of the project. This consist of constructing an elevated rack system and a linear robot.

A6 Test and calibration

Test and calibration will be a major part in the end of the project. The work will include extensive hardware and software testing and optimization.

A7 HSE

During the building and operating of the machine it is important to consider the HSE. Dangerous tools will be used and the machine will contain moving parts.

5.7.2 Management utilities

- Asana with Instagantt for project planning and time scheduling.
- LaTex for writing the report with the template from Master Thesis by NTNU.
- Dropbox and OneDrive for filesharing.

5.7.3 Development utilities

- Enviroments for programming: Netbeans, PyCharm and Arduino IDE
- Open CV library for computer vision.
- Autodesk fusion 360 for mechanical drawings.
- PC schematic for electrical drawings

5.7.4 Internal control – evaluation

A report concerning the projects progress will be written every 2nd week. This report will be the foundation of discussion in the meeting between the group and the supervisors being held a few days later.

In order to accept a solution to a task it has to be verified by the whole group. The solution should meet the projected quality set by the group.

5.8 Decisions – decision-making process

During the pre-project the decisions concerning delimitation and specification of the project is done in cooperation with the supervisors. The specifications are chosen in order to create a prototype able to automate the hatching process of fish production.
During the project, the group plans to use the same method for decision making as in the pre-project.

6 DOCUMENTATION

6.1 Reports and technical documents

- Electrical drawings.
- Pneumatic diagram.
- Mechanical drawing
- Bachelor thesis.
- User manual.

All documents will be stored on a cloud storage.

7 PLANNED MEETINGS AND REPORTS

7.1 Meetings

7.1.1 Meetings with supervisors

The group has planned to have a meeting with all the group members and the supervisors every two weeks. In advance of these meetings the meeting participants will get an updated report of the project progression.

7.1.2 Project meetings

The group has planned to have a meeting each Friday at 08.30 for updating the Instagantt with the current progression, discussing problems met and possible solutions to the problems. The core topics from the meeting will be noted.

7.2 Periodical reports

7.2.1 Progress reports (incl. milestone)

There will be written a report at the end of every 2nd week. The report will contain the progress of the project, which problems were met and other relevant topics. These reports will be used for discussion in the meetings with the supervisors. A small summary of the work done will be written every week by each member of the group.

8 PLANNED DEVIATION MANAGEMENT

If the progress is behind schedule, the group will gather for a meeting and revise the arranged plan. In case of no solution to a problem is found voting will take place and less important content will be removed in order to complete a sufficient product. The project leader will have the responsibility to initiate the vote, every group member will have an equal vote.

9 EQUIPMENT/REQUIREMENTS FOR IMPLEMENTATION

9.1 Hardware

In the construction of the roe picking robot a list of hardware components is needed:

- Computer for processing data
- Microcontroller for I/O
- Aluminum profiles
- Electrical motors
- Drivers for electrical motors
- Linear bearings
- Transformer 230/12V
- Motor timing belt
- Camera or another device for locating roe
- Cables

10 REFERENCES

APPENDIX

Attachment 1: Proposal for bachelor thesis from NTNU

Attachment 2: Standard agreement between company and student

Attachment 3: Labor distribution

A.2 Original project plan

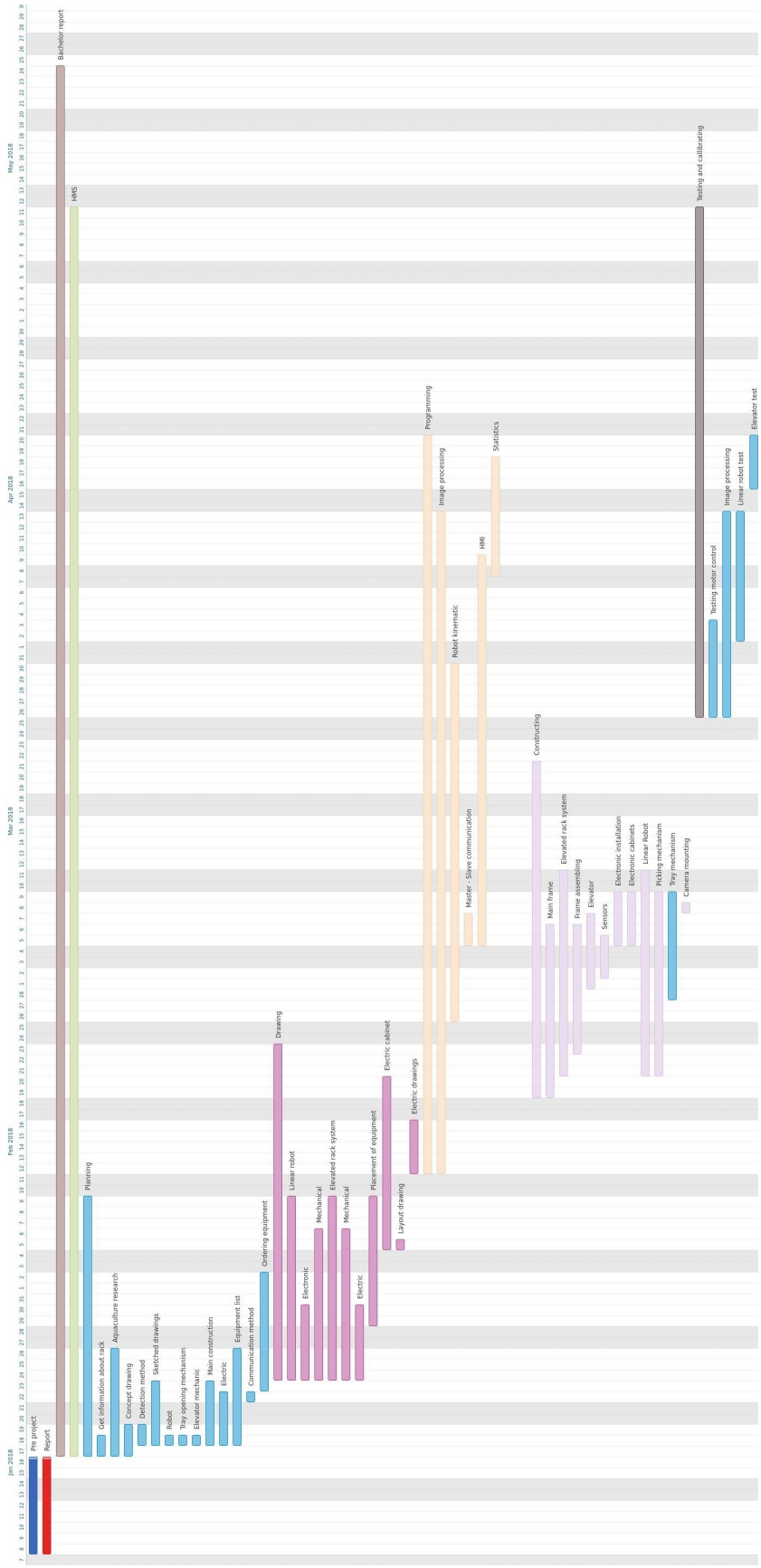
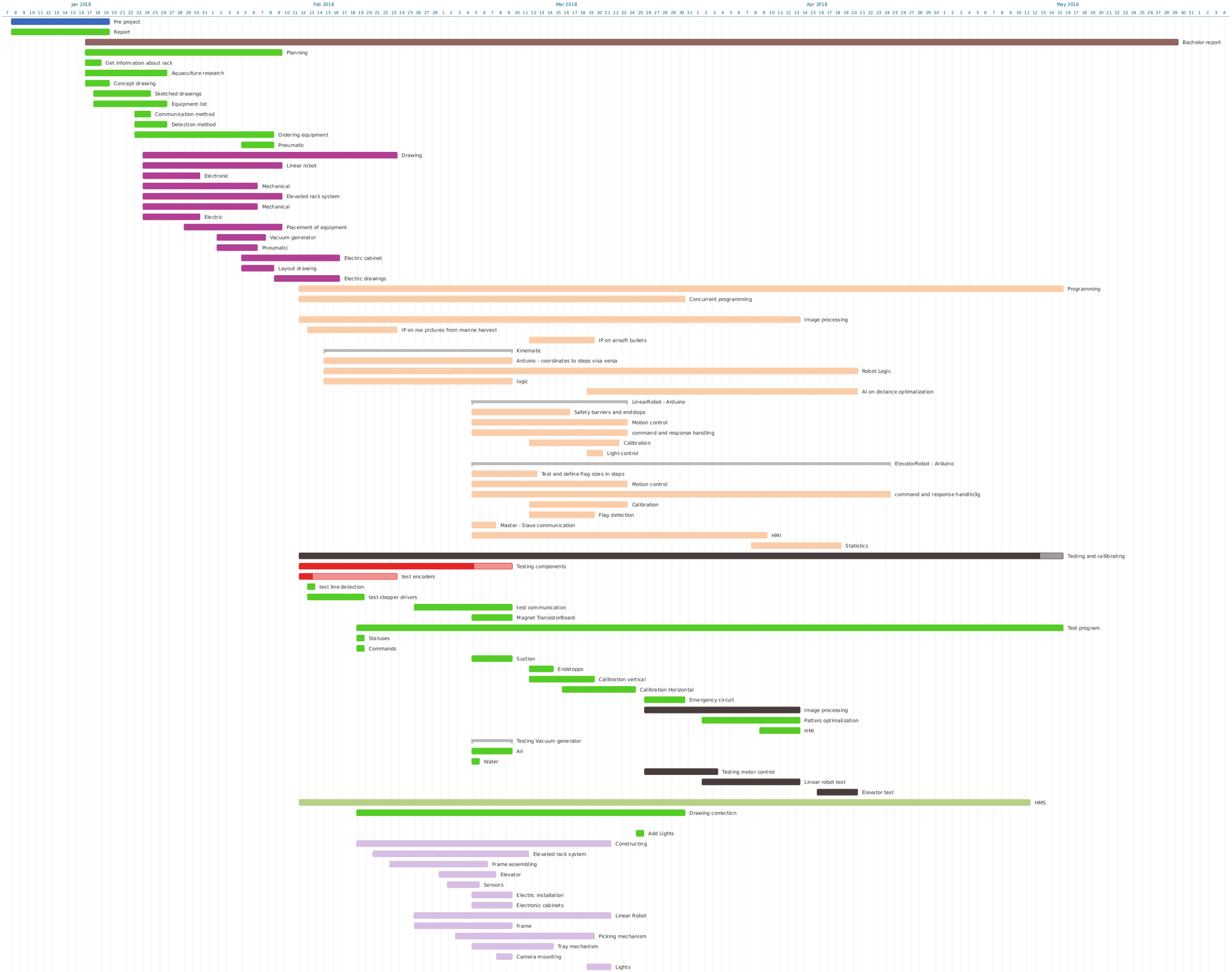


Figure A.1: Original project plan

A.3 Final project plan


Jan 2018
Feb 2018
Mar 2018
Apr 2018
May 2018
Pre project
Report
Bachelor report
Planning
Get information about rack
Aquaculture research
Concept drawing
Sketched drawings
Equipment list
Communication method
Detection method
Ordering equipment
Pneumatic
Drawing
Linear robot
Electronic
Mechanical
Elevated rack system
Mechanical
Electric
Placement of equipment
Vacuum generator
Pneumatic
Electric cabinet
Layout drawing
Electric drawings
Programming
Concurrent programming
Image processing
IP on roe pictures from marine harvest
IP on airsoft bullets
Kinematic
Arduino - coordinates to steps visa versa
Robot Logic
logic
AI on distance optimalization
LinearRobot - Arduino
Safety barriers and endstops
Motion control
command and response handling
Calibration
Light control
ElevatorRobot - Arduino
Test and define flag sizes in steps
Motion control
command and response handlin3g
Calibration
Flag detection
Master - Slave communication
HMI
Statistics
Testing and callibrating
Testing components
test encoders
test line detection
test stepper drivers
test communication
Magnet TransistorBoard
Test program
Statuses
Commands
Suction
Endstopps
Calibration vertical
Calibration Horizontal
Emergency circuit
Image processing
Patters optimalization
HMI
Testing Vacuum generator
Air
Water
Testing motor control
Linear robot test
Elevator test
HMS
Drawing correction
Add Lights
Constructing
Elevated rack system
Frame assembling
Elevator
Sensors
Electric installation
Electronic cabinets
Linear Robot
frame
Picking mechanism
Tray mechanism
Camera mounting
Lights

A.4 Timesheet

Timeliste Kristoffer			Timeliste Kristian			Timeliste Yngve			Timeliste Per Espen		
Dato:	Antall timer:	Kommentar:	Antall timer:	Kommentar		Antall timer:	Kommentar:		Antall timer:	Kommentar:	
8.01.18	7	Forprosjekt	8,50	Forprosjekt		7,5			7	Forprosjekt	
9.01.18	7,5	Forprosjekt	8,5	Forprosjekt		8			7,5	Forprosjekt	
15.01.18	7	Forprosjekt	8,5	Forprosjekt		7			7	Møte/forprosjekt	
16.01.18	7	Forprosjekt	8,5	Forprosjekt		7,5			7,5		
17.01.18	2	Forprosjekt	7,5	Forprosjekt		8			7,5		
18.01.18	7,5	Forprosjekt	8	Forprosjekt, finne deler		8			7		
22.01.2018	7	Utflukt	8,5	Marine Harvest ytre Standal, finne deler		8	Utflukt hos Marine Harvest ytrestandal		7	Marine Harvest ytrestandal	
23.01.2018	7,5	finne deler, tegne main constructio	8,5	Finne deler, tegne main construction		9,5	Finne deler, tegne liear robot		7,5	Finne deler, tegnet	
24.01.2018	7,5	Tegning	8,5	Tegne main construction, skjære ut traller		10	Tegning av lineær robot		8		
25.01.2018	7,5	Rapportskriving		ikke tilstede		3	Tegning av lineær robot		1	Tegnet	
26.01.2018	7,5			ikke tilstede					3		
29.01.2018	7,5	Bestille deler	9	lage bestillingsliste		8			8		
30.01.2018	7,5	Rapportskriving		syk		8	Rapport		8		
31.01.2018	0	Syk		syk		9,5	Vakuum design. Robot design.		7		
01.02.2018	7,5		8,5	rapportskriving		8			6,5		
02.02.2018	7,5	Rapportskriving	12	tegning av heismekanisme		7,5	Venturi Generator		7		
05.02.2018	8	Møte, forprosjektrapport	9,5	møte med veileder, heis med gjenger, forpro		10			8		
06.02.2018	7,5		9,5	research threaded rod vs timing belt, tegne v		8			9		
07.02.2018	7,5	Bestilling av deler	9	rapport, tegne verktøy		9			8		
08.02.2018	9	Finne info om atepper motor	11	tegne endebrytere, lodde endebrytere, heis		5			8,5		
09.02.2018	7,5		10	ferdigstille heis, lage pulleyu fester		10	Ferdistilling av robot tegninger og reserch om ki		9		
10.02.2018 lørdag	0		8	printe deler, skrive rapport					0		
12.02.2018	9		12	printe deler, skrive rapport, kappe metall, set		10			9,5		
13.02.2018	12		14	bestille skruer, hente kommode, planlegge pr		8,5			12		
14.02.2018	8		12	tegne, printe, programmere, diskutere		7			7		
15.02.2018	8		11	diskutere kommunikasjon, print, BB		10			8		
16.02.2018	7		14,5	diskutere, printe, BB		13			8		
17.02.2018 lørdag	0		6	bygge main construction :D		5			1		
19.02.2018	6		10			9	Program struktur		7,5	Program struktur	
20.02.2018	9		11	BB		10	Program metoder		9	Programmere genrell I2CCommunication	
21.02.2018	9		10	BB		9	Program metoder		7,5	Sette opp generell program struktur	
22.02.2018	8		5	BB		9	Program GA pattern		8	programmere videre på I2C communication	
23.02.2018	8		12	BB, rapport			15 kilometer klassisk Råkollen		8	Videreutvikle I2CCommunication	
24.02.2018 lørdag		søv	2	Rapport			10 kilometer klassisk				
26.02.2018	8,5		11	Rapport, møte		9			7,5	Rapport	
27.02.2018	10		9	BB, rapport		9			6,5 + 1,5	Rapport	
28.02.2018	0		11	Bygge El-skap + div		8			10		
01.03.2018	0		11	koble El-skap, skjære hjørner		9			8		
02.03.2018	0		9	Rapport, stramme ramme		9			6,5	Rapport :D	
03.03.2018 Lørdag	5	kinematikk							0		
04.03.2018 Søndag	5	Rapportskriving							0		
05.03.2018	11	kinematikk	11	programmere GUI, lage kalibreringsmetode x		10			9	Utfylling og implementering av RoeDevice	
06.03.2018	8	kinematikk	13	GUI, montasje, Arduino, test kalibrer		8			9,5	testing og feilsøking av i2c dritt	
07.03.2018	7,5	Bresenham	7,5	GUI		8			6+1,5		
08.03.2018	8	kinematikk	10	Bygge linearbot		8			7,5		
09.03.2018		Jobbintervju	13	Bygge bygge :		8			8		
10.03.2018 Lørdag									0		
11.03.2018									0		
12.03.2018		Jobbintervju	10	Rapport		9			8		
13.03.2018	7,5	Rapportskriving	9	Review		9			8		
14.03.2018				industri 4.0					2		
15.03.2018			2	industri 4.0, airhockey lokk		1	Printing av deler		0		
16.03.2018				industri 4.0		1	Printing av deler		0		
17.03.2018 Lørdag									0		
18.03.2018 Søndag									0		
19.03.2018	9	Rapportskriving	10	rapport		9	Montere og redesigne		8		
20.03.2018	8	Bolleeting og arduino	9	koble elskap		11	Montere og redesigne		8		
21.03.2018	7,5	Rapportskriving	11	montere ting og tang		10	Montere og redesigne		8		
22.03.2018	8	Test	9	funksjonstesting av heis		9	Montere og redesigne		9		
23.03.2018	7	Rapportskriving	2	Kvistian var på effesE kurs, rapport		7	Montere og redesigne		4		
24.03.2018 lørdag									0		
25.03.2018 Søndag									0		
26.03.2018	7,5		8	Rapport		9	Rapport skriving		8		
27.03.2018	7,5		13	Rapport		6	Rapport skriving		7,5		
28.03.2018	4		3	Rapport		8	Rapport skriving				
29.03.2018 påske						1,00	Rapport lesing				
30.03.2018 påske											
31.03.2018 påske			2	Lese rapport		1,00	Rapport lesing				
01.04.2018 påske			1	lese rapport							
02.04.2018 påsk	2		1	lese rapport		1,00	Rapport lesing				
03.04.2018	8		7,5	fikse kommode osv.		13,00	Kobling og montering		10	Arduino programmering, testing, montering	
04.04.2018	7		14	koble og montere					10	Testing av ard og koblinger.	
05.04.2018	7		8	programmere listener							
06.04.2018	8		10	rapport							
07.04.2018											
08.04.2018											
09.04.2018											
10.04.2018									2	i2c testing, fant feil	
11.04.2018									1,5	i2c testing	
12.04.2018									2,5	i2c party	
13.04.2018									3,5	i2c dritt	
14.04.2018	5		3	rapport					2,5	i2c sjukt fest	
15.04.2018											
16.04.2018	12	GUI	12	Rapport, testing i2c		12	I2c testing.		10		
17.04.2018	11	GUI	13	bildebehandling, testing		13	Serial coding		11,5	Kika på serial	
18.04.2018	11		11	Bildebehandling		11	Serial coding		10,5	Koda ny serial klasse	
19.04.2018	9		11			11			11	Håndtering av serial reader writers	
20.04.2018	8		10			10			10		
21.04.2018 Lørd	6								2,5		
22.04.2018 Søndag											
23.04.2018	10	GUI	13	Bildebehandling rapport		12			12		
24.04.2018	11	GUI	11	bildebehandling, testing		10			10,5		
25.04.2018	8		8	Rapport		8			7		
26.04.2018 syk			10	Rapport		10			10,5		
27.04.2018 syk			9	rapport		12			12		
28.04.2018 Lørdag						4			4		
29.04.2018 sønc	4	Rapportskriving									
30.04.2018	11	Rapportskriving	9	Rapport		9	Rapport		7,5	Rapport	
01.05.2018	11	Rapportskriving	10	Rapport		8	Rapport		8,5	Rapport, og knakk nøke på roboten	
02.05.2018	7	Rapportskriving	11	Rapport, testing		12	Rapport, testing		10	Rapport, testing	
03.05.2018	8	Rapportskriving	10	Rapport, testing		12	Rapport, testing		12	Rapport, testing	
04.05.2018	6		10	Rapport, testing					11		
05.05.2018 Lørd	4								4		
06.05.2018 søndag											
07.05.2018	3		9	Rapport, testing		10			10		
08.05.2018	8		10	Rapport, testing		10			10		
09.05.2018	0		11	Rapport, testing		11			11		
10.05.2018	10		10	Rapport		12			8		
11.05.2018	8	rapport	10	Rapport, testing		11			11		
12.05.2018	9	rapport	3	Testing		0			4		
13.05.2018 Sønc	9	rapport	6	rapport		12					
14.05.2018	14	rapport	11	Rapport		14			11		
15.05.2018	10	rapport	9	Rapport		16			10		
16.05.2018	10	rapport	9	Rapport		10			11		
17.05.2018									9		
18.05.2018	10	rapport	4	Rapport		7	Rapport		12		

19.05.2018	Lørd	8	rapport	10	Rapport	9	Rapport	7
20.05.2018	Sønc	6	rapport	5	Lese rapport	9		1
21.05.2018		12		13	Rapport	13		12
22.05.2018		13		13	Rapport	13		11
23.05.2018		12		12	Rapport	13		11
24.05.2018		13		13	Rapport	13		12
25.05.2018		10		10	Rapport	5		9
26.05.2018	lørd	7		7	Rapport	8		7
27.05.2018	sønc	11		11	Rapport	11		7
28.05.2018		14		14	Rapport	15,50		14
29.05.2018		13		13	Rapport	14,5		12
30.05.2018		13		13	Rapport	14,5		12
31.05.2018		3		7	Rapport	7		2
01.06.2018	leveringsfrist							
total		**757,5**		**910,00**		**809,00**		**791,50**
Totalt antall timer			**3268,00**					

Appendix B

Marine harvest visit

ISSUE
1
MONTH: 4
YEAR: 18

Marine Harvest

Besøket P.1

Klekkeriet på Ytre Standal P.2

Marine Harvest

Marine Harvest har eksistert siden 1965. De startet opp som en liten gründerbedrift, men har nå et godt fotfeste som et av verdens ledende oppdrettsselskap. De tar hele produksjonen fra rognkorn til filleten serveres på borde. De satser på forskning som skal utvikle fremtidens løsninger for fiskeoppdrett og er en viktig pådriver for innovasjon både i Norge og internasjonalt. (Hentet fra marinehavest.no)

Marine Harvest i Norge

- Norges største oppdrettsselskap
- Dekker hele verdikjeden
- Har virksomhet langs hele kysten fra flekkefjord i sør til Kvænangen i nord.
- Selskapet er en del av konsernet Marine Harvest ASA som har virksomhet i 24 land og er børsnotert. Aksene handles på det amerikanske markedet. Det globale hovedkontoret ligger i Bergen

En senvinterdag I Februar tok vi en telefon til gutta i Marine Harvest. Vi la ut om informasjonen vi trengte for å gjøre bacheloroppgaven vår så realistisk som mulig. Nærmest før vi hadde snakket ferdig ønsket de oss hjertlig velkommen når vi måtte ønske. Dagen etter satt vi på den korresponderende fergen over til Festøy med ei svele i hver hånd.

Daglig leder Frode Sætre ønsket oss velkommen med et bredt smil om munnen. Han inviterte oss inn rundt bordet hvor resten av de ansatte hadde lunsj. Innledningsvis satt praten løst om hva de bedrev i den daglige driften. Deretter fikk vi anledning til å stille generelle spørsmålene vi måtte lure på. Og hvem er vel bedre til å svare enn personer med over 20 års fartstid i oppdrettsnæringen?

«Hvordan produserer dere rogn?» Lød det første spørsmålet. De så på hverandre, litt oppgitt, så på oss, og la ut: vi produserer ikke rogn selv. Det er kun et fåtall rognprodusenter i Norge, og disse eksporterer ut til de som handler. Vi driver rognproduksjon fra øyrognstadiet, hvor de da blir lagt i egne klekkeskap. Her blir de liggende i 7-15 dager.

«Den er grei. Det hender at rogn dør i klekkeprosessen, hvordan håndterer dere dette? » Her er dere absolutt på ball, sier en av de ansatte, og legger til; noen dør, men de fleste overlever. Sånn er naturen bygd opp. Uansett, vi plukker de ut manuelt med en sugepinsett. Her er rognproduksjonen såpass liten at vi ansatte har flytende oppgaver. Og da når det er mindre å gjøre generelt på anlegget, tar vi en runde innom klekkeanlegget. Da drar vi ut skuff for skuff og plukker ut de døde. Siden disse er hvit, grunnet edikken som vannet er tilsatt med, er det ikke noe problem å finne de. Man trenger ikke å stå med mikroskop, for å si det sånn.

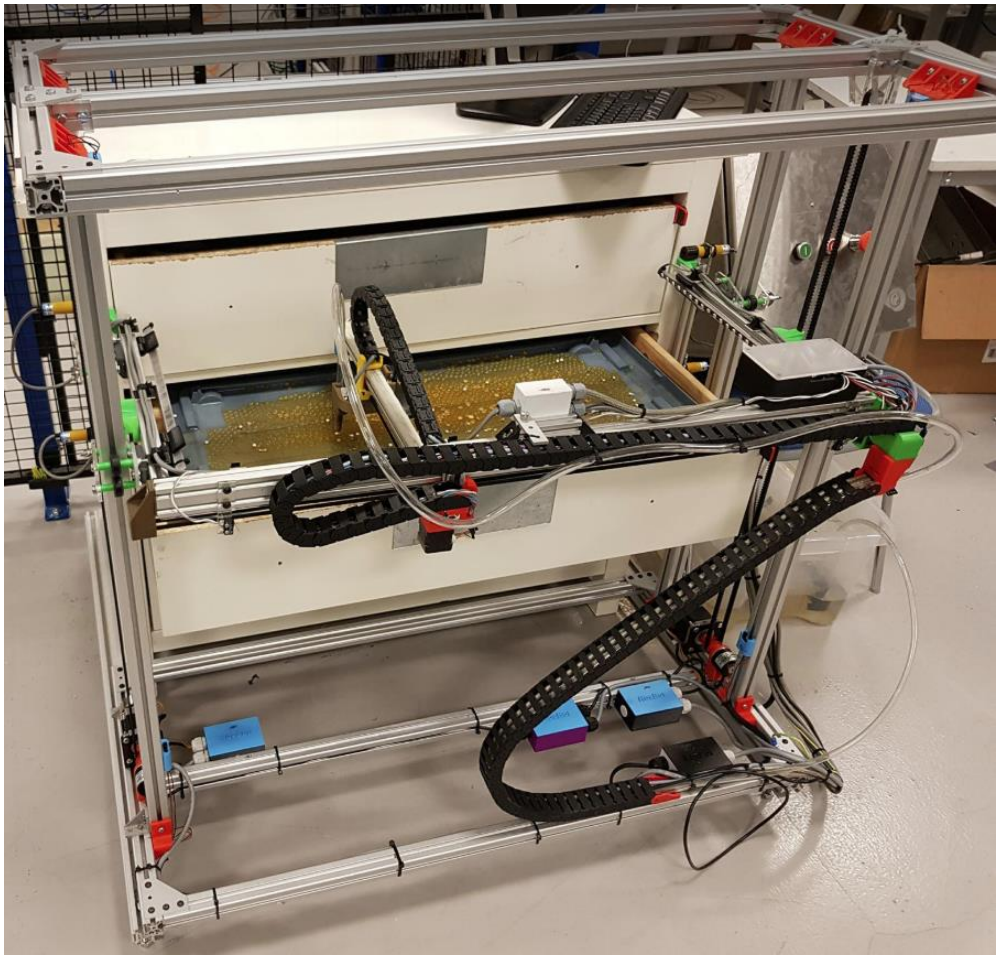
«Dere sier at dere plukker ut de døde manuelt. Hvor stort problem utgjør er dette?» Nei, egentlig er det ganske problemfritt. Så klart, på hektiske dager kan de t by på utfordringer, men veldig sjeldent problem. I snitt har vi rundt 100 utplukkinger på en batch. Så for vår del dreier det seg ikke om noen høye tall. Men så har vi en liten produksjon, også. Det er viktig at vi plukker de ut så raskt som mulig, siden de rotner raskt og forgifter de resterende rognkornene. Dermed er det en viktig oppgave å ha noen til å sjekke rognen.

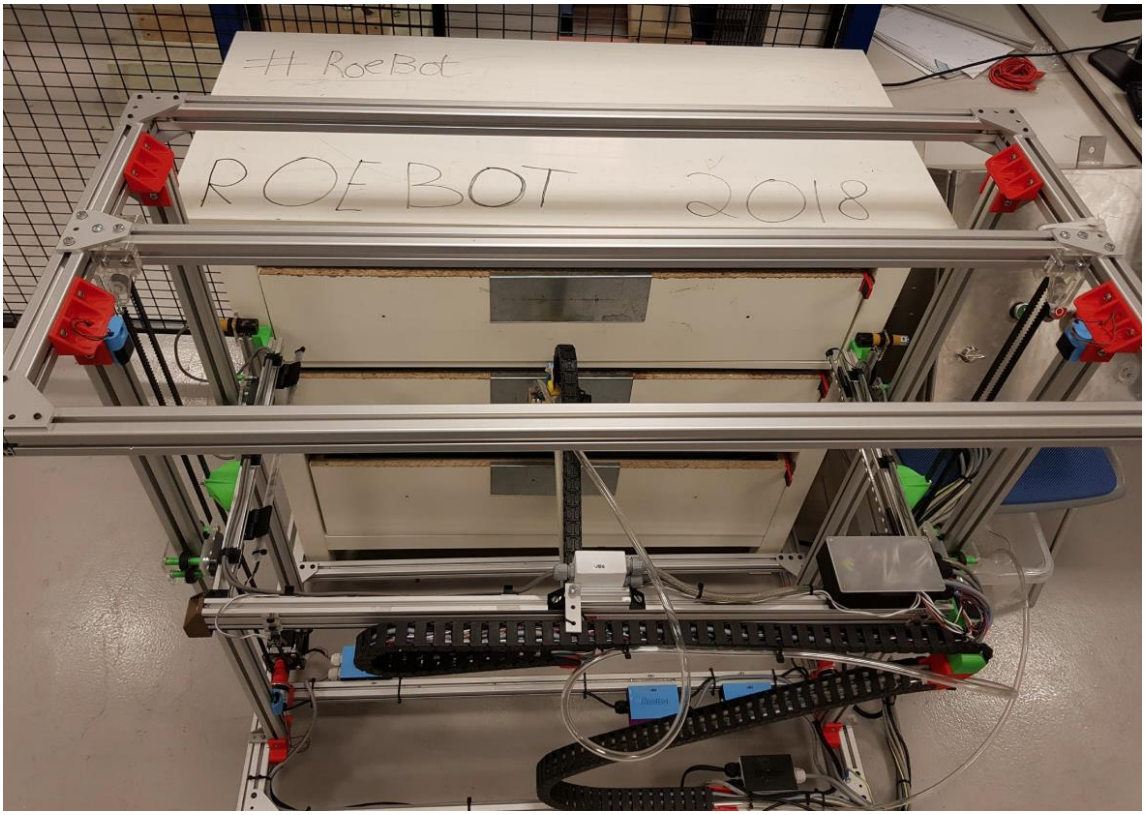
NTNU

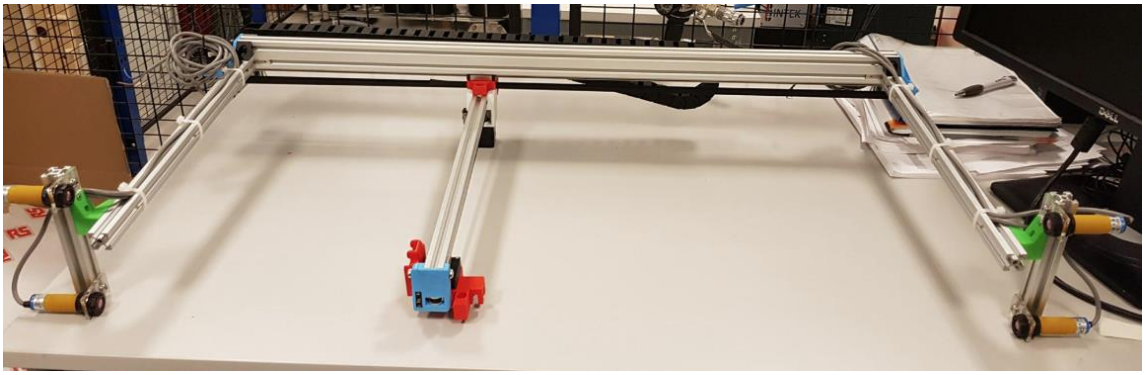
Vi fikk deretter anledning til å ta en titt rundt om på anlegget. Prekært var klekkeriet hvor mye godt bildemateriale ble produsert. Under ser man et bilde av klekkeriet på Ytre Standal.

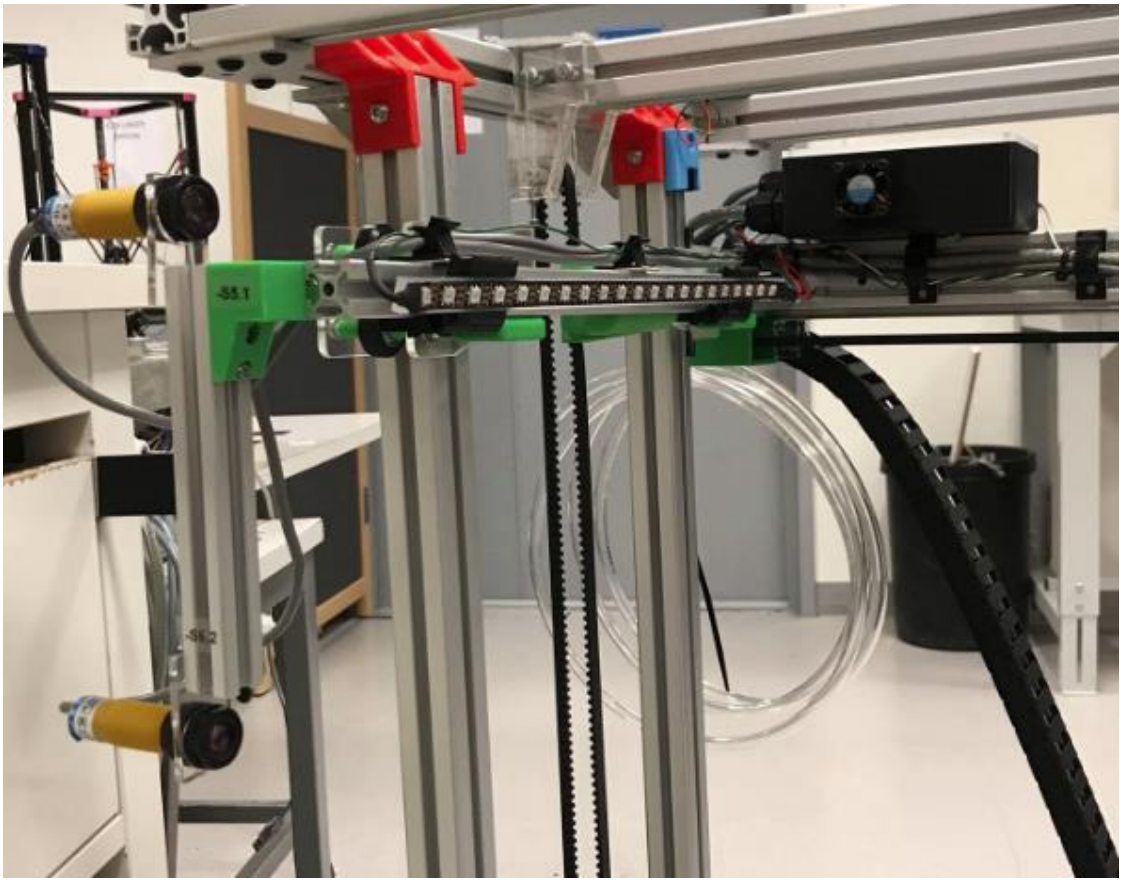
Appendix C

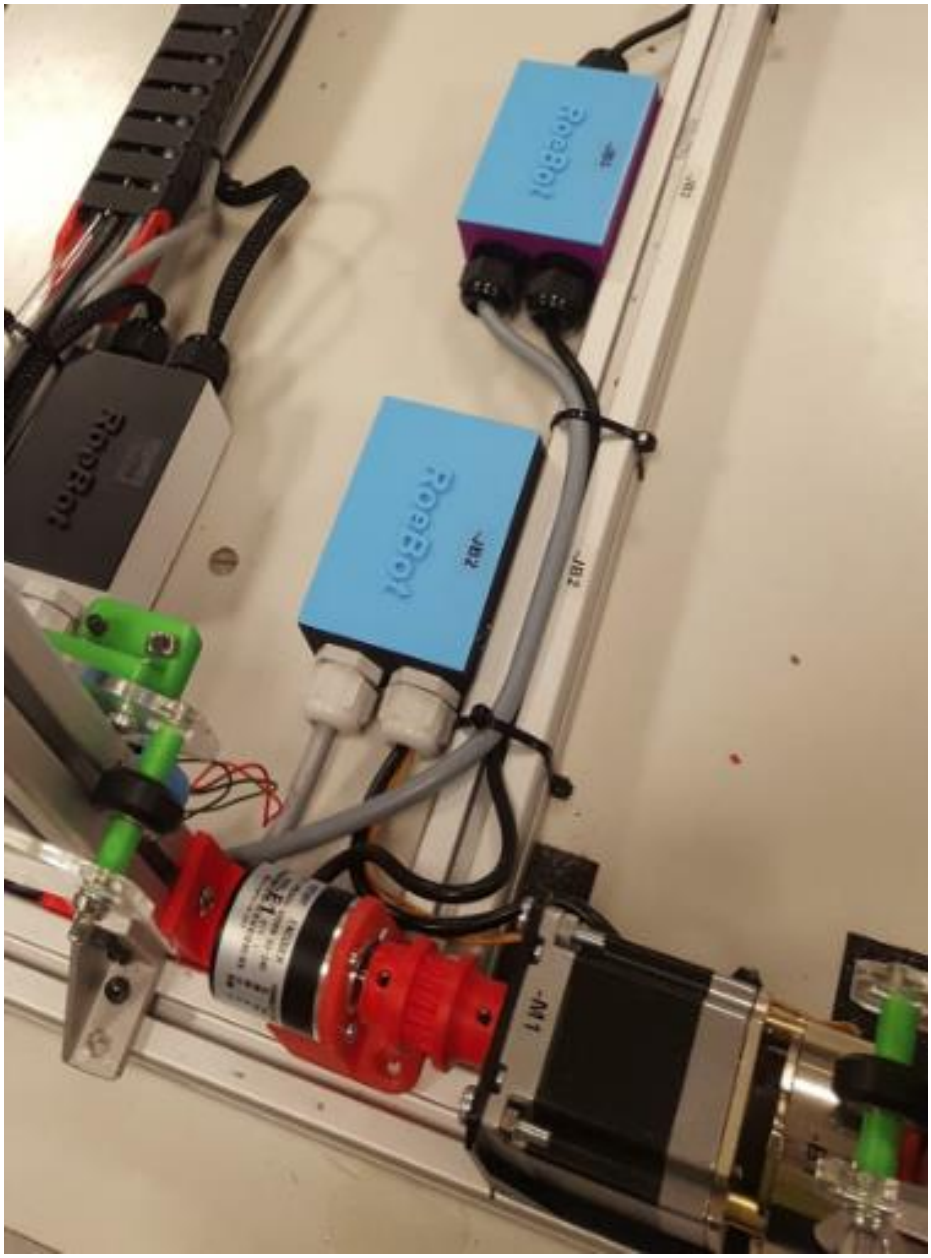
Images from assembling

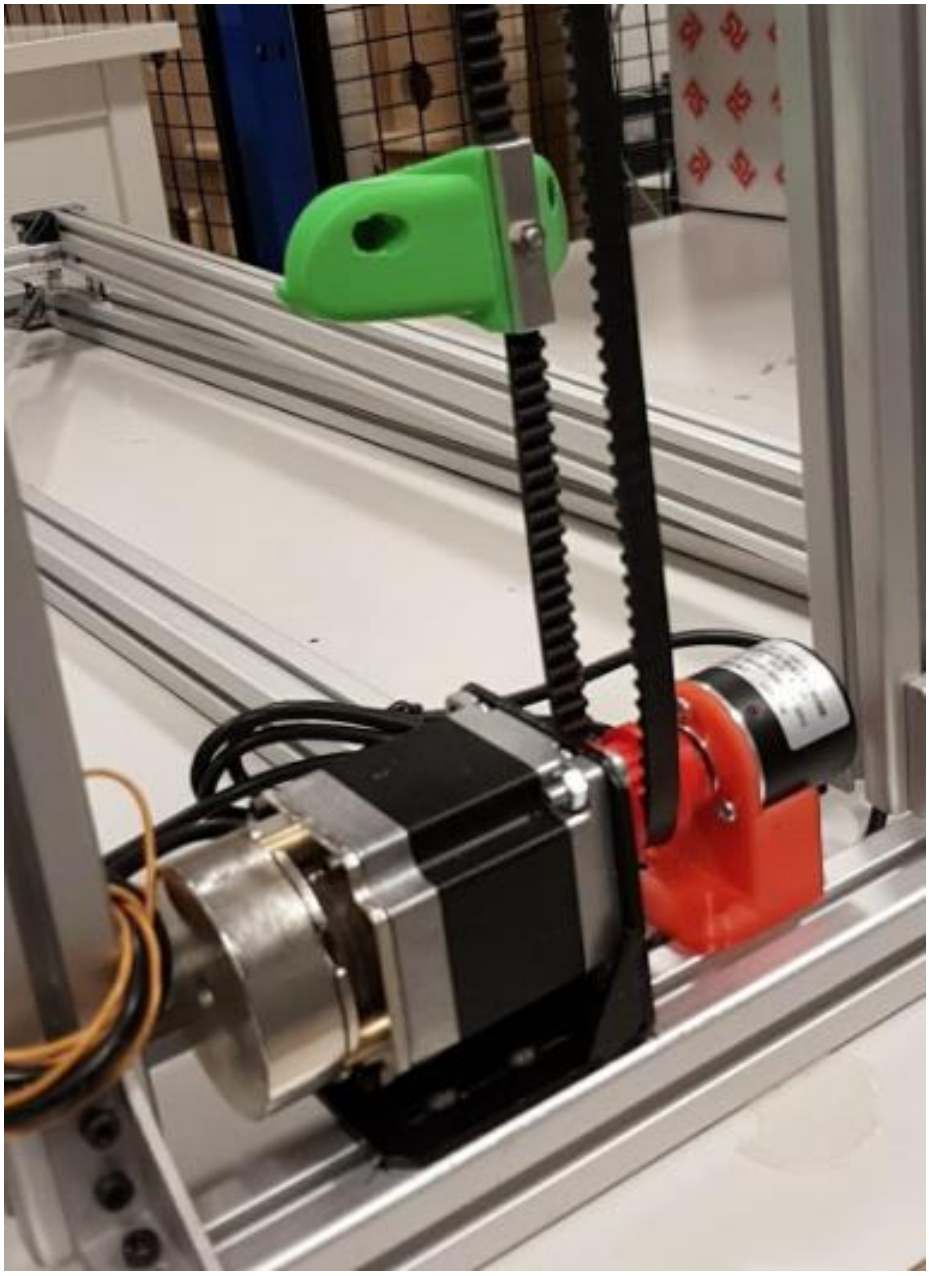


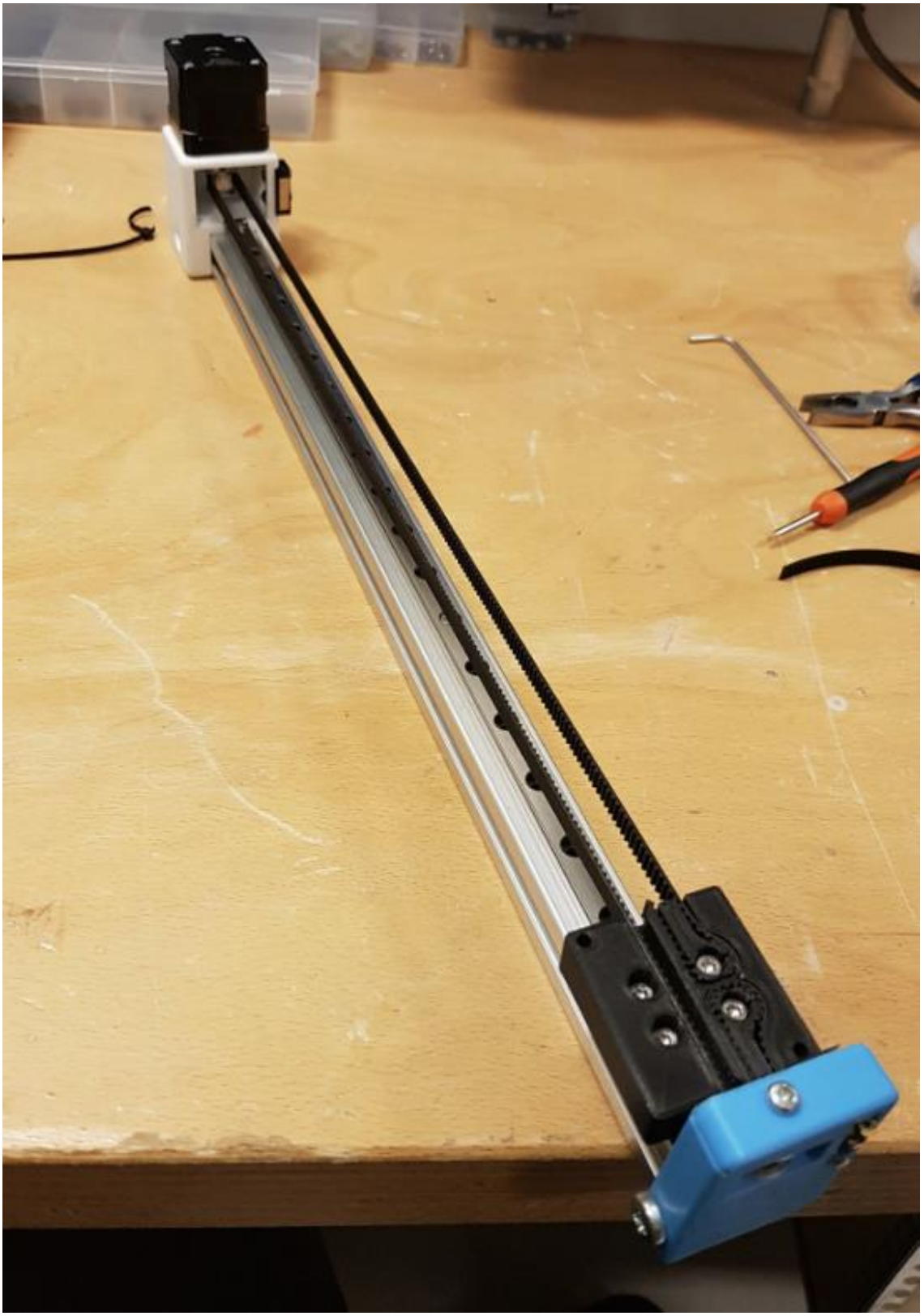

#1 RoeBot
ROEBOT 2018

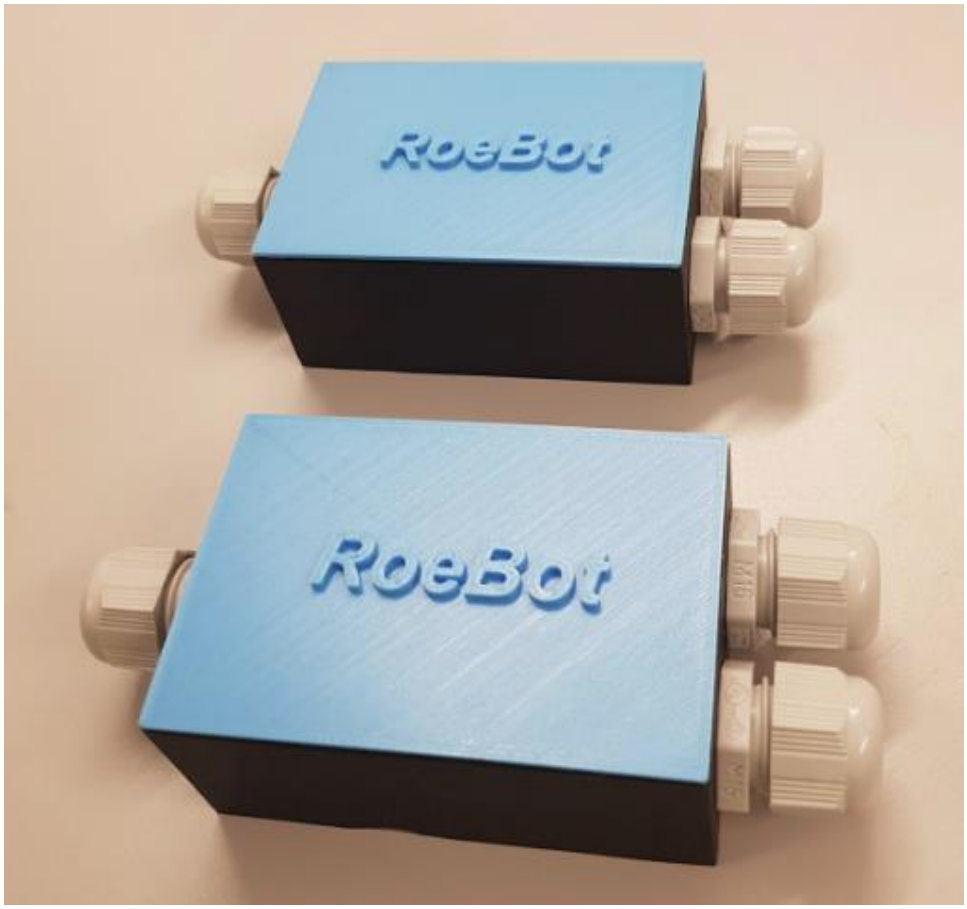


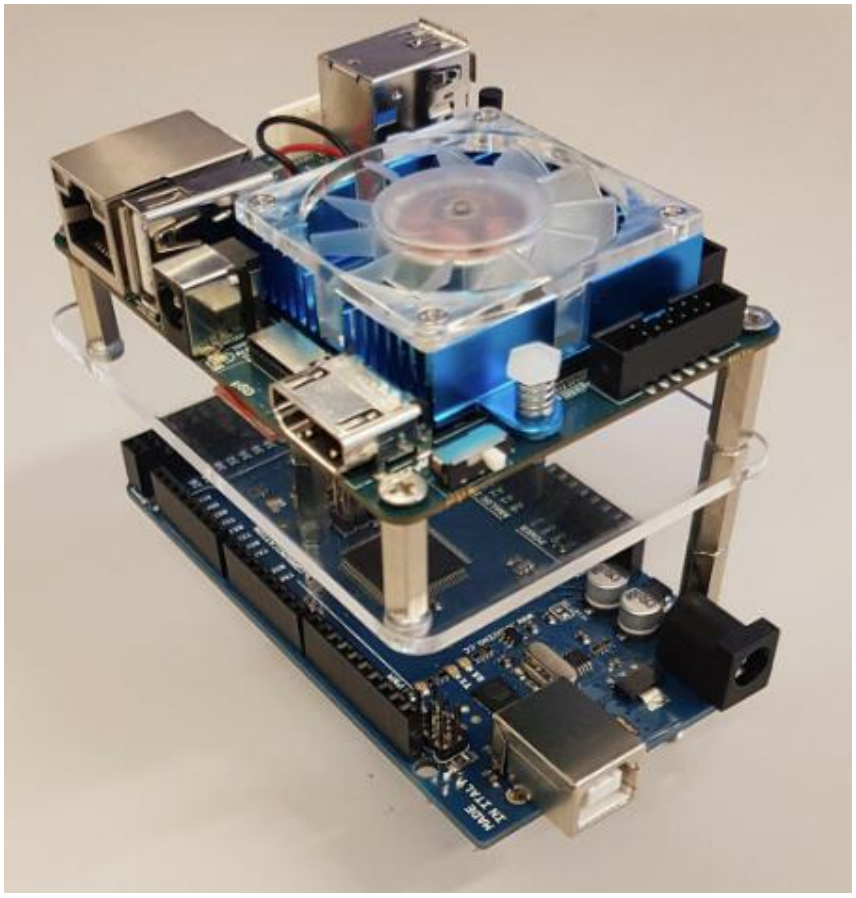

-S5.1


ROBOT
ROBOT
ROBOT






RoeBot
RoeBot



Appendix D

Bill of material (BOM)

D.1 BOM

Component	Description	Manufacturer/supplyer	Quantity	Price Each	Total Price
Eletrical cabinet					
Odroid XU4	Main controller	Hardkernel	1	kr 477,00	kr 477,00
Arduino Mega	Vertical controller	Arduino	1	kr 311,59	kr 311,59
Arduino Nano	Horizontal controller	Arduino	1	kr 178,00	kr 178,00
DM556	Vertical motor driver	NA	2	kr 145,68	kr 291,36
PS5R-SG24	24V power supply	IDEC	1	kr 2 249,00	kr 2 249,00
MDR-40-12	12V power supply	MEAN WELL	1	kr 135,00	kr 135,00
DR-15-5	5V power supply	MEAN WELL	2	kr 107,26	kr 214,52
A9F84203	2P 3A 230V fuse	Schneider Electric	2	kr 400,00	kr 800,00
A9F06206	2P 6A 230V fuse	Schneider Electric	1	kr 470,00	kr 470,00
857-304	24V relay	Wago	2	186,25	kr 372,50
Optocoupler 10A	5V Relay	Robotdigg	2	kr 8,00	kr 16,00
Optocoupler 10A x4	5V Relay module	Ebay	1	kr 35,00	kr 35,00
Vertical robot					
Aluminium Alloy Strut	30 x 30 mm 6mm Groove	Robotdigg	4	kr 433,00	kr 1 732,00
L-connector	20x20 mm	Robotdigg	16	kr 2,50	kr 40,00
Set Screw	M4*5 x 20	Robotdigg	2	kr 3,00	kr 6,00
T nut	M5 30 x 100	Robotdigg	1	kr 50,00	kr 50,00
Screw	CH-M5*8 x 50	Robotdigg	1	kr 6,60	kr 6,60

Aluminum profile connector	Alu2020CT	Robotdigg	16	kr 6,60	kr 105,60
Stepper motor Nema23	B23HD4404	Robotdigg	2	kr 218,20	kr 436,40
stepper motor bracket	SMMP-23	Robotdigg	2	kr 9,71	kr 19,42
Encoder	600 Incremental	Ebay	2	kr 92,62	kr 185,24
Timing-belt	5 m HTD5M 9mm	Robotdigg	1	kr 186,00	kr 186,00
Timing pulley	Idler9-16T-3B-S	Robotdigg	2	kr 15,00	kr 30,00
Toothed clamp	9mm wide	Robotdigg	2	kr 11,00	kr 22,00
Photoelectric IR beam sensor	5m sensing distance	Ebay	2	kr 80,00	kr 160,00
POM wheels	V-slot	Ebay	12	kr 10,00	kr 120,00
Drag chain	DC-40_Open	Robotdigg	1	kr 31,56	kr 31,56
Horizontal robot					
Stepper motor Nema 14	SMNEMA14-08	Robotdigg	2	kr 55,00	kr 110,00
DRV8825	Stepper motor driver	Robotdigg	3	kr 17,00	kr 51,00
KeyeStudio V4.0	Motor Driver Shield	Ebay	1	kr 66,00	kr 66,00
Timing belt	GT2 6mm	Ebay	1	kr 30,00	kr 30,00
Timing pulley bearing	16T 3mm thoothed	Robotdigg	1	kr 14,00	kr 14,00

Timing pulley	16T 5mm	Robotdigg	1	kr 24,00	kr 24,00
Timing pulley bearing	20T 4mm thoothed	Robotdigg	1	kr 14,00	kr 14,00
Timing pulley	20T 5mm	Robotdigg	1	kr 24,00	kr 24,00
Endstop	SS-5GL	Robotdigg	4	kr 4,50	kr 18,00
Linear Slider	MGN9	Robotdigg	1	kr 283,00	kr 283,00
Linear Slider	SS_MGN9_1H-350	Robotdigg	1	kr 161,00	kr 161,00
Screws hex socket	3*8 mm x 20	Robotdigg	3	kr 4,85	kr 14,55
Inner L-connector	I2020L	Robotdigg	4	kr 2,50	kr 10,00
Drag chain	DC-20_Open	Robotdigg	1	kr 31,56	kr 31,56
Pozidrive screw	3*6mm x 100	Robotdigg	1	kr 53,70	kr 53,70
T-nut 20	PostNut20-M5	Robotdigg	1	kr 50,00	kr 50,00
T-nut 20	PostNut20-M3	Robotdigg	1	kr 50,00	kr 50,00
Air tubing	7.5 inner mm, 10 outer	RS online	1	kr 444,00	kr 444,00
Legirs	R 3/8 Male 10mm	RS online	2	kr 38,00	kr 76,00
Legris	R 3/8 Male 12mm	RS online	2	kr 62,60	kr 125,20
Total			116		kr 10 330,80

D.2 URL for bill of material

Component	URL
Eletrical cabinet	
Odroid XU4	http://www.hardkernel.com/main/products/prdt_info.php
Arduino Mega	https://store.arduino.cc/usa/arduino-mega-2560-rev3
Arduino Nano	https://store.arduino.cc/usa/arduino-nano
DM556	https://www.robotdigg.com/product/1285/DM542,-DM556-stepper-motor-driver
PS5R-SG24	http://us.idec.com/Catalog/ProductDetails.aspx?ProductId=PS5R-SG24&FamilyName=Family&SeriesName=PS5R_Slim_Series
MDR-40-12	https://www.meanwell-web.com/en-gb/ac-dc-industrial-din-rail-power-supply-output-mdr--40--12
DR-15-5	https://www.meanwell-web.com/en-gb/ac-dc-industrial-din-rail-power-supply-output-5vdc-dr--15--5
A9F84203	https://octopart.com/a9f84203-schneider+electric-30380459
A9F06206	https://no.rs-online.com/web/p/products/7762115/?grossPrice=Y&cm_mmc=NO-PLA-DS3A-_-google-_-PLA_NO_NO_Automation_And_Control_Gear-_-Circuit_Protection_And_Circuit_Breakers%7CMcbs-_-PRODUCT+GROUP&matchtype=&gclid=CjwKCAjwxZnYBRAVEiwANMTRX1t9GEx2FBKDq30Ue14yzmvoIrDt2HMHHQikjrga-HGcdvUknwLnsRoCmx4QAvD_BwE&gclsrc=aw.ds
857-304	https://www.elfadistrelec.no/no/pluggsokkel-med-miniatyrkoblingsrele-wago-857-304/p/11051783?channel=b2c&price_gs=186.25&source=googleps&ext_cid=shgooaqnono-na&pup_e=1&pup_cid=35879&pup_id=11051783&ext_cid=shgooaqnono-na-NO|+Shopping+|+Manufacturer&kw=&gclid=CjwKCAjwxZnYBRAVEiwANMTRXxlrO3ZgWFSmBxZzJD9573QrNOAWAin97T6keDUtnymrXYNiXzEbsBoCaEoQAvD_BwE
Optocoupler 10A	https://www.robotdigg.com/product/1195/5V-Relay-Module-10A-optocoupler-isolated
Optocoupler 10A x4	https://www.ebay.com/itm/4-Channel-Relay-Module-Controller-5V-Arduino-Mega-2560-UNO-R3-Raspberry-Pi-/222244532164
Vertical robot	
Aluminium Alloy Strut	https://no.rs-online.com/web/p/tubing-struts/7613293/
L-connector	https://www.robotdigg.com/product/704/Inner-L-connector-for-2020-or-3030-Alu-Profile
Set Screw	https://www.robotdigg.com/product/91/M4*5-Set-Screw-Pack-Set-for-Timing-Pulleys
T nut	https://www.robotdigg.com/product/237/Post-assembly-T-nuts-for-1515,-2020-or-3030-aluminum-profile
Screw	https://www.robotdigg.com/product/706/M3-or-M5-Hex-Socket-Cap-Head-Screw
Aluminum profile connector	https://www.robotdigg.com/product/1002/2020-aluminum-profile-connector-n-cover
Stepper motor Nema23	https://www.robotdigg.com/product/558/Brake-stepper-motor-Nema17-or-Nema23
stepper motor bracket	https://www.robotdigg.com/product/240/Mounting-bracket-for-stepper-motors

Encoder	https://www.ebay.com/itm/360-600P-R-Photoelectric-Incremental-Rotary-Encoder-5V-24V-AB-Two-Phases-Shaft/401197069294?hash=item5d693573ee:m:m4Tfk3PwjPrkbCDcd9ylRQA
Timing-belt	https://www.robotdigg.com/product/598/9mm-or-12mm-wide-open-ended-5M-belt
Timing pulley	https://www.robotdigg.com/product/402/9mm-wide-belt-idler-wheel,-pulley-16-or-20-tooth
Toothed clamp	https://www.robotdigg.com/product/491/Toothed-clamp-for-open-ended-belt
Photoelectric IR beam sensor	https://www.ebay.com/itm/2pcs-5M-IR-Photoelectric-Sensor-Switch-Photoswitch-Through-Beam-PNP-DC-12V-24V/142399982113?hash=item2127b34a21:g:960AAOSwLF1X4LMC
POM wheels	https://www.ebay.com/itm/Plastic-Wheel-POM-w-Bearing-Passive-Round-Wheel-Gear-for-V-slot-3D-Printer-Lot/401468175015?_trkparms=aid%3D222007%26algo%3DSIM.MBE%26ao%3D2%26asc%3D44040%26meid%3D4afb6159485049adbe502f16a728b65d%26pid%3D100005%26rk%3D6%26rkt%3D6%26sd%3D272975187396&_trksid=p2047675.c100005.m1851
Drag chain	https://www.robotdigg.com/product/546/Drag-Chain-Inner-15*20mm-or-15*40mm
Horizontal robot	
Stepper motor Nema 14	https://www.robotdigg.com/product/28/NEMA14-34mm-0.8A-or-1.25A-stepper-motor
DRV8825	https://www.robotdigg.com/product/208/DRV8825-Stepper-Driver
KeyeStudio V4.0	https://www.ebay.com/itm/3D-Printer-Stepper-Motor-Driver-CNC-Shield-v4-0-Board-for-Arduino-Nano/332193623231?hash=item4d58486cbf:g:qGwAAOSwRUhY~een
Timing belt	https://www.ebay.com/itm/5m-GT2-Open-Rubber-Timing-Belt-2GT-6mm-Width-For-3D-Printer-CNC-Reprap-Prusa-i3/311964032721?hash=item48a2815ed1:g:p5wAAOSwwGdZwcKt
Timing pulley bearing	https://www.robotdigg.com/product/637/2GT-Idler-Pulley-w/-Bearings
Timing pulley	https://www.robotdigg.com/product/59/Rostock-16-Tooth-GT2-Pulley
Timing pulley bearing	https://www.robotdigg.com/product/637/2GT-Idler-Pulley-w/-Bearings
Timing pulley	https://www.robotdigg.com/product/9/GT2-Pulley-20-Tooth-4mm-or-5mm-Bore
Endstop	https://www.robotdigg.com/product/141/Endstop-Subminiature-Switch-SS-5GL
Linear Slider	https://www.robotdigg.com/product/347/Custom-length-GCr15-MGN9-or-MGN12-Linear-Rail-n-Carriage
Linear Slider	https://www.robotdigg.com/product/779/Quality-440C-SUS-MGN9-linear-rail-with-carriage
Screws hex socket	https://www.robotdigg.com/product/1278/M3-hex-socket-button-head-screw
Inner L-connector	https://www.robotdigg.com/product/704/Inner-L-connector-for-2020-or-3030-Alu-Profile

Drag chain	https://www.robotdigg.com/product/546/Drag-Chain-Inner-15*20mm-or-15*40mm
Pozidrive screw	https://uk.rs-online.com/web/p/machine-screws/0528946/?searchTerm=528-946&relevancy-data=636F3D3126696E3D4931384E525353746F636B4E756D626572266C753D656E266D6D3D6D61746368616C6C26706D3D5E285C647B362C377D5B4161426250705D297C285C647B337D5B5C732D2F255C2E2C5D5C647B332C347D5B4161426250705D3F292426706F3D3126736E3D592673743D52535F53544F434B5F4E554D4245522677633D4E4F4E4526 7573743D3532382D393436267374613D3035323839343626
T-nut 20	https://www.robotdigg.com/product/237/Post-assembly-T-nuts-for-1515,-2020-or-3030-aluminum-profile
T-nut 20	https://www.robotdigg.com/product/237/Post-assembly-T-nuts-for-1515,-2020-or-3030-aluminum-profile
Air tubing	https://no.rs-online.com/web/p/air-hose/7747078/
Legirs	https://no.rs-online.com/web/p/pneumatic-straight-threaded-to-tube-adaptors/2655630/
Legris	https://no.rs-online.com/web/p/pneumatic-straight-threaded-to-tube-adaptors/2655652/

Appendix E

Electrical drawings

Roe picking robot

Skoleversion

PCSCHEMATIC Automation

NTNU

Project title:	**Roe picking robot**	**Case no.:**	**Project rev.:**	**Page**	**1**
Customer:	NTNU Aalesund	DCC:		Scale:	1:1
Page title:	Frontpage	Dwg. no.:	Page rev.:	Previous page:	
File name:	Drawings	Eng. (proj/page):	Last print: 23.05.2018	Next page:	2
Page ref.:		Appr. (date/init):	Last edit: 23.05.2018	Total no. of pages:	16

Roe picking robot

Skoleversion

Diagrams	1
Lists	2
	3
	4
	5
	6
	7
	8
	9
	10

Title	Revision	Last corrected	Page no.
Diagrams			
Power		23.05.2018 13.38.20	5
Communication Controllers and pheripherals		30.04.2018 15.02.24	6
IR sensors and Endsensors Linear robot		23.05.2018 12.58.16	7
Arduino Mega schematic		23.05.2018 13.41.20	8
Emergency stop schematic		23.05.2018 13.33.12	9
Emergency stop schematic		23.05.2018 13.06.34	10
Emergency stop schematic		23.05.2018 13.20.44	11
Lists			
Components list		23.05.2018 13.42.54	12
Terminal list External / Internal		23.05.2018 13.41.44	15

Title	Revision	Last corrected	Page no.

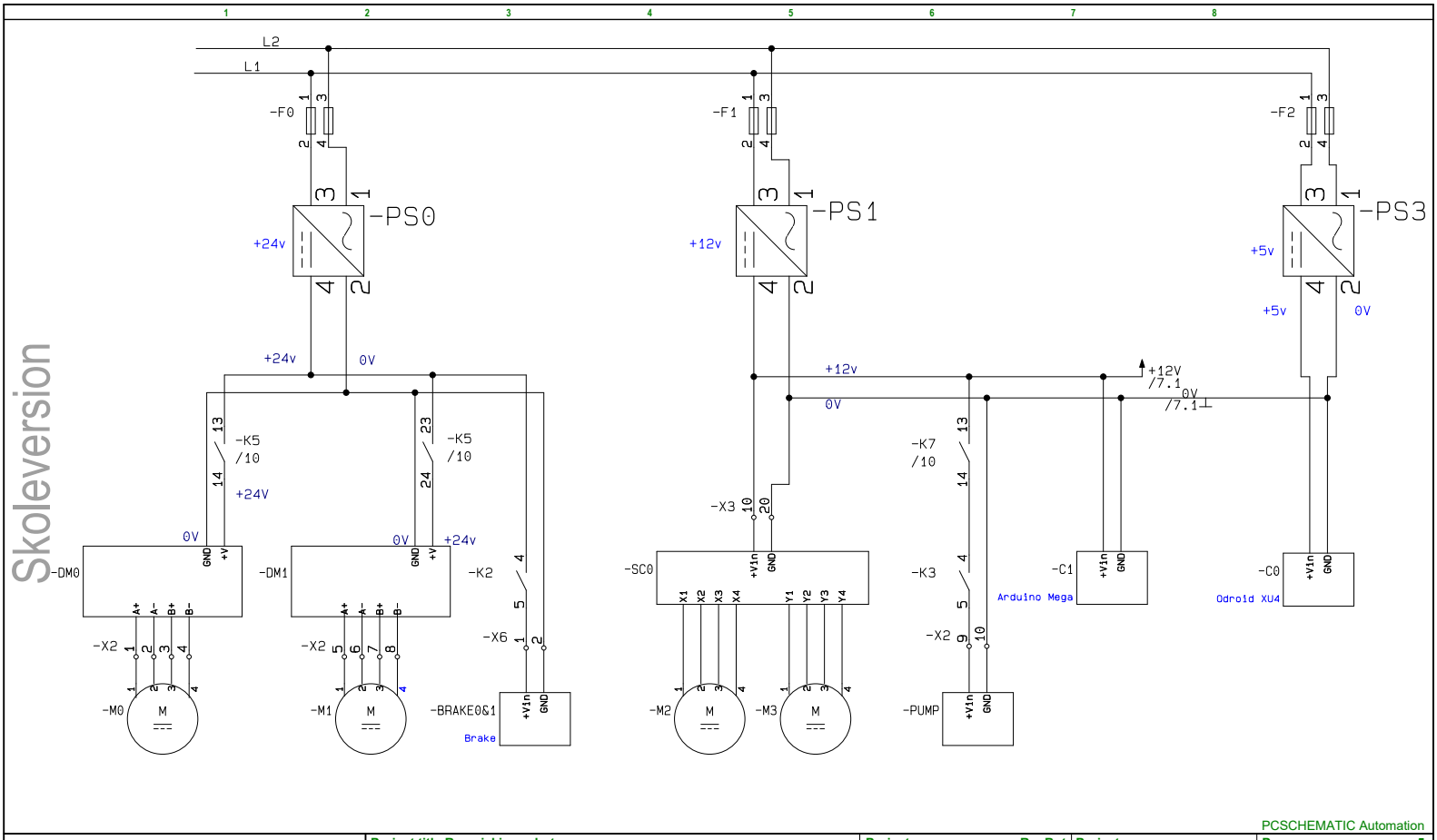
NTNU

Project title:Roe picking robot		**Project no.:** **RoeBot**	**Project rev.:**	**Page** **3**
Customer: NTNU Aalesund		DCC:		Scale: 1:1
Page title: Table of content		Drawing no.:	Page rev.:	Previous page: 2
Filename: Drawings		Constructor (project/page)	Last printed: 23.05.2018	Next page: 5
Page ref.:		Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

Skoleversion

Diagrams

Power Schematic



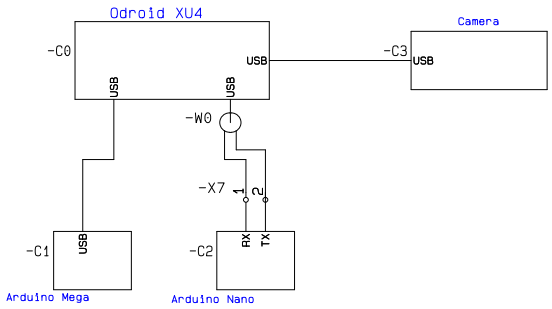
Skoleversion

PCSCHEMATIC Automation

NTNU

Project title:Roe picking robot	**Project no.:** RoeBot	**Project rev.:**	**Page**	5
Customer: NTNU Aalesund	DCC:		Scale:	1:1
Page title: Power	Drawing no.:	Page rev.:	Previous page:	3
Filename: Drawings	Constructor (project/page)	Last printed: 23.05.2018	Next page:	6
Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages:	16

Communication - Controllers Serial



Skoleversion

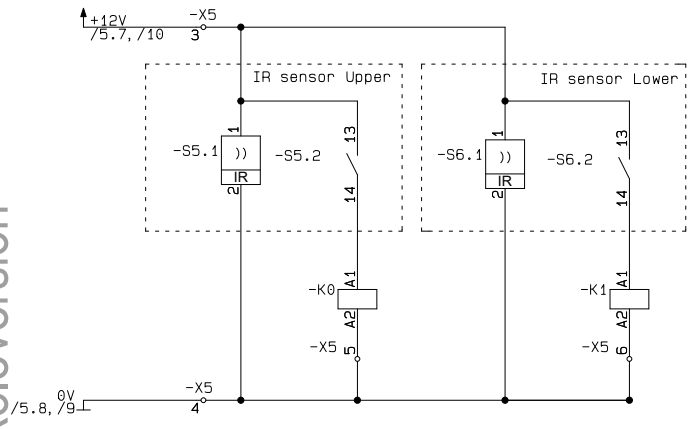
PCSCHEMATIC Automation

NTNU

Project title:	**Roe picking robot**	**Project no.:**	**RoeBot**	**Project rev.:**	**Page** 6
Customer:	NTNU Aalesund	DCC:			Scale: 1:1
Page title:	Communication Controllers and pheripherals	Drawing no.:	1	Page rev.:	Previous page: 5
Filename:	Drawings	Constructor (project/page)		Last printed: 23.05.2018	Next page: 7
Page ref.:		Appr. (date/sign.)		Last correction: 23.05.2018	Number of pages: 16

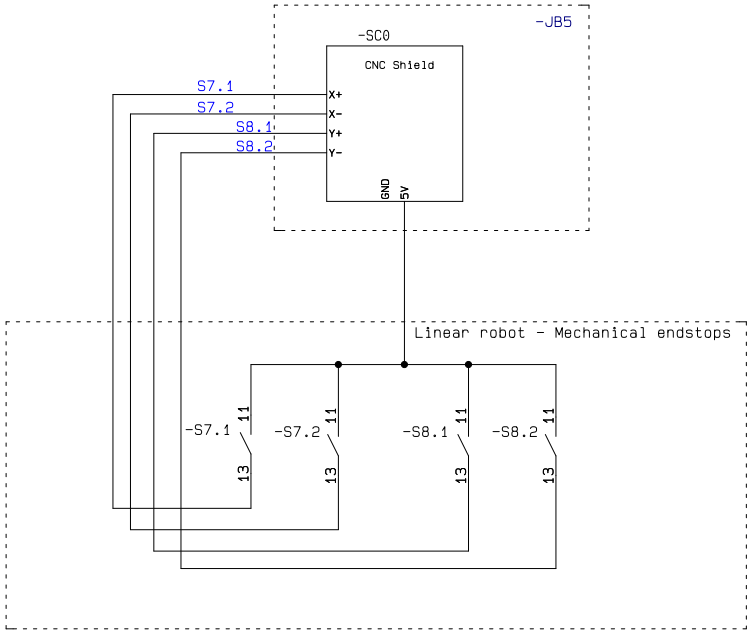
Safety IR Sensor

-Vertical robot



Endsensors to CNC Shield

-Horizontal robot

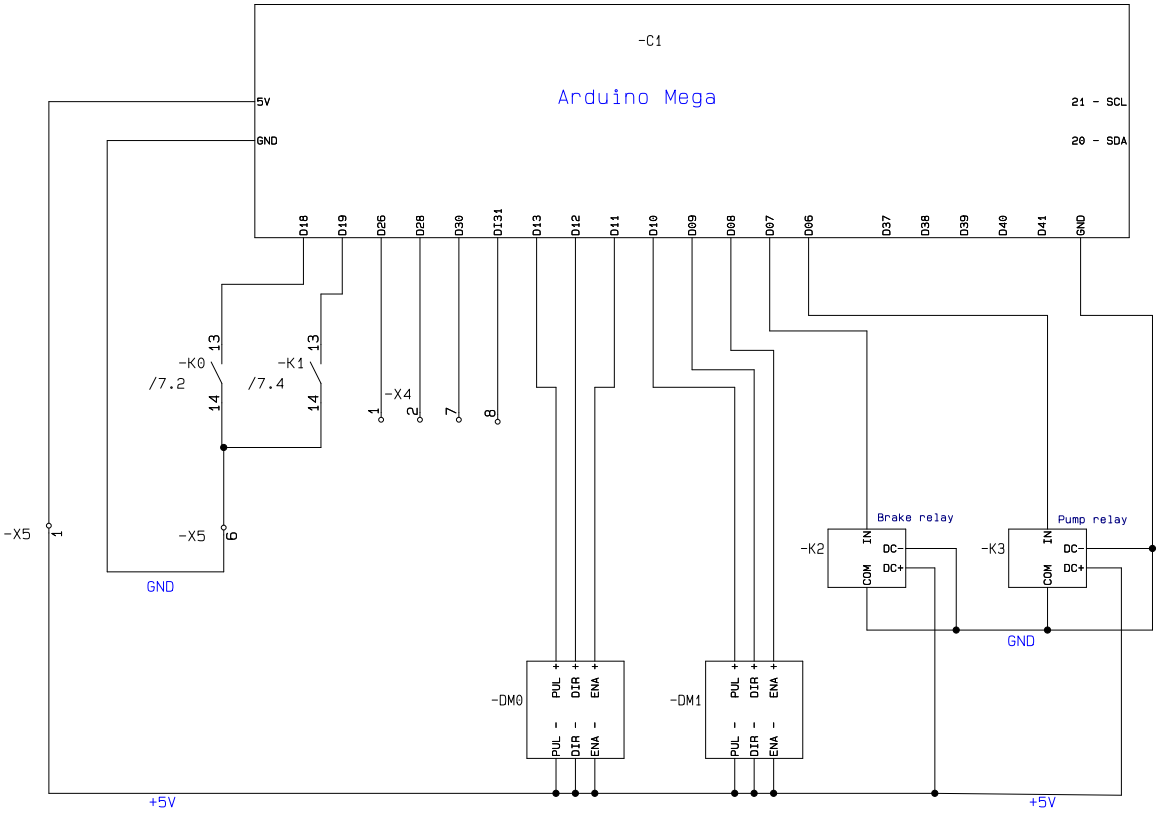


Skoleversion

PCSCHEMATIC Automation

NTNU	**Project title:** Roe picking robot		**Project no.:** RoeBot	**Project rev.:**	**Page** 7
	Customer: NTNU Aalesund		DCC:		Scale: 1:1
	Page title: IR sensors and Endsensors Linear robot		Drawing no.:	Page rev.:	Previous page: 6
	Filename: Drawings		Constructor (project/page)	Last printed: 23.05.2018	Next page: 8
	Page ref.:		Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

Vertical Controller - Signal

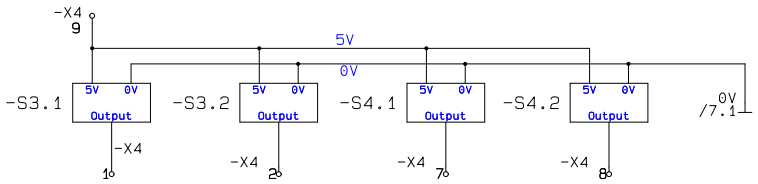


Skoleversion

PCSCHEMATIC Automation

Project title: Roe picking robot		**Project no.:** RoeBot	**Project rev.:**	**Page** 8
Customer: NTNU Aalesund		DCC:		Scale: 1:1
Page title: Vertical Controller - Signals		Drawing no.:	Page rev.:	Previous page: 7
Filename: Drawings		Constructor (project/page)	Last printed: 23.05.2018	Next page: 9
Page ref.:		Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

Optical endstops - Vertical robot



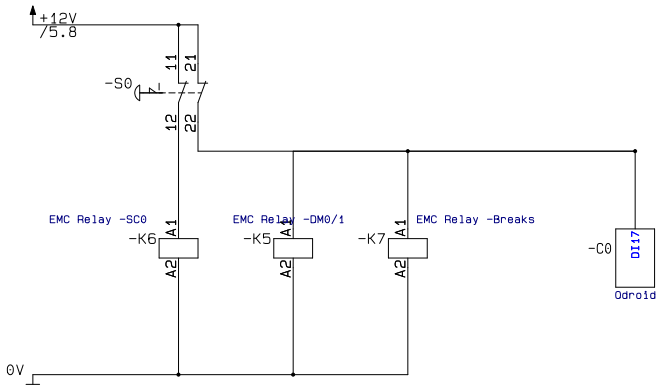
Skoleversion

PCSCHEMATIC Automation

NTNU	**Project title:** Roe picking robot	**Project no.:** RoeBot	**Project rev.:**	**Page** 9
	Customer: NTNU Aalesund	DCC:		Scale: 1:2
	Page title: Vertical Robot - Optical endstops	Drawing no.:	Page rev.:	Previous page: 8
	Filename: Drawings	Constructor (project/page)	Last printed: 23.05.2018	Next page: 10
	Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

Emergency stop

Skoleversion

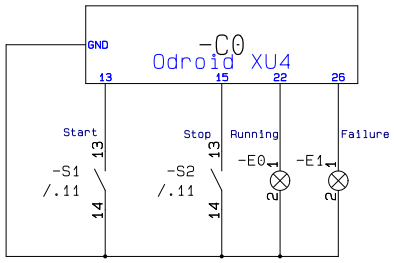


PCSCHEMATIC Automation

NTNU	**Project title:** Roe picking robot	**Project no.:** RoeBot	**Project rev.:**	**Page** 10
	Customer: NTNU Aalesund	DCC:		Scale: 1:2
	Page title: Emergency stop schematic	Drawing no.:	Page rev.:	Previous page: 9
	Filename: Drawings	Constructor (project/page)	Last printed: 23.05.2018	Next page: 11
	Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

HMI - Buttons and lights

Skoleversion



PCSCHEMATIC Automation

NTNU	**Project title:** Roe picking robot	**Project no.:** RoeBot	**Project rev.:**	**Page** 11
	Customer: NTNU Aalesund	DCC:		Scale: 1:2
	Page title: HMI - Buttons and lights	Drawing no.:	Page rev.:	Previous page: 10
	Filename: Drawings	Constructor (project/page)	Last printed: 23.05.2018	Next page: 12
	Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

Skoleversion

Lists

Line	Component	Part no.	Type	Manufacturer	Source	Description	Position
1	-BRAKE0&1		Brake for M0 and M1				/5.3
2	-C0		Odroid XU4				/5.8
3	-C1		Arduino Mega				/5.7
4	-C2		Arduino Nano				/6.4
5	-C3		Camera				/6.6
6	-DM0		Motor Driver DM556				/5.1
7	-DM1		Motor Driver DM556				/5.2
8	-E0		Ready				/11
9	-E1		Failure				/11
10	-F0		Fuse +24V				/5.2
11	-F1		Fuse +12V				/5.5
12	-F2		Fuse +5V				/5.8
13	-K0		+12 Relay				/7.2
14	-K1		+12 Relay				/7.4
15	-K2		Optocoupler Motor brake				/8.6
16	-K3		Optocoupler Pump				/8.7
17	-K5		Releè for supply to -DM0&1				/10
18	-K6		EMC Releè for supply to -SC0				/10
19	-K7		EMC Releè for -Break0&1				/10
20	-M0		Nema 23 - Z-axis				/5.1
21	-M1		Nema 23 - Z-axis				/5.2
22	-M2		Nema 17 - X-axis				/5.4
23	-M3		Nema 17 - Y-axis				/5.5
24	-PS0		230V to +24V Supply				/5.2
25	-PS1		230V to +12V supply				/5.5
26	-PS3		230V to +5V Supply				/5.8
27	-PUMP		Vacuum pump				/5.6
28	-S0		Emergency Stop				/10

Line	Component	Part no.	Type	Manufacturer	Source	Description	Position
29	-S1		Start button				/11
30	-S2		Stop button				/11
31	-S3.1		Optical sensor bottom				/9
32	-S3.2		Optical sensor bottom				/9
33	-S4.1		Optical sensor top				/9
34	-S4.2		Optical sensor top				/9
35	-S5.1		IR Light Upper				/7.1
36	-S5.2		IR sensor Upper				/7.2
37	-S6.1		IR Light Lower				/7.3
38	-S6.2		IR sensor Lower				/7.4
39	-S7.1		Mechanical endstop - X inner/left				/7.6
40	-S7.2		Mechanical endstop - X outer/right				/7.7
41	-S8.1		Mechanical endstop - Y inner				/7.7
42	-S8.2		Mechanical endstop - Y outer				/7.8
43	-SC0		Driver shield for Ard. Nano				/5.5
44	-W0						/6.4
45	-X2						/5.6
46	-X3						/5.5
47	-X4						/5.2
48	-X5						/7.1
49	-X6						/5.3
50	-X7						/6.4

NTNU

Project title:Roe picking robot	**Project no.: RoeBot**	**Project rev.:**	**Page 13**
Customer: NTNU Aalesund	DCC:		Scale: 1:1
Page title: Components list	Drawing no.:	Page rev.:	Previous page: 12
Filename: Drawings	Constructor (project/page)	Last printed: 23.05.2018	Next page: 15
Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2018	Number of pages: 16

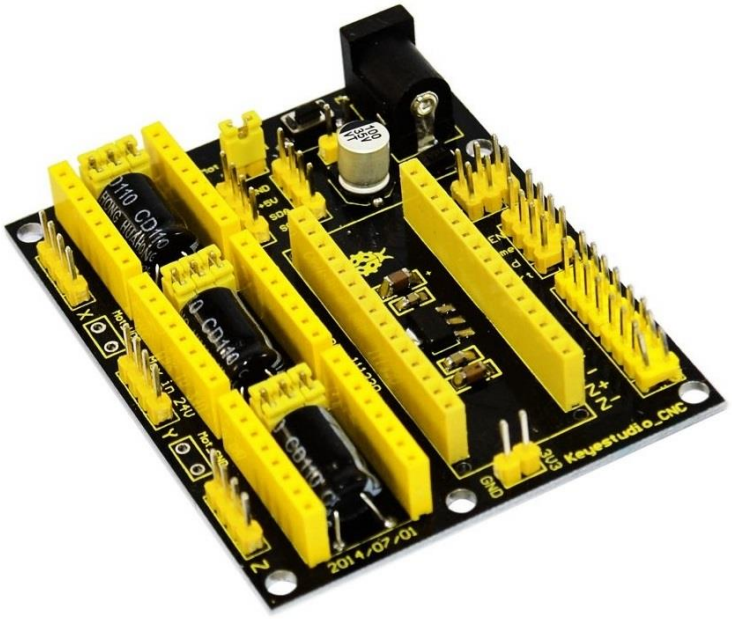
Line	From (external)	Cable (external)	Terminal	To (internal)	Wire no. (internal)	Type	Position
1	-M0 1		**-X2** 1	-DM0 A+			/5.1
2	-M0 2		**-X2** 2	-DM0 A-			/5.1
3	-M0 3		**-X2** 3	-DM0 B+			/5.1
4	-M0 4		**-X2** 4	-DM0 B-			/5.1
5	-M1 1		**-X2** 5				/5.2
6	-PUMP +Vin		**-X2** 9	-K3 5			/5.6
7	-PUMP GND		**-X2** 10	0V			/5.6
8							
9	-SC0 +Vin		**-X3** 10	+12V			/5.5
10	-SC0 GND		**-X3** 20	0V			/5.5
11							
12			**-X4** 1	-C1 D26			/8.3
13			**-X4** 2	-C1 D28			/8.3
14	-M1 2		**-X4** 6				/5.2
15	-M1 3		**-X4** 7				/5.2
16			**-X4** 7	-C1 D30			/8.3
17	-M1 4		**-X4** 8				/5.2
18			**-X4** 8	-C1 DI31			/8.3
19							
20	-C1 5V		**-X5** 1	-DM0 PUL -			/8.1
21	-S6.1 1		**-X5** 3				/7.1
22	-S5.1 1		**-X5** 3	+12V			/7.1
23	-S5.1 2		**-X5** 4	0V			/7.1
24	-S6.1 2		**-X5** 5				/7.2
25	-S5.1 2		**-X5** 5	-K0 A2			/7.2
26	-K0 14		**-X5** 6	-C1 GND			/8.1
27	-S6.1 2		**-X5** 6	-K1 A2			/7.4
28							

Line	From (external)	Cable (external)	Terminal	To (internal)	Wire no. (internal)	Type	Position
29	-BRAKE0&1 +Vin		**-X6** 1	-K2 5			/5.3
30	-BRAKE0&1 GND		**-X6** 2	-DM1 GND			/5.3
31							
32	-C2 RX		**-X7** 1				/6.4
33			**-X7** 2	-C2 TX			/6.4
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							
49							
50							
51							
52							
53							
54							
55							
56							

Appendix F

Keye Studio V4.0 I/O

Keyestudio CNC shield v4.0



I/O On nano	Tag on shield	Function
D13	D13	
D12	D12	
D11	Z- Z+ (D11)	PWM
D10	Y- Y+ (D10)	PWM
D9	X- X+ (D09)	PWM
D8	ENABLE X Y Z	
D7	DIR Z	
D6	DIR Y	PWM
D5	DIR X	PWM
D4	Step Z	
D3	Step Y	PWM
D2	Step X	
A7	A7	Analog In
A6	A6	Analog In
A5	SCL	Analog In
A4	SDA	Analog In
A3	CoolEN	Analog In
A2	Resume	Analog In
A1	Hold	Analog In
A0	Abort	Analog In
RX	RX	
TX	TX	

Appendix G

Timesheet

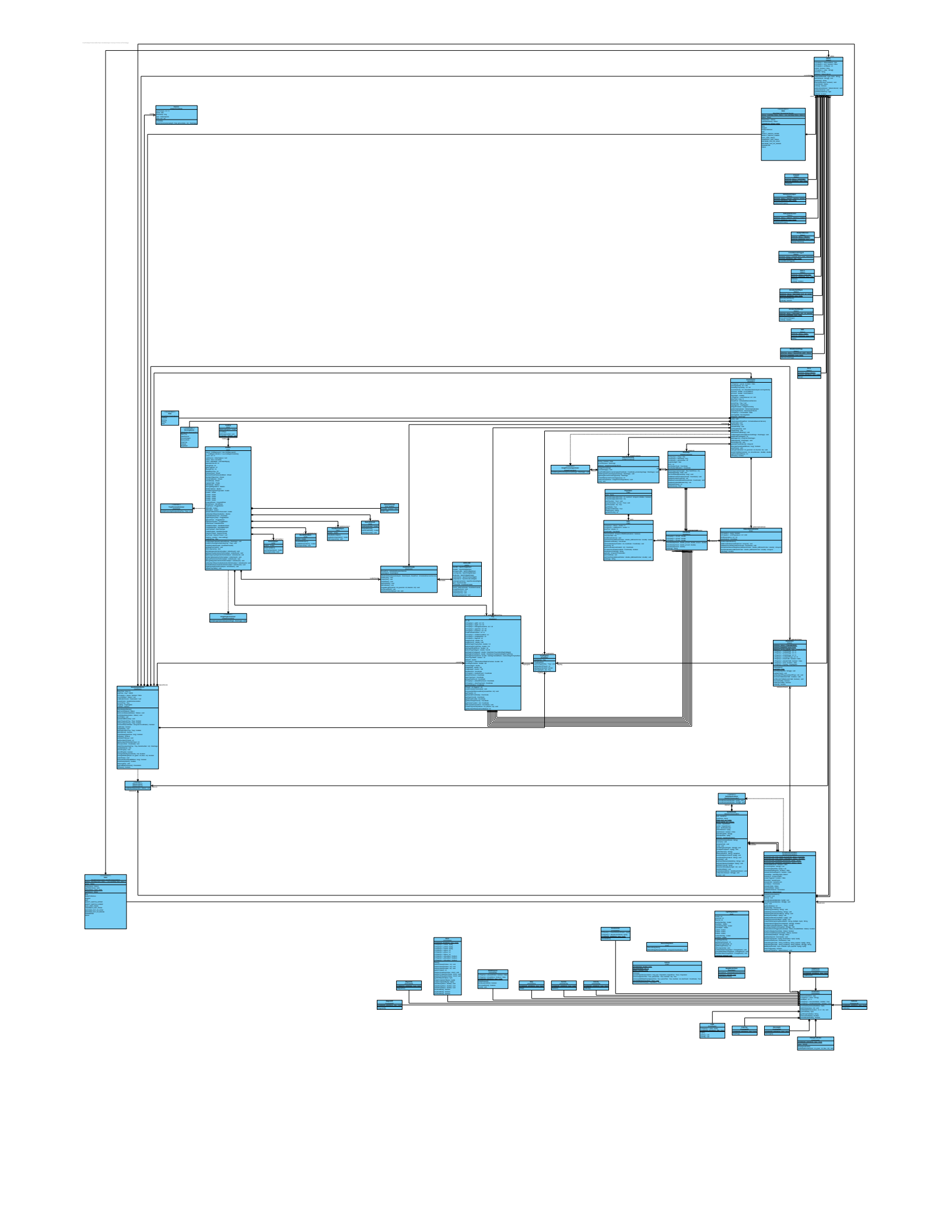
Timeliste Kristoffer			Timeliste Kristian			Timeliste Yngve			Timeliste Per Espen		
Dato:	Antall timer:	Kommentar:	Antall timer:	Kommentar		Antall timer:	Kommentar:		Antall timer:	Kommentar:	
8.01.18	7	Forprosjekt	8,50	Forprosjekt		7,5			7	Forprosjekt	
9.01.18	7,5	Forprosjekt	8,5	Forprosjekt		8			7,5	Forprosjekt	
15.01.18	7	Forprosjekt	8,5	Forprosjekt		7			7	Møte/forprosjekt	
16.01.18	7	Forprosjekt	8,5	Forprosjekt		7,5			7,5		
17.01.18	2	Forprosjekt	7,5	Forprosjekt		8			7,5		
18.01.18	7,5	Forprosjekt	8	Forprosjekt, finne deler		8			7		
22.01.2018	7	Utflukt	8,5	Marine Harvest ytre Standal, finne deler		8	Utflukt hos Marine Harvest ytrestandal		7	Marine Harvest ytrestandal	
23.01.2018	7,5	finne deler, tegne main constructio	8,5	Finne deler, tegne main construction		9,5	Finne deler, tegne liear robot		7,5	Finne deler, tegnet	
24.01.2018	7,5	Tegning	8,5	Tegne main construction, skjære ut traller		10	Tegning av lineær robot		8		
25.01.2018	7,5	Rapportskriving		ikke tilstede		3	Tegning av lineær robot		1	Tegnet	
26.01.2018	7,5			ikke tilstede					3		
29.01.2018	7,5	Bestille deler	9	lage bestillingsliste		8			8		
30.01.2018	7,5	Rapportskriving		syk		8	Rapport		8		
31.01.2018	0	Syk		syk		9,5	Vakuum design. Robot design.		7		
01.02.2018	7,5		8,5	rapportskriving		8			6,5		
02.02.2018	7,5	Rapportskriving	12	tegning av heismekanisme		7,5	Venturi Generator		7		
05.02.2018	8	Møte, forprosjektrapport	9,5	møte med veileder, heis med gjenger, forpro		10			8		
06.02.2018	7,5		9,5	research threaded rod vs timing belt, tegne v		8			9		
07.02.2018	7,5	Bestilling av deler	9	rapport, tegne verktøy		9			8		
08.02.2018	9	Finne info om atepper motor	11	tegne endebrytere, lodde endebrytere, heis		5			8,5		
09.02.2018	7,5		10	ferdigstille heis, lage pulleyu fester		10	Ferdistilling av robot tegninger og reserch om ki		9		
10.02.2018 lørdag	0		8	printe deler, skrive rapport					0		
12.02.2018	9		12	printe deler, skrive rapport, kappe metall, set		10			9,5		
13.02.2018	12		14	bestille skruer, hente kommode, planlegge pr		8,5			12		
14.02.2018	8		12	tegne, printe, programmere, diskutere		7			7		
15.02.2018	8		11	diskutere kommunikasjon, print, BB		10			8		
16.02.2018	7		14,5	diskutere, printe, BB		13			8		
17.02.2018 lørdag	0		6	bygge main construction :D		5			1		
19.02.2018	6		10			9	Program struktur		7,5	Program struktur	
20.02.2018	9		11	BB		10	Program metoder		9	Programmere genrell I2CCommunication	
21.02.2018	9		10	BB		9	Program metoder		7,5	Sette opp generell program struktur	
22.02.2018	8		5	BB		9	Program GA pattern		8	programmere videre på I2C communication	
23.02.2018	8		12	BB, rapport			15 kilometer klassisk Råkollen		8	Videreutvikle I2CCommunication	
24.02.2018 lørdag		søv	2	Rapport			10 kilometer klassisk				
26.02.2018	8,5		11	Rapport, møte		9			7,5	Rapport	
27.02.2018	10		9	BB, rapport		9			6,5 + 1,5	Rapport	
28.02.2018	0		11	Bygge El-skap + div		8			10		
01.03.2018	0		11	koble El-skap, skjære hjørner		9			8		
02.03.2018	0		9	Rapport, stramme ramme		9			6,5	Rapport :D	
03.03.2018 Lørdag	5	kinematikk							0		
04.03.2018 Søndag	5	Rapportskriving							0		
05.03.2018	11	kinematikk	11	programmere GUI, lage kalibreringsmetode x		10			9	Utfylling og implementering av RoeDevice	
06.03.2018	8	kinematikk	13	GUI, montasje, Arduino, test kalibrer		8			9,5	testing og feilsøking av i2c dritt	
07.03.2018	7,5	Bresenham	7,5	GUI		8			6+1,5		
08.03.2018	8	kinematikk	10	Bygge linearbot		8			7,5		
09.03.2018		Jobbintervju	13	Bygge bygge :		8			8		
10.03.2018 Lørdag									0		
11.03.2018									0		
12.03.2018		Jobbintervju	10	Rapport		9			8		
13.03.2018	7,5	Rapportskriving	9	Review		9			8		
14.03.2018				industri 4.0					2		
15.03.2018			2	industri 4.0, airhockey lokk		1	Printing av deler		0		
16.03.2018				industri 4.0		1	Printing av deler		0		
17.03.2018 Lørdag									0		
18.03.2018 Søndag									0		
19.03.2018	9	Rapportskriving	10	rapport		9	Montere og redesigne		8		
20.03.2018	8	Bolleeting og arduino	9	koble elskap		11	Montere og redesigne		8		
21.03.2018	7,5	Rapportskriving	11	montere ting og tang		10	Montere og redesigne		8		
22.03.2018	8	Test	9	funksjonstesting av heis		9	Montere og redesigne		9		
23.03.2018	7	Rapportskriving	2	Kvistian var på effesE kurs, rapport		7	Montere og redesigne		4		
24.03.2018 lørdag									0		
25.03.2018 Søndag									0		
26.03.2018	7,5		8	Rapport		9	Rapport skriving		8		
27.03.2018	7,5		13	Rapport		6	Rapport skriving		7,5		
28.03.2018	4		3	Rapport		8	Rapport skriving				
29.03.2018 påske						1,00	Rapport lesing				
30.03.2018 påske											
31.03.2018 påske			2	Lese rapport		1,00	Rapport lesing				
01.04.2018 påske			1	lese rapport							
02.04.2018 påsk	2		1	lese rapport		1,00	Rapport lesing				
03.04.2018	8		7,5	fikse kommode osv.		13,00	Kobling og montering		10	Arduino programmering, testing, montering	
04.04.2018	7		14	koble og montere					10	Testing av ard og koblinger.	
05.04.2018	7		8	programmere listener							
06.04.2018	8		10	rapport							
07.04.2018											
08.04.2018											
09.04.2018											
10.04.2018									2	i2c testing, fant feil	
11.04.2018									1,5	i2c testing	
12.04.2018									2,5	i2c party	
13.04.2018									3,5	i2c dritt	
14.04.2018	5		3	rapport					2,5	i2c sjukt fest	
15.04.2018											
16.04.2018	12	GUI	12	Rapport, testing i2c		12	I2c testing.		10		
17.04.2018	11	GUI	13	bildebehandling, testing		13	Serial coding		11,5	Kika på serial	
18.04.2018	11		11	Bildebehandling		11	Serial coding		10,5	Koda ny serial klasse	
19.04.2018	9		11			11			11	Håndtering av serial reader writers	
20.04.2018	8		10			10			10		
21.04.2018 Lørd	6								2,5		
22.04.2018 Søndag											
23.04.2018	10	GUI	13	Bildebehandling rapport		12			12		
24.04.2018	11	GUI	11	bildebehandling, testing		10			10,5		
25.04.2018	8		8	Rapport		8			7		
26.04.2018 syk			10	Rapport		10			10,5		
27.04.2018 syk			9	rapport		12			12		
28.04.2018 Lørdag						4			4		
29.04.2018 sønc	4	Rapportskriving									
30.04.2018	11	Rapportskriving	9	Rapport		9	Rapport		7,5	Rapport	
01.05.2018	11	Rapportskriving	10	Rapport		8	Rapport		8,5	Rapport, og knakk nøke på roboten	
02.05.2018	7	Rapportskriving	11	Rapport, testing		12	Rapport, testing		10	Rapport, testing	
03.05.2018	8	Rapportskriving	10	Rapport, testing		12	Rapport, testing		12	Rapport, testing	
04.05.2018	6		10	Rapport, testing					11		
05.05.2018 Lørd	4								4		
06.05.2018 søndag											
07.05.2018	3		9	Rapport, testing		10			10		
08.05.2018	8		10	Rapport, testing		10			10		
09.05.2018	0		11	Rapport, testing		11			11		
10.05.2018	10		10	Rapport		12			8		
11.05.2018	8	rapport	10	Rapport, testing		11			11		
12.05.2018	9	rapport	3	Testing		0			4		
13.05.2018 Sønc	9	rapport	6	rapport		12					
14.05.2018	14	rapport	11	Rapport		14			11		
15.05.2018	10	rapport	9	Rapport		16			10		
16.05.2018	10	rapport	9	Rapport		10			11		
17.05.2018									9		
18.05.2018	10	rapport	4	Rapport		7	Rapport		12		

19.05.2018	Lørd	8	rapport	10	Rapport	9	Rapport	7
20.05.2018	Sønc	6	rapport	5	Lese rapport	9		1
21.05.2018		12		13	Rapport	13		12
22.05.2018		13		13	Rapport	13		11
23.05.2018		12		12	Rapport	13		11
24.05.2018		13		13	Rapport	13		12
25.05.2018		10		10	Rapport	5		9
26.05.2018	lørd	7		7	Rapport	8		7
27.05.2018	sønc	11		11	Rapport	11		7
28.05.2018		14		14	Rapport	15,50		14
29.05.2018		13		13	Rapport	14,5		12
30.05.2018		13		13	Rapport	14,5		12
31.05.2018		3		7	Rapport	7		2
01.06.2018	leveringsfrist							
total		**757,5**		**910,00**		**809,00**		**791,50**
Totalt antall timer			**3268,00**					

Appendix H

Java Code

H.1 Java Class Diagram



H.2 Roe Robot Main

```java
package RoeRobot;

import GUI.RoeBot;
import com.pi4j.platform.PlatformAlreadyAssignedException;
import java.util.concurrent.Executors;
import GPIO.GPIO_HMI;
import RoeRobot.RoeAnalyser;
import RoeRobot.RoeRobotFasade;

import java.util.concurrent.ScheduledExecutorService;




/**
 *  THE MAIN CLASS
 *  This class creates the thread. Has control of all the running threads
 *  And creates the objects in the order they need to be
 * @author Yngve
 */
public class MegaMasterClass {

    /**
     * @param args the command line arguments
     * @throws com.pi4j.platform.PlatformAlreadyAssignedException
     */
    public static void main(String[] args) throws PlatformAlreadyAssignedException {

        //Load the open cv

        System.load("/home/odroid/NetBeansProjects/RoeRobotV3-All/RoeRobotV4/lib/opencv-package-xu4/libopencv_java310.so");
       // System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        new MegaMasterClass();
    }



    // Thread pool for keeping track of threads.
    private ScheduledExecutorService threadPool;

    public MegaMasterClass() throws PlatformAlreadyAssignedException
    {
        //
        this.threadPool = Executors.newScheduledThreadPool(10);

        RoeAnalyser roeAnalyser = new RoeAnalyser(this.threadPool);
        RoeRobotFasade roeRobotFasade = new RoeRobotFasade(roeAnalyser, this.threadPool);
         GPIO_HMI gpioHMI = new GPIO_HMI(roeRobotFasade);

        //START the GUI
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new RoeBot(roeRobotFasade).setVisible(true);
            }
        });

    }

}
```

```java
package RoeRobot;

import java.util.concurrent.ScheduledExecutorService;

/**
 *
 * @author Yngve
 */
public class RoeRobotFasade
{

    // Roe analyser.
    private final RoeAnalyser roeAnalyser;

    // Threadpoool for running roe analyser
    private final ScheduledExecutorService threadPool;

    /**
     * Constructor. Create the RoeAnalyser.
     * @param roeAnalyser
     * @param threadPool
     */
    public RoeRobotFasade(RoeAnalyser roeAnalyser, ScheduledExecutorService
    threadPool)
    {
        this.roeAnalyser = roeAnalyser;
        this.threadPool = threadPool;
        //this.startCycle();
    }

    /**
     * Start the dead roe detecting cycle.
     */
    public void startCycle()
    {
        // Set the state of the robot as start
        this.roeAnalyser.startRobot();
        // start the robot
        this.threadPool.execute(roeAnalyser);

    }

    /**
     * Stop the system.
     *
     */
    public void stopCycle()
    {
        // instant stop the threads from running
        this.threadPool.shutdownNow();
    }

    /**
     * this will stop the roebot
     */
    public void stopRobot()
    {
        roeAnalyser.stopRobot();
    }

    /**
     * this will pause the roebot
     */
    public void pauseRobot()
    {
        this.roeAnalyser.pauseRobot();
    }

    /**
     * Perform calibration of the robot
     */
    public void doCalibrate()
    {
```

```
73              this.roeAnalyser.startRobotCalibrating();
74               threadPool.execute(roeAnalyser);
75          }
76
77          /**
78           * Change the level of the lights
79           *
80           * @param redVal RED value
81           * @param greenVal GREEN value
82           * @param blueVal BLUE value
83           */
84          public void regulateLights(int redVal, int greenVal, int blueVal)
85          {
86              this.roeAnalyser.setLightVal(redVal, greenVal, blueVal);
87          }
88
89          /**
90           *
91           */
92          public void continueRobot()
93          {
94             this.roeAnalyser.unPauseRobot();
95          }
96
97          /**
98           * set the search interval
99           * @param input interval in minutes
100          */
101         public void setSearchInterval(int input)
102         {
103             this.roeAnalyser.setSearchInterval(input);
104         }
105 }
106
```

```java
package RoeRobot;

import ImageProcessing.ImageProcessing;
import ImageProcessing.ImageProcessingListener;
import ImageProcessing.RoeImage;
import java.util.ArrayList;
import java.util.concurrent.ScheduledExecutorService;
import TSP.PatternOptimalization;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * The main switch case Roe Analyser
 *
 * @author Yngve
 */
public class RoeAnalyser implements ImageProcessingListener, Runnable {

    @Override
    public void run() {
         while(running)
        {
        cycleCase();
        }
    }



    //State enum for the switchcase
    private enum State {
        Calibrate,
        Running,
        Waiting,
        Done,
        Fault;
    }

    //State enum for the running switchcase
    private enum RunningStates {
        OpenTray,
        TakePictures,
        ProcessImages,
        RemoveRoes,
        CloseTray,
        Finished,
        StopRobot;

    }

    //Pause boolean
    private boolean pause = false;

         //Pause boolean
    private boolean running = true;

    // Velocity for running
    private int runningVelocity = 150; // rev/min
    // Velocity while handeling tray
    private int handelingTrayVelicity = 60; // rev/min
    // Current Velocity
    private int currentVelocity = runningVelocity;

    //Pulley circumference for X and Y axiz
    // Diameter * pi
    private double xCircumf = 12.22*Math.PI;
    private double yCircumf = 9.678*Math.PI;


    //Flag to remember if the tray is open or not
    private boolean trayIsOpen;

    //Search interval in minutes
```

```
74      private int searchInterval = 100;
75      private long timerTime = 0;
76
77      //Current working tray
78      private Tray currentTray = null;
79      //
80      int takePictureNr = 0;
81        //Number of the next tray
82      int trayNumber = 1;
83
84      // Tray register
85      private TrayRegister trayRegister;
86
87      // Immage prosseser
88      private ImageProcessing imageProsseser;
89      // Thread pool for keeping track of threads.
90      private ScheduledExecutorService threadPool;
91
92      // Patterns optimalizer
93      private PatternOptimalization patternOptimalizater;
94
95      private RoeAnalyserDevice roeAnalyserDevice;
96      //State enum
97      private State currentState;
98      private RunningStates runningState;
99
100     // Roe image containing all dead roa coodrinates.
101     private ArrayList<RoeImage> imageList;
102
103     public RoeAnalyser(ScheduledExecutorService threadPool)
104     {
105         this.threadPool = threadPool;
106         this.roeAnalyserDevice = new RoeAnalyserDevice();
107
108         //Create image processor and add listener
109         this.imageProsseser = new ImageProcessing();
110         this.imageProsseser.addListener(this);
111         this.threadPool.execute(imageProsseser);
112
113         this.patternOptimalizater = new PatternOptimalization();
114
115         this.imageList = new ArrayList<>();
116         this.trayIsOpen = false;
117
118         //Set the pause to fault
119         this.pause = false;
120     }
121
122
123     private void cycleCase() {
124
125
126                 // System.out.println("currentState " + currentState);
127
128
129         switch (currentState) {
130             // CALIBRATE
131             // Sends calibatrion cmd.
132             case Calibrate:
133                 // Call on calibrate method in roeAnalyser
134                 // Call on nrOfTrays from raoAnalyser.
135                 //this.roeAnalyserDevice.changeVelocity(this.runningVelocity);
136                // Starts the calibration cycle
137                 this.roeAnalyserDevice.calibrate();
138                 this.trayRegister =
                    this.roeAnalyserDevice.getCalibrationParams().getTrayReg();
139                 currentState = State.Done;
140                 break;
141
142
143             // RUNNING
144             case Running:
145
```

```
146                 //Check if the tray is already open, and there are more trays to
                    handle
147                 if(!this.trayIsOpen && trayNumber <= trayRegister.getNumberOfTrays())
148                 {
149                     //Set the running state to open tray
150                 runningState = RunningStates.OpenTray;
151                  }
152
153
154
155                     //If pause is set
156                 if(!this.pause)
157                 {
158                     //The different sates in running
159                     switch (runningState)
160                     {
161
162                         //Open the tray
163                         case OpenTray:
164                             if (trayNumber <= trayRegister.getNumberOfTrays())
165                             {
166
167                                //Get the tray
168                                 this.currentTray =
                                    this.trayRegister.getTray(trayNumber);
169
170                                 //Open the current tray
171                                 if
                                    (this.roeAnalyserDevice.openTray(this.currentTray))
172                                 {
173                                     // Turn on ligth
174                                     this.roeAnalyserDevice.changeRGBLight(4, 4, 4);
175                                     try {
176                                         Thread.sleep(5);
177                                     } catch (InterruptedException ex) {
178
                                         Logger.getLogger(RoeAnalyser.class.getName()).
                                         log(Level.SEVERE, null, ex);
179                                     }
180
181                                     //Set variables for a open tray
182                                     this.trayIsOpen = true;
183                                     takePictureNr = 0; //Set picture nr to 0
184                                     runningState = RunningStates.TakePictures;
185
186                                 } else
187                                 {
188                                     //Failure happened, set the main state to failure
189                                     this.setCurrentState(State.Fault);
190                                 }
191                             }
192                             else
193                             {
194                                 //No more trays, set to finished
195                                 runningState = RunningStates.Finished;
196
197                             }
198                             break;
199
200                             // Take picture
201                         case TakePictures:
202                              System.out.println("Taking picture " + takePictureNr);
203
204                              if(takePictureNr <
                                 this.currentTray.getNumberOfCameraCoordinates())
205                                  {
206                                      RoeImage currentImage =
                                         this.roeAnalyserDevice.takePicture(this.currentTr
                                         ay, takePictureNr);
207                                      takePictureNr++;
208
                                     this.imageProsseser.addImageToProcessingQueue(curr
                                     entImage);
```

```
209                         }
210                      else
211                         {
212
213                             runningState = RunningStates.ProcessImages;
214                         }
215                     break;
216
217                 case ProcessImages:
218                     //Wait for all the images to get processed
219                     if(this.getNumberOfImages() ==
                        this.currentTray.getNumberOfCameraCoordinates())
220                     {
221                         //Generate the list of coordinates from the
                            processed images
222                         ArrayList<Coordinate> deadRoeList =
                            this.generateCoordinatList();
223
224                         //Optimize the pattern
225                         // Add all dead roe coodinates to the optimalisation
226                         this.patternOptimalizater.addCoordinates(deadRoeList);
227                         runningState = RunningStates.RemoveRoes;
228                     }
229                    // System.out.println("Waiting process");
230                     break;
231
232                     //Remove the dead roe
233                 case RemoveRoes:
234
235                        // test for reducing nr of points
236                         ArrayList<Coordinate> newArray = new ArrayList();
237                         // Covert from rev/min to mm/sec
238                         double xMMPerSec =
                            this.revMinToMMSec(this.currentVelocity,
                            this.xCircumf);
239                         double yMMPerSec =
                            this.revMinToMMSec(this.currentVelocity,
                            this.yCircumf);
240                         //Do the optimalisation
241                         newArray =
                            this.patternOptimalizater.doOptimalization(xMMPerSec,y
                            MMPerSec);
242                         //Remove the roe
243
                            this.roeAnalyserDevice.removeRoe(newArray);//this.patt
                            ernOptimalizater.doOptimalization());
244                         //Close the tray, after removal is done
245                         runningState = RunningStates.CloseTray;
246                     break;
247
248                     //Close the tray
249                 case CloseTray:
250                       // Close the tray.
251                         if
                            (this.roeAnalyserDevice.closeTray(this.currentTray))
252                         {
253                                             //Turn off lights
254                             this.roeAnalyserDevice.changeRGBLight(0, 0, 0);
255                             this.trayIsOpen = false;
256                             this.currentTray = null;
257                             trayNumber++;
258                              runningState = RunningStates.OpenTray;
259                         } else {
260                             this.setCurrentState(State.Fault);
261                         }
262                     break;
263
264                     //The robot is finished
265                 case Finished:
266                 //Reset the timer for search interval
267                     this.resetTimer();
268                     setCurrentState(State.Waiting);
269                     break;
```

```
270
271                             //Stop the robot
272                              case StopRobot:
273                                     //Check if tray is open and close it
274                                 if(this.trayIsOpen)
275                                 {
276
                                        this.roeAnalyserDevice.closeTray(this.currentTray)
                                        ;
277                                 }
278                                 //Set to done, as no tasks are to be performed
279                                 currentState = State.Done;
280                             break;
281
282
283                              default:
284                                  break;
285
286                             }
287
288                     }
289
290
291
292                 break;
293                 //Wait for the next searching interval
294             case Waiting:
295             //Wait until search interval timer has passed
296                 if(timerHasPassed(this.searchInterval))
297                     {
298                 //Set tray number to start
299                         trayNumber = 1;
300                         //Initiate the search
301                         this.setCurrentState(State.Running);
302                     }
303                 break;
304
305             case Fault:
306                 //Faulty status. Set to done, return messages
307                 running = false;
308                 break;
309
310                   case Done:
311                 //Done.. Just wait..
312                 break;
313
314             default:
315                 break;
316         }
317
318
319     }
320
321     /**
322      * Start the robot
323      */
324     public void startRobot() {
325         setCurrentState(State.Running);
326     }
327
328     public void pauseRobot() {
329         this.roeAnalyserDevice.setPause(true);
330         this.pause = true;
331     }
332     /**
333      * Unpause the robot
334      */
335     public void unPauseRobot() {
336         this.pause = false;
337         this.roeAnalyserDevice.setPause(false);
338
339     }
340
```

```
341         public void stopRobot() {
342             this.runningState = RunningStates.StopRobot;
343         }
344 
345         /**
346          * Start the robot
347          */
348         public void startRobotCalibrating() {
349             setCurrentState(State.Calibrate);
350         }
351 
352         @Override
353         public void notifyImageProcessed(RoeImage processedImage) {
354             this.addImage(processedImage);
355         }
356 
357         /**
358          * Get number of images in the list.
359          *
360          * @return number of images inlist.
361          */
362         private synchronized int getNumberOfImages() {
363             return this.imageList.size();
364         }
365 
366         /**
367          * Get list of proccesed images
368          *
369          * @return list of proccesed images
370          */
371         private synchronized ArrayList<RoeImage> getImageList() {
372             return imageList;
373         }
374 
375         /**
376          * Adds a image to the image list.
377          *
378          * @param img image
379          */
380         private synchronized void addImage(RoeImage img) {
381             this.imageList.add(img);
382         }
383 
384         /**
385          * Flushes the imgae list.
386          */
387         private synchronized void flushImageList() {
388             this.imageList.clear();
389         }
390 
391         /**
392          * Generate a coordinate list for dead roe relativ to the robot origion.
393          *
394          * @return list of coordinates for dead roe relative to the robot origin.
395          */
396         private ArrayList generateCoordinatList() {
397             ArrayList<Coordinate> coordList = new ArrayList<>();
398 
399             // For all roe images
400             for (RoeImage roeImage : this.imageList)
401             {
402 
                this.roeAnalyserDevice.currentTray.getFrameCoord(roeImage.getPictureIndex(
                ));
403                 if (roeImage.getRoePositionMillimeterList().size() > 0)
404                 {
405                     for (int i = 0; i < roeImage.getRoePositionMillimeterList().size();
                    i++)
406                     {
407                         // Get Position of dead roe relative to image origin
408                         Coordinate roeCoord = (Coordinate)
                        roeImage.getRoePositionMillimeterList().get(i);
409                         // Update position raltive to robot origin.
```

```
                    double xPos = roeCoord.getxCoord() +
                    this.roeAnalyserDevice.currentTray.getFrameCoord(roeImage.getPictu
                    reIndex()).getxCoord();
                    double yPos = roeCoord.getyCoord() +
                    this.roeAnalyserDevice.currentTray.getFrameCoord(roeImage.getPictu
                    reIndex()).getyCoord() - 50; // TODO // remove - 50
                    Coordinate newCoord = new Coordinate(xPos, yPos);
                    // Adds coodrinate to list.
                    coordList.add(newCoord);
                }
            }
        }
        // add coordinate of last captured image

        coordList.add(this.roeAnalyserDevice.currentTray.getFrameCoord(this.roeAnalyse
        rDevice.currentTray.getNumberOfCameraCoordinates() - 1));
        // Flush the image list to be ready for next tray.
        this.flushImageList();
        return coordList;
    }

    /**
     * Return the current state
     *
     * @return the state of the robot
     */
    public synchronized State getCurrentState() {
        return currentState;
    }


    /**
     * Set current state
     *
     * @param currentState
     */
    public synchronized void setCurrentState(State currentState) {
        this.currentState = currentState;
    }


    /**
     * Set the search interval in minutes
     * @param minutes Minutes
     */
   public synchronized void setSearchInterval(int minutes) {
        this.searchInterval = minutes;
    }


   /**
    * Return the pause
    * @return Return the pause boolean
    */
   public boolean isPause()
           {
               return this.pause;
           }



      /**
     * Returns true if the timer has passed given nanoseconds;
     *
     * @param waitNanosecs
     * @return Returns true if timer has passed given nanoseconds
     */
    private boolean timerHasPassed(long waitMinutes) {
        waitMinutes = waitMinutes * 100000000;
        boolean timerPassed = false;
        //When (nanotime - timertimer) is bigger than wait time,
        //timer has passed given time
        if (waitMinutes < (System.nanoTime() - timerTime)) {
```

```
477                 timerPassed = true;
478             }
479 
480             return timerPassed;
481         }
482          /**
483          * Resets the timer
484          */
485         private void resetTimer() {
486             timerTime = System.nanoTime();
487         }
488 
489 
490          /**
491          * Change the value of the lights
492          *
493          * @param redVal value for red light
494          * @param greenVal value for green light
495          * @param blueVal value for blue light
496          */
497         public void setLightVal(int redVal, int greenVal, int blueVal)
498         {
499             this.roeAnalyserDevice.changeRGBLight(redVal, greenVal, blueVal);
500         }
501 
502 
503 
504         /**
505          * Convert from rev/min to mm/sec
506          */
507         private double revMinToMMSec(int velocity, double circumference){
508             double newDouble = velocity*circumference/60;
509             return newDouble;
510         }
511 
512         /**
513          * Return the number of dead roes found in the current tray
514          * @return Return the number of dead roes found in the current tray
515          */
516         public int getFoundDeadRoes()
517                 {
518                     return this.currentTray.getNrOfDeadRoe();
519                 }
520 
521     }
522 
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package RoeRobot;

import Commands.CalibParam;
import Commands.Calibrate;
import Commands.ChangeLedColor;
import Commands.Light;
import Commands.MagnetOn;
import Commands.Move;
import Commands.MagnetOff;
import Commands.StateRequest;
import Commands.Stop;
import Commands.Suction;
import Commands.FindTray;
import Commands.DiscoLight;
import Commands.Velocity;
import ImageProcessing.Camera;
import ImageProcessing.RoeImage;

import Status.Busy;
import Status.EMC;
import Status.ElevatorLimitTrigg;
import Status.EncoderOutOfRange;
import Status.EncoderOutOfSync;
import Status.LinearBotLimitTrigged;
import Status.Parameters;
import Status.ReadyToRecieve;
import Status.SafetySwitchLower;
import Status.SafetySwitchUpper;
import Status.Status;
import Status.Stopped;
import StatusListener.StatusListener;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import org.junit.rules.Stopwatch;
import SerialCommunication.SerialCommunication;
import Status.Failure;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.opencv.core.Mat;

/**
 * This class represents a robot and all its possible commands and actions. It
 * can open trays, close trays, move robot to specified x,y,z, pickup and remove
 * roe from given coordinate.
 *
 * @author Yngve & Per Espen
 */
public class RoeAnalyserDevice implements StatusListener
{

    //The timer for this object
    //Timer variabales
    private long timerTime = 0;
    private long waitTime = 30000;

    //Holds the current status sent by the roerobot
    Status currentStatus = null;
    //The calibration params
    Parameters calibrationParam = null;

    //Serial communication
    SerialCommunication serialComm;

    //Tray
    Tray currentTray;
    TrayRegister trayReg;
```

```
74
75      //Image processing variables
76      Camera camera;
77
78      //pause boolean
79      private boolean pause = false;
80
81      public RoeAnalyserDevice()
82      {
83
84          //Create and connect the serial communication
85          this.serialComm = new SerialCommunication();
86          this.serialComm.connect();
87          this.serialComm.addListener(this);
88
89          //Start the serial thread
90          this.serialComm.start();
91
92          //Create a calibration parameter
93          this.calibrationParam = new Parameters();
94
95          //Create and open the camera feed
96          this.camera = new Camera();
97
98          this.setPause(false);
99
100     }
101
102     /**
103      * Return the current Status this object has - State of the ROBOT
104      *
105      * @return Return the current Status
106      */
107     private synchronized Status getCurrentStatus()
108     {
109         return this.currentStatus;
110     }
111
112     /**
113      * Set the current status of the robot
114      *
115      * @param setStatus Set the current status of the robot
116      */
117     private synchronized void setCurrentStatus(Status setStatus)
118     {
119         this.currentStatus = setStatus;
120     }
121
122     /**
123      * Notification of incoming statuses
124      *
125      * @param status New incomming status
126      */
127     @Override
128     public synchronized void notifyNewStatus(Status status)
129     {
130         //Check if its parameter
131         if
            (State.PARAMETER.getStateStatus().getString().contentEquals(status.getString()
            ))
132         {
133             calibrationParam = (Parameters) status;
134             //printCalib();
135         }
136
137         setCurrentStatus(status);
138
139     }
140
141     /**
142      * Print the calibration parameters
143      */
144     private void printCalib()
```

```
145         {
146             System.out.println("X:" + calibrationParam.getxCalibRange());
147             System.out.println("Y:" + calibrationParam.getyCalibRange());
148             System.out.println("Z:" + calibrationParam.getzCalibRange());
149             System.out.println("Trays:" + calibrationParam.getNumberOfTrays());
150         }
151 
152         /**
153          * Update the necessary parameter stuff
154          */
155         private void updateCalibParams()
156         {
157             this.trayReg = this.calibrationParam.getTrayReg();
158         }
159 
160         //Enum for holding the states
161         private enum State
162         {
163             Busy(new Busy()),
164             Stopped(new Stopped()),
165             ReadyToRecieve(new ReadyToRecieve()),
166             EMC(new EMC()),
167             SAFETY_SWITCH_UPPER(new SafetySwitchUpper()),
168             SAFETY_SWITCH_LOWER(new SafetySwitchLower()),
169             ELEV_LIMIT_TRIGG(new ElevatorLimitTrigg()),
170             LINEARBOT_LMIT_TRIGG(new LinearBotLimitTrigged()),
171             ENCODER_OUT_OF_SYNC(new EncoderOutOfSync()),
172             ENCODER_OUT_OF_RANGE(new EncoderOutOfRange()),
173             PARAMETER(new Parameters()),
174             Failure(new Failure());
175 
176             //Hashmap for lookup
177             private static final HashMap<Status, State> lookup = new HashMap<Status,
                State>();
178 
179             //Put the states with the accompanied value in the hashmap
180             static
181             {
182                 //Create reverse lookup hash map
183                 for (State s : State.values())
184                 {
185                     lookup.put(s.getStateStatus(), s);
186                 }
187             }
188             //Satus address
189             private Status status;
190 
191             private State(Status status)
192             {
193                 this.status = status;
194             }
195 
196             public Status getStateStatus()
197             {
198                 return status;
199             }
200 
201             public static State get(String address)
202             {
203                 //the reverse lookup by simply getting
204                 //the value from the lookup HsahMap.
205                 return lookup.get(address);
206             }
207         }
208 
209         /**
210          * Open tray will open a tray with a specific number.
211          *
212          * @param trayNumber is the number of the tray wanted to open.
213          * @return False if the tray number do not exist.
214          */
215         public boolean openTray(Tray workTray)
216         {
```

```
217
218         //Return bool how the task went
219         boolean succesful = true;
220         boolean searching = true;
221         //Switch case variables
222         int task = 0;
223         // the tasks to be completed
224         final int moveRobotToHandle = 0, lockGripper = 1, moveOpenTray = 2,
            releaseGripper = 3, moveToDefault = 4, done = 5, findTray = 6;
225
226         //Check if the tray was retrieved succesfully, else exit the method
227         if (workTray == null)
228         {
229             task = done;
230             succesful = false;
231         }
232
233         //updateStatus();
234         //While loop to keep in the case until done or failure
235         while (succesful && searching)
236         {
237             //if its paused, dont perform any actions
238             if (!this.isPause())
239             {
240                 //Switch case to do the tasks;
241                 switch (task)
242                 {
243                     //Move robot to the position of the handle
244                     case moveRobotToHandle:
245                         //Wait for the Robot to finish(get in ready to recieve
                            state) before sending more requests to it
246                         if (robotIsReady(waitTime))
247                         {
248                             //Get the coordinates from the handle
249                             Coordinate handleCord = workTray.getHandleCoordinate();
250                             //Move to the handle
251                             this.move(handleCord);
252                             setStatusToBusy();
253                             task = findTray;
254                         }
255                         //Something is faulty, end the task
256                         else
257                         {
258                             succesful = false; //Set succesful to false
259                             task = done;        //End the task
260                         }
261
262                         break;
263                     //Find the tray by find tray command
264                     case findTray:
265                     //Wait for robot to be ready
266                         if (robotIsReady(waitTime))
267                         {
268                             //Send command for the linear bot to move until tray is
                                detected
269                             FindTray ftray = new FindTray();
270                             this.serialComm.addSendQ(ftray);
271                             //Set status to busy and
272                             setStatusToBusy();
273                             task = lockGripper;
274                         }
275                         //Something is faulty, end the task
276                         else
277                         {
278                             succesful = false; //Set succesful to false
279                             task = done;        //End the task
280                         }
281                         break;
282                     case lockGripper:
283                         //Wait for the Robot to finish(get in ready to recieve
                            state) before sending more requests to it
284                         if (robotIsReady(waitTime))
285                         {
```

```
286                             //Create the lock grip command
287                             MagnetOn cmdLockGripper = new MagnetOn();
288                             //Lock the gripper
289                             this.serialComm.addSendQ(cmdLockGripper);
290                             // setStatusToBusy();
291 
292                             System.out.println("Magnet turned on");
293                             task = moveOpenTray;
294                         } //Something is faulty, end the task
295                         else
296                         {
297                             succesful = false; //Set succesful to false
298                             task = done;        //End the task
299                         }
300 
301                         break;
302 
303                     case moveOpenTray:
304                         //Wait for the Robot to finish(get in ready to recieve
                            state) before sending more requests to it
305                         if (robotIsReady(waitTime))
306                         {
307                             //Send move command to open the tray;
308                             if (workTray.getOpenCoord() != null)
309                             {
310                                 move(workTray.getOpenCoord());
311                                 setStatusToBusy();
312                                 task = releaseGripper;
313                             }
314 
315                         } //Something is faulty, end the task
316                         else
317                         {
318                             succesful = false; //Set succesful to false
319                             task = done;        //End the task
320                         }
321 
322                         break;
323 
324                     case releaseGripper:
325                         //Wait for the Robot to finish
326                         if (robotIsReady(waitTime))
327                         {
328                             //Create the release grip command
329                             MagnetOff cmdReleaseGrip = new MagnetOff();
330                             //Send command to release the gripper
331                             this.serialComm.addSendQ(cmdReleaseGrip);
332                             // setStatusToBusy();
333 
334                             task = moveToDefault;
335                         } //Something is faulty, end the task
336                         else
337                         {
338                             succesful = false; //Set succesful to false
339                             task = done;        //End the task
340                         }
341 
342                         break;
343                     case moveToDefault:
344                         //Wait for the Robot to finish(get in ready to recieve
                            state) before sending more requests to it
345                         if (robotIsReady(waitTime))
346                         {       //Move to default after opening tray
347                             move(workTray.getDefaultZPosCoord());
348                             setStatusToBusy();
349                         } //Something is faulty, end the task
350                         else
351                         {
352                             succesful = false; //Set succesful to false
353                             task = done;        //End the task
354                         }
355 
356                         if (robotIsReady(waitTime))
```

```
                        {       //Move to default after opening tray
                            move(workTray.getDefaultCoord());
                            setStatusToBusy();

                            task = done;
                        } //Something is faulty, end the task
                        else
                        {
                            succesful = false; //Set succesful to false
                            task = done;        //End the task
                        }
                        break;

                    case done:
                        searching = false;
                        //updateStatus();
                        break;
                    default:
                        break;
                }

                //Save this opening tray to the currentTray
                currentTray = workTray;
            }

        }
        return succesful;

    }

    /**
     * Close Tray will close a tray with a specific number.
     *
     * @param trayNumber is the nuber of the tray wanted to close
     * @return False if the tray number do not exist.
     */
    public boolean closeTray(Tray workTray)
    {

        //Return bool how the task went
        boolean sucessful = true;
        boolean working = true;
        //Switch case variables
        int task = 0;
        // the tasks to be completed
        final int moveRobotToCloseHandleXY = 0, lockGripper = 1, moveCloseTray = 2,
        releaseGripper = 3, moveToDefault = 4, done = 5, moveToZHandle = 6;

        //Check if the tray was retrieved succesfully, else exit the method
        if (workTray == null)
        {
            task = done;
            sucessful = false;
        }
        while (working)
        {
            //Switch case to do the tasks;
            switch (task)
            {
                  case moveToDefault:
                    //Move to default after opening tray
                    move(workTray.getDefaultZPosCoord());
                    setStatusToBusy();
                    task = moveRobotToCloseHandleXY;
                    break;
                //Move robot to the position of the handle
                case moveRobotToCloseHandleXY:
                    //Wait for the Robot to finish(get in ready to recieve state)
                    before sending more requests to it
                    if (robotIsReady(waitTime))
                    {
                        //Get the coordinates from the handle
                        this.move(workTray.getOpenCoord());
```

```
428                         setStatusToBusy();
429                         task = moveToZHandle;
430                     } //Something is faulty, end the task
431                     else
432                     {
433                         sucessful = false; //Set succesful to false
434                         task = done;        //End the task
435                     }
436                     break;
437 
438                 case moveToZHandle:
439                     //Wait for the Robot to finish(get in ready to recieve state)
                        before sending more requests to it
440                     if (robotIsReady(waitTime))
441                     {
442                         //Get the coordinates from the handle
443                         this.move(workTray.getHandleZCoord());
444                         setStatusToBusy();
445                         task = lockGripper;
446                     } //Something is faulty, end the task
447                     else
448                     {
449                         sucessful = false; //Set succesful to false
450                         task = done;        //End the task
451                     }
452                     break;
453 
454                 case lockGripper:
455                     //Wait for the Robot to finish(get in ready to recieve state)
                        before sending more requests to it
456                     if (robotIsReady(waitTime))
457                     {
458                         //Create the lock grip command
459                         MagnetOn cmdLockGripper = new MagnetOn();
460                         //Lock the gripper
461                         this.serialComm.addSendQ(cmdLockGripper);
462                         task = moveCloseTray;
463                         // setStatusToBusy();
464                         System.out.println("Magnet ON");
465                     } //Something is faulty, end the task
466                     else
467                     {
468                         sucessful = false; //Set succesful to false
469                         task = done;        //End the task
470                     }
471                     break;
472                 case moveCloseTray:
473                     //Wait for the Robot to finish(get in ready to recieve state)
                        before sending more requests to it
474                     if (robotIsReady(waitTime))
475                     {   //Send move command to open the tray;
476                         move(workTray.getCloseTrayCoord());
477                         setStatusToBusy();
478                         task = releaseGripper;
479                         System.out.println("Moving tray to close pos");
480                     } //Something is faulty, end the task
481                     else
482                     {
483                         sucessful = false; //Set succesful to false
484                         task = done;        //End the task
485                     }
486                     break;
487                 case releaseGripper:
488                     //Wait for the Robot to finish(get in ready to recieve state)
                        before sending more requests to it
489                     if (robotIsReady(waitTime))
490                     {
491                         //Create the release grip command
492                         MagnetOff cmdReleaseGrip = new MagnetOff();
493                         //Send command to release the gripper
494                         this.serialComm.addSendQ(cmdReleaseGrip);
495                         //setStatusToBusy();
496                         task = done;
```

```
497
498                     } //Something is faulty, end the task
499                     else
500                     {
501                         sucessful = false; //Set succesful to false
502                         task = done;        //End the task
503                     }
504                     break;
505
506
507                 case done:
508                     working = false;
509                     break;
510             }
511         }
512
513         //set current tray to null
514         currentTray = null;
515
516         //Return the completion bool
517         return sucessful;
518     }
519
520     /**
521      * Removes roe from all the coordinates given in the Arraylist.
522      *
523      * @param coordinates Arraylist of coordinates to be removed from.
524      */
525     public boolean removeRoe(ArrayList<Coordinate> cordinates)
526     {
527
528         //Return bool if the task was completed or not
529         boolean succesful = true;
530
531         Iterator itr = cordinates.iterator();
532         while (itr.hasNext() && succesful)
533         {
534
535             //if paused is set, dont perform any actions
536             if (!this.isPause())
537             {
538                 //Get the next coordinate
539                 Coordinate cord = (Coordinate) itr.next();
540
541                 //Wait for the Robot to finish(get in ready to recieve state) before
                    sending more requests to it
542                 if (robotIsReady(waitTime) && succesful)
543                 {
544                     //Move the robot to the dead roe position
545                     this.move(cord);
546                     this.setStatusToBusy();
547                 } //Something is faulty, end the task
548                 else
549                 {
550                     succesful = false;
551                 }
552
553                 //Wait for the Robot to finish(get in ready to recieve state) before
                    sending more requests to it
554                 if (robotIsReady(waitTime) && succesful)
555                 {
556                     //Pick up the dead roe
557                     pickUpRoe(currentTray);
558                 } //Something is faulty, end the task
559                 else
560                 {
561                     succesful = false;
562                 }
563             }
564         }
565         return succesful;
566     }
567
```

```
568         /**
569          * Calibrate will send a calibration command to the roerobot
570          */
571         public boolean calibrate()
572         {
573             boolean succesful = true;
574             long calibWaitTime = 8000;
575             // Generate a Calibration command.
576             // Send cmd.
577             Calibrate calicmd = new Calibrate();
578             serialComm.addSendQ(calicmd);
579 
580             //Wait for the Robot to finish(get in ready to recieve state) before sending
                more requests to it
581             //This will but the robot to ask about state until it is ready for next
                command
582             if (robotIsReady(calibWaitTime))
583             {
584                 //Check if calib parameter has been updated with both
585                 //Set in loop until both calibration parameters are returned
586                 while (!this.calibrationParam.isSent())
587                 {
588                     //Send calib param command to get calibration parameters
589                     CalibParam cmdCalibPar = new CalibParam();
590                     if (timerHasPassed(calibWaitTime))
591                     {
592                         serialComm.addSendQ(cmdCalibPar);
593                         resetTimer();
594                     }
595 
596                 }
597             } //Something is faulty, end the task
598             else
599             {
600                 succesful = false;
601             }
602 
603             //Update the calibe parameter
604             updateCalibParams();
605 
606             //Send state request to the robot
607             updateStatus();
608 
609             return succesful;
610         }
611 
612         /**
613          * Sends a stop Command to the robot
614          */
615         public void stopRobot()
616         {
617             Stop stop = new Stop();
618             serialComm.addSendQ(stop);
619         }
620 
621         /**
622          * Send all the required commands for picking up roe Go down to Z height,
623          * and send suction
624          */
625         public boolean pickUpRoe(Tray thisTray)
626         {
627             //Return bool for result of task
628             boolean succesful = true;
629             boolean working = true;
630             //Switch case variables
631             int task = 0;
632             // the tasks to be completed
633             final int moveDown = 0, suck = 1, moveUp = 2, done = 3;
634 
635             while (working)
636             {
637                 //if paused is set, dont perform any actions
638                 if (!this.isPause())
```

```
639             {
640                 //Switch case to do the tasks;
641                 switch (task)
642                 {
643                     //Move down to the roe pickup height
644                     case moveDown:
645 
646                         //Send command if robot becomes ready
647                         if (robotIsReady(waitTime))
648                         {
649                             this.move(currentTray.getRoePickupZCoord());
650                             this.setStatusToBusy();
651                             task = suck;
652                         } //Something is faulty, end the task
653                         else
654                         {
655                             succesful = false;
656                             task = done;
657                         }
658 
659                         break;
660                     //Do the suction task - Check robot is ready, send suction command
661                     case suck:
662                         //Send command if robot becomes ready
663                         if (robotIsReady(waitTime))
664                         {
665                             Suction cmdSuck = new Suction();
666                             //Send suction command
667                             serialComm.addSendQ(cmdSuck);
668                             setStatusToBusy();
669                             task = moveUp;
670                         } //Something is faulty, end task
671                         else
672                         {
673                             succesful = false;
674                             task = done;
675                         }
676                         break;
677                     //Move the robot up, from the tray
678                     case moveUp:
679                         //Send command if robot becomes ready
680                         if (robotIsReady(waitTime))
681                         {
682                             this.move(thisTray.getDefaultZPosCoord());
683                             this.setStatusToBusy();
684                             task = done;
685                         } //Something is faulty, end task
686                         else
687                         {
688                             succesful = false;
689                             task = done;
690                         }
691                         break;
692                     //The task is done, break
693                     case done:
694                         working = false;
695                         break;
696                 }
697             }
698         }
699         return succesful;
700     }
701 
702     /**
703      * Send all the required commands for picking up roe Go down to Z height,
704      * and send suction
705      */
706     public boolean pickUpRoe()
707     {
708         //Return bool for result of task
709         boolean succesful = true;
710         boolean working = true;
711         //Switch case variables
```

```
712             int task = 0;
713             // the tasks to be completed
714             final int moveDown = 0, suck = 1, moveUp = 2, done = 3;
715 
716             while (working)
717             {
718                 //Switch case to do the tasks;
719                 switch (task)
720                 {
721                     //Move down to the roe pickup height
722                     case moveDown:
723 
724                         //Send command if robot becomes ready
725                         if (robotIsReady(waitTime))
726                         {
727                             this.move(currentTray.getRoePickupZCoord());
728                             this.setStatusToBusy();
729                             task = suck;
730                         } //Something is faulty, end the task
731                         else
732                         {
733                             succesful = false;
734                             task = done;
735                         }
736 
737                         break;
738                     //Do the suction task - Check robot is ready, send suction command
739                     case suck:
740                         //Send command if robot becomes ready
741                         if (robotIsReady(waitTime))
742                         {
743                             Suction cmdSuck = new Suction();
744                             //Send suction command
745                             serialComm.addSendQ(cmdSuck);
746 
747                             task = moveUp;
748                             System.out.println("Suction command sent");
749                         } //Something is faulty, end task
750                         else
751                         {
752                             succesful = false;
753                             task = done;
754                         }
755                         break;
756                     //Move the robot up, from the tray
757                     case moveUp:
758                         //Send command if robot becomes ready
759                         if (robotIsReady(waitTime))
760                         {
761                             this.move(currentTray.getDefaultCoord());
762                             this.setStatusToBusy();
763                             task = done;
764                         } //Something is faulty, end task
765                         else
766                         {
767                             succesful = false;
768                             task = done;
769                         }
770                         break;
771                     //The task is done, break
772                     case done:
773                         working = false;
774                         break;
775                 }
776             }
777 
778             return succesful;
779         }
780 
781         /**
782          * Returns true if robot is in ready state, false if some status is marked
783          * as critical
784          *
```

```
785         * @return Returns true if robot is in ready state, false if robot has error
786         */
787        private boolean robotIsReady(long pollTime)
788        {
789            boolean robotState = true;
790            //Reset the timer
791            resetTimer();
792
793            //Check if robot is ready for new command & no faults are present
794            while ((!isReady() && !robotFaultyStatus()) || this.isPause())
795            {
796                //No need to poll, the isReady or robotFaulty will be updated when the
                   robot changes state
797                /*
798                //After a set wait time, update the status
799                if (timerHasPassed(pollTime))
800                {
801                    //Send status update request
802                    updateStatus();
803                    resetTimer();   //Reset timer
804                }
805                 */
806
807            }
808            //Check if robot has a critical error
809            if (robotFaultyStatus())
810            {
811                robotState = false;
812            }
813
814            return robotState;
815        }
816
817        /**
818         * Wait for status to be ready, return true if status is ready, false if not
819         *
820         * @return Return true if status is ready
821         */
822        private boolean isReady()
823        {
824            //Return bool
825            boolean ready = false;
826            //Check status
827            if (getCurrentStatus() != null)
828            {
829                //Check if status is ready to recieve
830                if
                   (getCurrentStatus().getString().equalsIgnoreCase(State.ReadyToRecieve.getS
                   tateStatus().getString().toLowerCase()))
831                {
832                    ready = true;
833                }
834            }
835            //Return the decided bool state
836            return ready;
837        }
838
839        /**
840         * Set the status to busy - meant to be used when sending a command, and new
841         * waiting for new update
842         *
843         * @return
844         */
845        private void setStatusToBusy()
846        {
847            this.setCurrentStatus(State.Busy.getStateStatus());
848        }
849
850        /**
851         * Return the number of trays.
852         *
853         * @return
854         */
```

```
855         public int getNumberOfTrays()
856         {
857             // TODO: Fill method
858             // Generate cmd for requesting nr of trays in rack from arduino.
859             return this.calibrationParam.getNumberOfTrays(); // TODO: Return number of
                trays in rack.
860         }
861 
862         /**
863          * Return the number of pictures to capture in current tray. .
864          *
865          * @return int with the nuber of pictures to take in current tray.
866          */
867         public int getNumberOfPicturesInTray()
868         {
869             return this.currentTray.getNumberOfCameraCoordinates();
870         }
871 
872         /**
873          * Move method used for moving the end-effector to a specific X,Y,Z
874          * coordinat
875          *
876          * @param coordinat in a global coordinat system.
877          */
878         public void move(Coordinate cordinat)
879         {
880             //Do what necessary form moving the end-effector to a spesific coordinat.
881             //Create the command and set the appropriate values
882             Move moveCmd = new Move();
883 
884             //Check which coordinate value should be sent
885             //Check against -1
886             if (Double.compare(cordinat.getxCoord(), -1) != 0)
887             {
888                 moveCmd.setxMove(cordinat.getxCoord());
889                 moveCmd.setxMoveBool(true);
890             }
891 
892             if (Double.compare(cordinat.getyCoord(), -1) != 0)
893             {
894                 moveCmd.setyMove(cordinat.getyCoord());
895                 moveCmd.setyMoveBool(true);
896             }
897 
898             if (Double.compare(cordinat.getzCoord(), -1) != 0)
899             {
900                 moveCmd.setzMove(cordinat.getzCoord());
901                 moveCmd.setzMoveBool(true);
902             }
903 
904             //Add to the communication sending queue
905             serialComm.addSendQ(moveCmd);
906         }
907 
908         /**
909          * Take a picture at an specific frame number. Return true if completed,
910          * false if frame couldn't be found or robot got faulty status
911          *
912          * @param frameNumber The number wanted to take picture of
913          */
914         public RoeImage takePicture(Tray workingTray, int frameNumber)
915         {
916             //Return bool for result of task
917             boolean working = true;
918 
919             //RoeImage to return
920             RoeImage imageTaken = null;
921             //Switch case variables
922             int task = 0;
923 
924 
925             // the tasks to be completed
926             final int moveToFrame = 0, takePic = 1, done = 2;
```

```
927
928             //Keep the program in loop until its done
929             while (working)
930             {
931                 //Switch case to do the tasks;
932                 switch (task)
933                 {
934                     //Move to the given frame number
935                     case moveToFrame:
936                         //Send command if robot becomes ready
937                         if (robotIsReady(waitTime))
938                         {
939                             //Get the frame coord from the current tray
940                             Coordinate frameCord = workingTray.getFrameCoord(frameNumber);
941 
942                             //Check if the frame was found
943                             if (frameCord != null)
944                             {
945                                 this.move(frameCord);
946                                 this.setStatusToBusy();
947 
948                                 task = takePic;
949                             } //Set the completion of this task to fail and exit it
950                             else
951                             {
952                                 task = done;
953                             }
954                         } //Something is faulty, end the task
955                         else
956                         {
957                             task = done;
958                         }
959                         break;
960                     //Move down to the roe pickup height
961                     //Do the suction task - Check robot is ready, send suction command
962                     case takePic:
963                         //Send command if robot becomes ready
964                         if (robotIsReady(waitTime))
965                         {
966                             imageTaken = this.camera.takePicture((float)
                                workingTray.getWaterSurfaceOffsetForCamera(), frameNumber);
967                             //Take the pic
968                             task = done;
969                         } //Something is faulty, end task
970                         else
971                         {
972                             task = done;
973                         }
974                         break;
975                     //The task is done, break
976                     case done:
977                         working = false;
978                         break;
979                 }
980             }
981 
982             return imageTaken;
983         }
984 
985         /**
986          * Send a request for state update
987          */
988         public void updateStatus()
989         {
990             StateRequest stateReq = new StateRequest();
991             serialComm.addSendQ(stateReq);
992         }
993 
994 
995 
996         /**
997          * Turn light on (send light command with value 1 as payload)
998          */
```

```
999         public void turnOnLight()
1000        {
1001            //Create command
1002            Light cmdLight = new Light();
1003            //Make the control byte to be sent
1004            cmdLight.setOn();
1005
1006            ChangeLedColor cmdColor = new ChangeLedColor();
1007
1008            cmdColor.setMultipleIntValue(200, 100, 50);
1009            //Return bool for result of task
1010            boolean succesful = true;
1011
1012            //Check if robot is ready for new command & no faults are present
1013            while (!isReady() && !robotFaultyStatus())
1014            {
1015                //After a set wait time, update the status
1016                if (timerHasPassed(waitTime))
1017                {
1018                    updateStatus(); //Send status update request
1019                    resetTimer();   //Reset timer
1020                }
1021            }
1022            //Check if robot has a critical error
1023            if (robotFaultyStatus())
1024            {
1025                succesful = false;
1026            }
1027
1028            //Send command
1029            serialComm.addSendQ(cmdColor);
1030        }
1031
1032        /**
1033         * Turn light off (send light command with value 0 as payload)
1034         */
1035        public boolean turnOffLight()
1036        {
1037            //Return bool for result of task
1038            boolean succesful = true;
1039
1040            //Toggle light
1041            //Create command
1042            Light cmdLight = new Light();
1043            cmdLight.setOff();
1044            //Check if robot is ready for new command & no faults are present
1045            while (!isReady() && !robotFaultyStatus())
1046            {
1047                //After a set wait time, update the status
1048                if (timerHasPassed(waitTime))
1049                {
1050                    updateStatus(); //Send status update request
1051                    resetTimer();   //Reset timer
1052                }
1053            }
1054            //Check if robot has a critical error
1055            if (robotFaultyStatus())
1056            {
1057                succesful = false;
1058            }
1059
1060            //Send command
1061            serialComm.addSendQ(cmdLight);
1062
1063            return succesful;
1064        }
1065
1066        /**
1067         * Sends a command to the robot to change its speed
1068         *
1069         * @param newVelocity The speed to change to
1070         * @return Returns true if it was succesfull, false it something happened
1071         */
```

```
1072        public synchronized boolean changeVelocity(int newVelocity)
1073        {
1074            // Boolean for check if sucsess
1075            boolean sucsess = false;
1076            //Create command
1077            Velocity cmdVelocity = new Velocity();
1078            //Make the control byte to be sent
1079            cmdVelocity.setIntValue(newVelocity);
1080            cmdVelocity.setForElevatorRobot(false);
1081
1082            if (this.robotIsReady(waitTime))
1083            {
1084                //Send command
1085                serialComm.addSendQ(cmdVelocity);
1086                sucsess = true;
1087            } else
1088            {
1089                sucsess = false;
1090            }
1091
1092            return sucsess;
1093        }
1094
1095        /**
1096         * Change the RGB color of the leds
1097         *
1098         * @param red Red value
1099         * @param green Green value
1100         * @param blue Blue value
1101         * @return Returns true if it was succesfull
1102         */
1103        public synchronized void changeRGBLight(int red, int green, int blue)
1104        {
1105            // Boolean for check if sucsess
1106            boolean sucsess = false;
1107            //Create command
1108            ChangeLedColor changeLedColor = new ChangeLedColor();
1109            //Make the control byte to be sent
1110            changeLedColor.setMultipleIntValue(red, green, blue);
1111            changeLedColor.setForElevatorRobot(false);
1112
1113            serialComm.addSendQ(changeLedColor);
1114
1115        }
1116
1117        /**
1118         * Resets the timer
1119         */
1120        private void resetTimer()
1121        {
1122            timerTime = System.nanoTime();
1123        }
1124
1125        /**
1126         * Returns true if the timer has passed given nanoseconds;
1127         *
1128         * @param waitNanosecs
1129         * @return Returns true if timer has passed given nanoseconds
1130         */
1131        private boolean timerHasPassed(long waitMillisec)
1132        {
1133            waitMillisec = waitMillisec * 1000000;
1134            boolean timerPassed = false;
1135            //When (nanotime - timertimer) is bigger than wait time,
1136            //timer has passed given time
1137            if (waitMillisec < (System.nanoTime() - timerTime))
1138            {
1139                timerPassed = true;
1140            }
1141
1142            return timerPassed;
1143        }
1144
```

```
1145        /**
1146         * Returns true if the the current status is regarded as a fault
1147         *
1148         * @return Returns true if the the current status is regarded as a fault
1149         */
1150        private boolean robotFaultyStatus()
1151        {
1152            boolean returnThis = false;
1153            if (getCurrentStatus() != null)
1154            {
1155                returnThis = getCurrentStatus().critical();
1156            }
1157            return returnThis;
1158        }
1159
1160        /**
1161         * Send discoLights command to the robots
1162         */
1163        public void discoLights()
1164        {
1165            DiscoLight disco = new DiscoLight();
1166            serialComm.addSendQ(disco);
1167        }
1168
1169
1170        /**
1171         * Return the calibration parameter status
1172         *
1173         * @return Return the status with the calibration parameters
1174         */
1175        public Parameters getCalibrationParams()
1176        {
1177            return this.calibrationParam;
1178        }
1179
1180        /**
1181         * Pause
1182         *
1183         * @return
1184         */
1185        public synchronized boolean isPause()
1186        {
1187            return this.pause;
1188        }
1189
1190        public synchronized void setPause(boolean pause)
1191        {
1192            this.pause = pause;
1193        }
1194
1195    }
1196
```

```java

package RoeRobot;

import java.util.ArrayList;


/**
 * Class represents a tray. The tray has a width, hight and a depth. The tray
 * will also know where it is plased in a coordinate system by knowing its upper
 * and lower posistion.
 *
 * @author Yngve
 */
public class Tray
{




    //Number for this tray
    private final int nr;

    private final int width = 10; // defined in mm
    private final int depth = 10; // defined in mm
    private final int distUpperLowerPos = 20; // Distanse form the upper position to the lowest point in the tray
    private final int upperPos = 30; // defined in mm parallel to the Z-axis
    private final int lowerPos = 40; // defined in mm parallel to the Z-axis

    private final int TotalCameraPositions = 24;  //Total amount of camera positions

    private int nrOfRemovedRoe;
    private int nrOfDeadRoe;


    //The flag position for this tray
    private int flagPosZ;


    //**** OFFSETS IN "mm"****
    //New alu brancket adds 32mm

   // x and y coordinates of first camera position
    private double imageCoordX = 25; //======================ENDRES
    private double imageCoordY = 200;  //======================ENDRES


      //Distance from bottom of flag to top of water surface on tray
    private double bottomFlagToTraySurface = 11 + 10;
    //Distance from bottom of flag to top of tray
    private double bottomFlagToTopOfTray = 26;
      //Distance from the flag Z pos to the magnet -> Z value offset
    private double distFlagToEndEffector = 34;
    //Distance from the flag Z pos to the magnet -> Z value offset
    private double bottomFlagToMagnet = 34+35;
    //Distance from the flag Z pos to the magnet -> Z value offset
    //distFlagToEndEffector + from endEffector to magnet
    private double distFlagPosToMagnetZ = bottomFlagToMagnet;
    //Distance from the flag Z pos to the default position of the robot on this tray -> Z value offset
    private double distFlagPosToDefaultZ = bottomFlagToTopOfTray;
    //Distance from flag Z pos to the suction point. In reality, from bottom sensor to the suction end. Z offset
    private double distFlagPosToSucktionZ = distFlagToEndEffector + bottomFlagToTraySurface;
    //Distance from the calibrated Y MAX pos to where the tray should be put back. In reality, where the robot should leave the tray. in Y position.
    private double closeTrayOffsetY = 15;
    private double openTrayOffsetY = 38;
    //The default Z position.. Used to store the calculated Z-pos
    private double defaultZ;


```

```
68      //Wanted distance from the tray water to the camera lens
69      private double waterSurfaceOffsetForCamera = 80;
70
71
72      //Offset between ir beam and camera lens
73     // private double beamToCamera = 50 + 20;
74      //Distance from the flag to the camera on end effector
75      private double fromFlagToCamera = 80;
76      //From camera to tray water, store the calculated camera height
77      private double cameraHeight;
78
79
80      //Coordinate for diff positions related to the tray
81      //Coord for the handle when tray its closed
82      private Coordinate handleCoord;
83      //Coord for the handle when tray its closed
84      private Coordinate handleZCoord;
85      //Default pos for roebot
86      private Coordinate defaultPosCoord;
87      //Coordinate for opening the tray - from holding tray handle to open
88      private Coordinate openTrayCoord;
89      //Coordinate for Z coordinate(down to tray roe level)
90      private Coordinate pickupRoeZCoord;
91       //Default Z pos for roebot - Over working tray
92      private Coordinate defaultZPosCoord;
93       //Coordinate for closing the tray, no Z movement
94      private Coordinate closeTrayCoord;
95
96
97
98      // image width and height in mm
99      private double imageWidth = 50;
100     private double imageHeight = 100;
101
102
103     // list holding positions of camera coordinates
104     private final ArrayList<Coordinate> cameraPositions;
105
106
107     public Tray(int nr, int flagposZ)
108     {
109         //Flag positions in Z axis, at bottom of flag
110         this.flagPosZ = flagposZ;
111         this.nr = nr;
112         //Set all the  coords to null
113         handleCoord = null;
114         defaultPosCoord = null;
115         openTrayCoord = null;
116         pickupRoeZCoord = null;
117         closeTrayCoord = null;
118
119         this.cameraPositions = new ArrayList<Coordinate>();
120
121         //Calculate camera height //flagposZ + bottomFlagToTraySurface - random
            offset;
122         cameraHeight = (double)flagposZ + bottomFlagToTraySurface - 8;
123
124         // fill list with camera coordinates
125         this.createCameraCoordinates();
126     }
127
128
129
130
131     /**
132      * fill the array of cameracoordinates
133      */
134     private void createCameraCoordinates()
135     {
136
137         // create top row of coordinates
138         // antall bilder skal være 12
139         for(int i = 0; i <= 11; i++)
```

```
140             {
141                 Coordinate nextCoord = new Coordinate(this.imageCoordX +
                    this.imageWidth*i, this.imageCoordY + this.imageHeight, cameraHeight);
142                 this.addCameraPos(nextCoord);
143             }
144
145             // create bottom row of coordinates
146             for(int i = 0; i >= 11; i--)
147             {
148                 Coordinate nextCoord = new Coordinate(this.imageCoordX +
                    this.imageWidth*i, this.imageCoordY, cameraHeight);
149                 this.addCameraPos(nextCoord);
150             }
151         }
152
153         /**
154          * Get upper position returns the upper limit position of the tray defined
155          * in mm from the bottom of a global coordinat system
156          *
157          * @return
158          */
159         public int getUpperPos() {
160             return this.upperPos;
161         }
162
163         /**
164          * Get lower position returns the lower limit position of the tray defined
165          * in mm from the bottom of a global coordinat system
166          *
167          * @return
168          */
169         public int getLowerPos() {
170             return lowerPos;
171         }
172
173         /**
174          * Get distanse between upper and lower position
175          *
176          * @return int with lower positon.
177          */
178         public int getDistUpperLowerPos() {
179             return distUpperLowerPos;
180         }
181
182         /**
183          * Increase the number of removed dead roe in the tray.
184          *
185          * @param removedRoe is the number of dead roe witch has been removed.
186          */
187         public void increaseNrOfRemovedRoe(int removedRoe) {
188             this.nrOfRemovedRoe = this.nrOfRemovedRoe + removedRoe;
189         }
190
191         /**
192          * Get number of removed roe returns the total number of dead roe witch has
193          * been removed from the tray.
194          *
195          * @return number of removed roe.
196          */
197         public int getNrOfRemovedRoe() {
198             return nrOfRemovedRoe;
199         }
200
201         /**
202          * Get width returns the width of the tray
203          *
204          * @return width of tray
205          */
206         public int getWidth() {
207             return width;
208         }
209
210         /**
```

```
211         * Get depth returns the width of the tray
212         *
213         * @return depth of tray
214         */
215        public int getDepth() {
216            return depth;
217        }
218
219        /**
220         * Return this trays number
221         * @return The number of this tray
222         */
223        public int getTrayNr()
224        {
225            return this.nr;
226        }
227
228        /**
229         * Coordinates for the handle to this tray
230         * @return Return coordinate for the handle of this tray
231         */
232        public Coordinate getHandleCoordinate()
233        {
234            return this.handleCoord;
235        }
236
237
238          /**
239         * Return the Z Coordinates for the handle to this tray
240         * @return Return the Z coordinate for the handle of this tray
241         */
242        /*
243        public Coordinate getZHandleCoord()
244        {
245            Coordinate zCord = new Coordinate(0,0,
                this.getHandleCoordinate().getzCoord());
246            return zCord;
247        }
248        */
249        /**
250         * Return the coords for pulling the tray to open position
251         * @return Return the coords for opening the tray
252         */
253        public Coordinate getOpenCoord()
254        {
255           return this.openTrayCoord;
256        }
257
258            /**
259         * Return the coords for opening the tray
260         * @return Return the coords for opening the tray
261         */
262        public Coordinate getDefaultCoord()
263        {
264           return this.defaultPosCoord;
265        }
266
267
268        public Coordinate getDefaultZPosCoord()
269        {
270            return defaultZPosCoord;
271        }
272
273
274        /**
275         * Return the z coord where roe should be pickuped up
276         * @return Return the z coord where roe should be pickuped up
277         */
278        public Coordinate getRoePickupZCoord()
279        {
280            return this.pickupRoeZCoord;
281        }
282
```

```
283         /**
284          * Returns the coord for grabbing the handle to close the tray
285          * @return  Returns the coord for grabbing the handle to close the tray
286          */
287         public Coordinate getCloseTrayCoord()
288         {
289          return this.closeTrayCoord;
290         }
291 
292          /**
293           * Return coordinate for the tray handle in Z position
294           * @return Return coordinate for the tray handle in Z position
295           */
296           public Coordinate getHandleZCoord()
297         {
298             return this.handleZCoord;
299         }
300 
301 
302         /**
303          * Return the coordinate for the desired frame
304          * @param nr Number for the fram wanted
305          * @return Return the coordinate for the frame corresponding with the param
              famre number
306          */
307         public Coordinate getFrameCoord(int nr)
308         {
309             Coordinate returnCoord = null;
310             //Check if the coordinate is in the array
311             if(nr < this.cameraPositions.size())
312                 returnCoord = this.cameraPositions.get(nr);
313 
314            return returnCoord;
315         }
316 
317         /**
318          * add position for camera
319          * @param camPos
320          */
321         private void addCameraPos(Coordinate camPos)
322         {
323             this.cameraPositions.add(camPos);
324         }
325 
326 
327            public int getFlagPosZ()
328         {
329             return flagPosZ;
330         }
331 
332           /**
333            * Take all the calib parameters and create the coordinates required for this
                tray to be handled by the system
334          * @param xParam
335          * @param yParam
336            */
337           public void createTrayCoords(int xParam, int yParam)
338           {
339               //Calculate the default pos'es
340               this.defaultZ = this.flagPosZ + distFlagPosToDefaultZ;
341               this.defaultPosCoord = new Coordinate(xParam/2, yParam/2, this.defaultZ);
342               this.defaultZPosCoord = new Coordinate(this.defaultZ);
343 
344               //Calculate the handle coordinate
345               this.handleCoord = new Coordinate(xParam/2, yParam/10,
                  this.flagPosZ-this.distFlagPosToMagnetZ);
346               //Create the coordinate for the Z position for the handle on the tray
347               this.handleZCoord = new
                  Coordinate(this.flagPosZ-this.distFlagPosToMagnetZ);
348               //Create the open tray coordinate
349               this.openTrayCoord = new Coordinate(xParam/2, openTrayOffsetY);
350               //Create the coordinate for closing the tray, just Y movement in reality
351               this.closeTrayCoord = new Coordinate(xParam/2,
```

```
                yParam-this.closeTrayOffsetY);
352             //this.closeTrayZCoord = new Coordinate(xParam/2,
                yParam-this.closeTrayOffsetX);
353             //The z coord for picking up roe
354             this.pickupRoeZCoord = new
                Coordinate(this.defaultZ-distFlagPosToSucktionZ);
355         }
356
357
358     /**
359     * Get number of camera coordinates
360     * @return number of camera coordinates
361     */
362     public int getNumberOfCameraCoordinates()
363     {
364         return this.cameraPositions.size();
365     }
366
367     /**
368      * Return the offset for the camera height to the water surface on the tray
369      * @return Returns camera height
370      */
371      public double getWaterSurfaceOffsetForCamera()
372     {
373         return waterSurfaceOffsetForCamera;
374     }
375
376      /**
377       * Return the number of dead roes in this tray
378       * @return Return how many dead roe in this tray
379       */
380       public int getNrOfDeadRoe() {
381         return nrOfDeadRoe;
382     }
383
384       /**
385        * Set dead roe found in this tray.
386        * @param nrOfDeadRoe
387        */
388     public void setNrOfDeadRoe(int nrOfDeadRoe) {
389         this.nrOfDeadRoe = nrOfDeadRoe;
390     }
391 }
```

```java

package RoeRobot;

import java.util.ArrayList;
import java.util.Iterator;

/**
 * Rack register class contains an list of trays in a rack.
 *
 * @author Yngve
 */
public class TrayRegister {

    private final ArrayList<Tray> trayRegister;

    public TrayRegister() {
        this.trayRegister = new ArrayList<>();
    }

    /**
     * Adds tray to the rack register
     *
     * @param tray
     */
    public void addToRegister(Tray tray) {
        this.trayRegister.add(tray);
    }

    /**
     * Get number of trays returns the number of trays in the rack.
     *
     * @return int representing number of trays in the rack.
     */
    public int getNumberOfTrays() {
        return this.trayRegister.size();
    }
    /**
     * Return register iterator
     * @return
     */
    public Iterator getRegisterIterator() {
        return this.trayRegister.iterator();
    }

    /**
     * Return the tray based on its Number
     * @param trayNr The tray number to the tray
     * @return Returns the tray with the given tray parameter, null if nothing found
     */
    public Tray getTray(int trayNr)
    {
        Tray returnTray = null;
        Iterator itr = (Iterator) this.getRegisterIterator();
        //Flag bool to exit loop
        boolean found = false;
        //Iterate over the trays
        while(itr.hasNext() && !found)
        {
            Tray itrTray = (Tray) itr.next();

            //If the tray nr parameter corresponds with the
            if(Integer.compare(itrTray.getTrayNr(), trayNr) == 0)
            {
                returnTray = itrTray;
                found = true;
            }
        }

        return returnTray;
    }
}

```

```java
package RoeRobot;

import ImageProcessing.*;

/**
 * Listener for processed trays
 * @author odroid
 */
public interface TrayProcessedListener
{

    public void noitfyProcessingDone(Tray workingTray);

}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package RoeRobot;



/**
 * Representate a coordinat in a global coodrinate system.
 *
 * @author Yngve
 */
public class Coordinate {

    private final double xCoord;
    private final double yCoord;
    private final double zCoord;



    public Coordinate(double xCoord, double yCoord, double zCoord) {
        this.xCoord = xCoord;
        this.yCoord = yCoord;
        this.zCoord = zCoord;

    }

    public Coordinate(double xCoord, double yCoord)
    {
        this.xCoord = xCoord;
        this.yCoord = yCoord;
        this.zCoord = -1;
    }

      public Coordinate(double zCoord)
    {
        this.xCoord = -1;
        this.yCoord = -1;
        this.zCoord = zCoord;
    }

    /**
     * Get x coordinat returns an x coordinat in a global coordinat system
     *
     * @return int representing a x coodrinat in a global coordinat system
     */
    public double getxCoord() {
        return this.xCoord;
    }

    /**
     * Get y coordinat returns an y coordinat in a global coordinat system
     *
     * @return int representing a y coodrinat in a global coordinat system
     */
    public double getyCoord() {
        return this.yCoord;
    }

    /**
     * Get z coordinat returns an z coordinat in a global coordinat system
     *
     * @return int representing a z coodrinat in a global coordinat system
     */
    public double getzCoord() {
        return this.zCoord;
    }


    @Override
     public String toString() {
```

```
74            String XYCoordString =  this.xCoord +  " " + this.yCoord;
75            return XYCoordString;
76        }
77
78    }
79
80
```

```java

package Commands;

/**
 * This class functions as superclass to all commandoes. All commandoes have
 * address and they can have value attached.
 *
 * @author PerEspen
 */
import java.nio.ByteBuffer;

public class Commando
{

    //Class fields
    private final byte commandAddress;
    //First input in value is the length of the byte[]
    private String[] value;
    //Default is 1 bytes
    private int nrOfBytes = 1;

    //Flag for what controller this command is designated
    public boolean forLinearRobot = true;
    public boolean forElevatorRobot = true;

    //Constructor
    public Commando(byte commandAddress)
    {
        this.commandAddress = commandAddress;
        //Creates value(byte[]) with default nr of bytes inside
        this.value = null;
    }

    /**
     * Set the value from int
     *
     * @param value The int to set as value
     */
    public void setIntValue(int value)
    {
        this.value = new String[1];
        this.value[0] = Integer.toUnsignedString(value);
    }

    /**
     * Set multiple int values
     *
     * @param value The number count for this value
     * @param nr The value to set
     */
    public void setMultipleIntValue(int value, int nr)
    {
        this.value[nr] = Integer.toUnsignedString(value);

    }

    /**
     * Return the string array value
     *
     * @return String array value
     */
    public String[] getValue()
    {
        return this.value;
    }

    /**
     * Set the string array value
     *
     * @param newValue The string array value to set
     */
    public void setValue(String[] newValue)
    {
```

```
74          this.value = newValue;
75      }
76
77      /**
78       * Returns the command address for this commando object
79       *
80       * @return Returns the command address for this commando in byte
81       */
82      public byte getCmdAddr()
83      {
84          return this.commandAddress;
85      }
86
87      /**
88       * Returns the command address for this commando object
89       *
90       * @return Returns the command address for this commando in String
91       */
92      public String getStringCmdAddr()
93      {
94
95          return Byte.toString(commandAddress);
96      }
97
98
99      /**
100      * Return if this is intended for linear bot
101      * @return Boolean which tells if it is meant for linear bot or not
102      */
103     public boolean isForLinearRobot()
104     {
105         return forLinearRobot;
106     }
107
108     /**
109      * Set the boolean for linear bot
110      * @param forLinearRobot The boolean to set
111      */
112     public void setForLinearRobot(boolean forLinearRobot)
113     {
114         this.forLinearRobot = forLinearRobot;
115     }
116
117     /**
118      * Return if this is intended for elevator bot
119      * @return Boolean which tells if it is meant for the elevator bot or not
120      */
121     public boolean isForElevatorRobot()
122     {
123         return this.forElevatorRobot;
124     }
125
126     /**
127      * Set the elevator bot boolean
128      * @param forElevatorRobot The boolean to set
129      */
130     public void setForElevatorRobot(boolean forElevatorRobot)
131     {
132         this.forElevatorRobot = forElevatorRobot;
133     }
134
135 }
136
```

```java
package Commands;

import Commands.Commando;

/**
 * Command to do calibration
 * @author PerEspen
 */
public class Calibrate extends Commando
{

    //The command address
    private static final byte COMMAND_ADDRESS = 0x10;

    public Calibrate( )
    {
        super(COMMAND_ADDRESS);
    }



}

```

```java

package Commands;

/**
 * Command to turn return the calibration parameters
 * @author PerEspen
 */
public class CalibParam extends Commando
{
    //The command address
    private static final byte COMMAND_ADDRESS = 0x31;


    public CalibParam( )
    {
        super(COMMAND_ADDRESS);
    }



}

```

```java

package Commands;

import Commands.Commando;

/**
 * Command to update Acceleration parameter
 *
 * @author Per Espen
 */
public class Acceleration extends Commando
{

    //The parameters
    private byte[] linearRobotAcclParam;
    private byte[] elevatorAcclParam;

    //The command address
    private static final byte COMMAND_ADDRESS = 0x21;

    public Acceleration()
    {
        super(COMMAND_ADDRESS);
        linearRobotAcclParam = null;
        elevatorAcclParam = null;
    }

    /**
     * Return the byte arr of the linear array
     *
     * @return Return the byte arr of the linear array
     */
    public byte[] getLinearRobotAcclParam()
    {
        return linearRobotAcclParam;
    }

    /**
     * Set the linear bot acceleration parameter
     *
     * @param linearRobotAcclParam The parameter to set
     */
    public void setLinearRobotAcclParam(byte[] linearRobotAcclParam)
    {
        this.linearRobotAcclParam = linearRobotAcclParam;
    }

    /**
     * Return the byte arr of the elevator array
     *
     * @return Return the byte arr of the elevator array
     */
    public byte[] getElevatorAcclParam()
    {
        return elevatorAcclParam;
    }

    /**
     * Set the Elevator bot acceleration parameter
     * @param elevatorAcclParam The parameter to set
     */
    public void setElevatorAcclParam(byte[] elevatorAcclParam)
    {
        this.elevatorAcclParam = elevatorAcclParam;
    }
}

```

```java
package Commands;

/**
 * Command to turn change RGB led values
 * @author PerEspen
 */
public class ChangeLedColor extends Commando
{
    //The command address
        private static final byte COMMAND_ADDRESS = 0x12;

    //The RGB value
    private String[] value;

    public ChangeLedColor( )
    {
        super(COMMAND_ADDRESS);
    }


      /**
       * Set the RGB values in a String[] and set the new String[] in the super value
       * @param red Red value
       * @param green Green value
       * @param blue Blue value
       */
    public void setMultipleIntValue(int red, int green, int blue)
  {
      //Create the string array for all the
      String[] value = new String[3];
      //Set the values to the string array in RGB format
     value[0] = Integer.toUnsignedString(red);
     value[1] = Integer.toUnsignedString(green);
     value[2] = Integer.toUnsignedString(blue);

     //Set the value
     super.setValue(value);
  }

}

```

```java

package Commands;

/**
 * Command to turn disco lights on
 * @author PerEspen
 */
public class DiscoLight extends Commando
{
        //The command address
        private static final byte COMMAND_ADDRESS = 0x13;

    public DiscoLight( )
    {
        super(COMMAND_ADDRESS);
        //Not meant for elevator controller
        super.setForElevatorRobot(false);
    }
}

```

```java

package Commands;

/**
 * Command to turn find tray
 * @author PerEspen
 */

public class FindTray extends Commando
{
    //The command address
        private static final byte COMMAND_ADDRESS = 0x14;

    public FindTray( )
    {
        super(COMMAND_ADDRESS);
        //Not meant for elevator controller
        super.setForElevatorRobot(false);
    }


}

```

```java

package Commands;

import Commands.Commando;

/**
 * Command to turn light on or off
 * @author PerEspen
 */
public class Light extends Commando
{
    //Payload in this command
     private byte[] value;
    //Command address
    private static final byte COMMAND_ADDRESS = 0x11;

    public Light( )
    {
        super(COMMAND_ADDRESS);
        //Not meant for elevator controller
        super.setForElevatorRobot(false);
    }



      public void setValue(byte[] value)
    {
        //The length of the given byte[]
        byte incSize = (byte) value.length;
        //Create big enough byte[] to store the inc []
        this.value = new byte[incSize];
        //Save the size of the byte in the first byte
        for(int i = 0; i< incSize; ++i)
        {
            this.value[i] = value[i];
        }
      //  this.setNrOfBytes(incSize);
        //Save the incomming byte[] value in the class value

    }


      /**
       * Set the byte to ON value
       */
      public void setOn()
      {
         byte[] controlByte = new byte[1];
        controlByte[0] = 1;

        this.setValue(controlByte);
      }

      /**
       * Set the byte to OFF value
       */
       public void setOff()
      {
         byte[] controlByte = new byte[1];
        controlByte[0] = 1;

        this.setValue(controlByte);
      }
}

```

```
1
2   package Commands;
3
4   /**
5    * Command to turn magnet off
6    * @author PerEspen
7    */
8   public class MagnetOff extends Commando
9   {
10       //The command address
11      private static final byte COMMAND_ADDRESS = 0x23;
12
13
14      public MagnetOff( )
15      {
16          super(COMMAND_ADDRESS);
17          //Not meant for elevator controller
18          super.setForElevatorRobot(false);
19      }
20
21  }
22
```

```java

package Commands;

/**
 * Command to turn magnet on
 * @author PerEspen
 */
public class MagnetOn extends Commando
{
     //The command address
    private static final byte COMMAND_ADDRESS = 0x22;


    public MagnetOn( )
    {
        super(COMMAND_ADDRESS);
        //Not meant for elevator controller
        super.setForElevatorRobot(false);
    }

}
```

```java

package Commands;

import Commands.Commando;
import java.nio.ByteBuffer;

/**
 * Move command to X, Y and Z position
 * @author PerEspen
 */
public class Move extends Commando
{
    //The command address
    private static final byte COMMAND_ADDRESS = 0x05;

    //The values for X, Z and Y movement
    private byte[] xValue;
    private byte[] yValue;
    private byte[] zValue;

    //The X,Y,Z move values in int
    private int xMove;
    private int yMove;
    private int zMove;

    //Bools tell which value this move contains
    private boolean xMoveBool;
    private boolean yMoveBool;
    private boolean zMoveBool;


    public Move()
    {
        super(COMMAND_ADDRESS);
        //Set all initial values to NULL
        xValue = null;
        yValue = null;
        zValue = null;

        //Initiate bools with false
        this.xMoveBool = false;
        this.yMoveBool = false;
        this.zMoveBool = false;
    }


    /**
     * Set the byte[] value with an int of 2 significant numbers
     *
     * @param intValue The int to set to value
     */
    public void setIntXValue(int intValue)
    {
        ByteBuffer dbuf = ByteBuffer.allocate(Integer.BYTES);

        dbuf.putInt(intValue);
        setxValue(dbuf.array()); // { 0, 1 }
    }

    /**
     * Set the byte[] value with an int of 2 significant numbers
     *
     * @param intValue The int to set to value
     */
    public void setIntYValue(int intValue)
    {
        ByteBuffer dbuf = ByteBuffer.allocate(Integer.BYTES);
        dbuf.putInt(intValue);
        setyValue(dbuf.array()); // { 0, 1 }
    }

    /**
     * Set the byte[] value with an int of 2 significant numbers
```

```
74          *
75          * @param intValue The int to set to value
76          */
77         public void setIntZValue(int intValue)
78         {
79             ByteBuffer dbuf = ByteBuffer.allocate(Integer.BYTES);
80             dbuf.putInt(intValue);
81             setzValue(dbuf.array()); // { 0, 1 }
82         }
83 
84         /**
85          * Returns byte[] value as int
86          *
87          * @return Returns byte[] value as int
88          */
89         public int getIntYValue()
90         {
91             byte[] arr = getyValue();
92             ByteBuffer wrapped = ByteBuffer.wrap(arr); // big-endian by default
93             int num = wrapped.getInt();// 1
94 
95             return num;
96         }
97 
98         /**
99          * Set the byte[] value with an int of 2 significant numbers
100          *
101          * @param intValue The int to set to value
102          */
103         public void setShortXValue(short intValue)
104         {
105             ByteBuffer dbuf = ByteBuffer.allocate(Short.BYTES);
106             dbuf.putShort(intValue);
107             setxValue(dbuf.array()); // { 0, 1 }
108         }
109 
110         /**
111          * Set the byte[] value with an int of 2 significant numbers
112          *
113          * @param shortValue to set to value
114          */
115         public void setShortYValue(short shortValue)
116         {
117             ByteBuffer dbuf = ByteBuffer.allocate(Short.BYTES);
118             dbuf.putShort(shortValue);
119             setyValue(dbuf.array()); // { 0, 1 }
120         }
121 
122         /**
123          * Set the byte[] value with an int of 2 significant numbers
124          *
125          * @param shortValue to set to value
126          */
127         public void setShortZValue(short shortValue)
128         {
129             ByteBuffer dbuf = ByteBuffer.allocate(Short.BYTES);
130             dbuf.putShort(shortValue);
131             setzValue(dbuf.array()); // { 0, 1 }
132         }
133 
134         /**
135          * Return the nr of bytes in the value
136          *
137          * @return Return the nr of bytes in value as int
138          */
139         public byte getNrOfBytesInByte()
140         {
141             int counter;
142             int sum = getxValue().length + getzValue().length + getyValue().length;
143 
144             return (byte) sum; // { 0, 1 }
145         }
146 
```

```
147         /**
148          * Return the XY payload of this command, with first byte as address for cmd
149          * and rest as X then Y position
150          *
151          * @return Return the payload including its size
152          */
153         public byte[] makeCompleteXYByte()
154         {
155             byte[] returnByte = null;
156 
157             //To count where in the byte array next pos is
158             int arrayCounter = 0;
159             //Extra length for the array, without x and y length
160             int extraLength = 1;
161             //Check for null
162             if (this.getxValue() != null)
163             {
164                 //Create new byte array for added size to the value
165                 returnByte = new byte[this.getxValue().length + this.getyValue().length
                    + extraLength];
166 
167                 //Make new byte to send to store the byte[] length in the first byte
168                 returnByte[arrayCounter++] = this.getCmdAddr();
169                 //returnByte[arrayCounter++] = (byte) ((byte) this.getxValue().length +
                    this.getyValue().length);
170                 //Add the X and Y value positions
171                 System.arraycopy(this.getxValue(), 0, returnByte, arrayCounter,
                    this.getxValue().length);
172                 //increment arrayCounter
173                 arrayCounter = arrayCounter + this.getxValue().length;
174                 System.arraycopy(this.getyValue(), 0, returnByte, arrayCounter,
                    this.getyValue().length);
175             }
176 
177             return returnByte;
178         }
179 
180 
181         /**
182          * Return the XY payload of this command, with first byte as address for cmd
183          * and second as total number of bytes
184          *
185          * @return Return the payload including its size
186          */
187         public byte[] makeCompleteZByte()
188         {
189             byte[] returnByte = null;
190 
191             //To count where in the byte array next pos is
192             int arrayCounter = 0;
193             //Check for null
194             if (this.getzValue() != null)
195             {
196                 //Create new byte array for added size to the value
197                 returnByte = new byte[this.getzValue().length + 1];
198 
199                 //Make new byte to send to store the byte[] length in the first byte
200                 returnByte[arrayCounter++] = this.getCmdAddr();
201                 // returnByte[1] = (byte) this.getzValue().length;
202                 //Add the z value
203                 System.arraycopy(this.getzValue(), 0, returnByte, arrayCounter,
                    this.getzValue().length);
204 
205                 arrayCounter = arrayCounter + this.getzValue().length;
206             }
207 
208             return returnByte;
209         }
210 
211 
212 
213         /**
214          * Return the X move value
```

```
215         * @return Return the X move value
216         */
217        public int getxMove()
218        {
219            return this.xMove;
220        }
221        /**
222         * Set the X move value
223         * @param xMove Set the X move value from double
224         */
225        public void setxMove(double xMove)
226        {
227            this.xMove = (int) Math.round(xMove);
228        }
229
230        /**
231         * Return the Y move value
232         * @return Return the Y move value
233         */
234        public int getyMove()
235        {
236            return this.yMove;
237        }
238
239        /**
240         * Set the Y move value
241         * @param yMove Set the Y move value from double
242         */
243        public void setyMove(double yMove)
244        {
245            this.yMove = (int) Math.round(yMove);
246        }
247
248        /**
249         * Return the Z move value
250         * @return Return the Z move value
251         */
252        public int getzMove()
253        {
254            return zMove;
255        }
256
257        /**
258         * Set the Z move value
259         * @param zMove Set the Z move value from double
260         */
261        public void setzMove(double zMove)
262        {
263            this.zMove = (int) Math.round(zMove);
264        }
265
266
267            /**
268         * Return the if this command contains should send X value
269         * @return
270         */
271        public boolean isxMoveBool()
272        {
273            return xMoveBool;
274        }
275
276        /**
277         * Set the X move bool
278         * @param xMoveBool The bool to be set
279         */
280        public void setxMoveBool(boolean xMoveBool)
281        {
282            this.xMoveBool = xMoveBool;
283        }
284
285        /**
286         * Return the if this command contains should send Y value
287         * @return
```

```
288         */
289        public boolean isyMoveBool()
290        {
291            return yMoveBool;
292        }
293        /**
294         * Set the Y move bool
295         * @param yMoveBool The bool to be set
296         */
297        public void setyMoveBool(boolean yMoveBool)
298        {
299            this.yMoveBool = yMoveBool;
300        }
301
302        /**
303         * Return the if this command contains should send Z value
304         * @return
305         */
306        public boolean iszMoveBool()
307        {
308            return zMoveBool;
309        }
310
311        /**
312         * Set the Z move bool
313         * @param zMoveBool The bool to be set
314         */
315        public void setzMoveBool(boolean zMoveBool)
316        {
317            this.zMoveBool = zMoveBool;
318        }
319
320
321          /**
322         * Return the X value byte arrau
323         * @return Return the X value byte array
324         */
325        public byte[] getxValue()
326        {
327            return xValue;
328        }
329        /**
330         * Set the X value byte array
331         * @param xValue Set the X value byte array
332         */
333        public void setxValue(byte[] xValue)
334        {
335            this.xValue = xValue;
336        }
337
338        /**
339         * Return the Y value byte array
340         * @return Return the Y value byte array
341         */
342        public byte[] getyValue()
343        {
344            return yValue;
345        }
346
347        /**
348         * Set the Y byte array value
349         * @param yValue The byte array to set
350         */
351        public void setyValue(byte[] yValue)
352        {
353            this.yValue = yValue;
354        }
355
356        /**
357         * Return the Y value byte array
358         * @return Return the Y value byte array
359         */
360        public byte[] getzValue()
```

```
361         {
362             return zValue;
363         }
364 
365         /**
366          * Set the Z byte array value
367          * @param zValue The byte array to set
368          */
369         public void setzValue(byte[] zValue)
370         {
371             this.zValue = zValue;
372         }
373     }
374 
```

```java
package Commands;

import Commands.Commando;

/**
 * Command to turn return the current state
 * @author PerEspen
 */
public class StateRequest extends Commando
{
    private boolean elevatorRobot = false;
    private boolean linearRobot = false;

     private static final byte COMMAND_ADDRESS = 0x30;

    public StateRequest()
    {
        super(COMMAND_ADDRESS);
    }

    public boolean forElevatorRobot()
    {
        return elevatorRobot;
    }

    public void setElevatorRobot(boolean elevatorRobot)
    {
        this.elevatorRobot = elevatorRobot;
    }

    public boolean forLinearRobot()
    {
        return linearRobot;
    }

    public void setLinearRobot(boolean linearRobot)
    {
        this.linearRobot = linearRobot;
    }

    public void reset()
    {
        this.elevatorRobot = false;
        this.linearRobot = false;
    }

}

```

```java

package Commands;

/**
 * Command to turn STOP
 * @author PerEspen
 */
public class Stop extends Commando
{
    //The commando address
      private static final byte COMMAND_ADDRESS = 0x07;

    public Stop()
    {
        super(COMMAND_ADDRESS);
    }
}

```

```java

package Commands;

import Commands.Commando;

/**
 * Command to turn to perform suction
 * @author PerEspen
 */
public class Suction extends Commando
{
    //The command address
     private static final byte COMMAND_ADDRESS = 0x06;

    public Suction()
    {
        super(COMMAND_ADDRESS);
    }

}

```

```java

package Commands;

import Commands.Commando;

/**
 * Commando to change the Velocity parameter of the Arduino's
 * @author PerEspen
 */
public class Velocity extends Commando
{
    //The command address
     private static final byte COMMAND_ADDRESS = 0x20;

    public Velocity()
    {
        super(COMMAND_ADDRESS);
    }

}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

/**
 * T
 * @author PerEspen
 */
public class Busy extends Status
{
      //Status name for this class
    private static final String STATUS = "BUSY";
    //Status address
    private static final byte STATUS_ADDRESS = 0x50;

    /**
     *
     */
    public Busy()
    {
        super(STATUS_ADDRESS, STATUS);
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;
/**
 *
 * @author PerEspen
 */
public class ElevatorLimitTrigg extends Status
{
      //Status name for this class
    private static final String STATUS = "ELEVATOR_LIMIT_TRIGG";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x63;

    public ElevatorLimitTrigg(  )
    {
        super(STATUS_ADDRESS, STATUS);

        // set critical value to true
        super.setCritical(true);
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

import StatusListener.StatusListener;

/**
 *
 * @author PerEspen
 */
public class EMC extends Status
{
      //Status name for this class
    private static final String STATUS = "EMC";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x60;

    public EMC( )
    {
        super(STATUS_ADDRESS, STATUS);

        // set critical value to true
        super.setCritical(true);
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;


/**
 *
 * @author PerEspen
 */
public class EncoderOutOfRange extends Status
{
      //Status name for this class
    private static final String STATUS = "ENCODER_OUT_OF_RANGE";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x66;

    public EncoderOutOfRange(  )
    {
        super(STATUS_ADDRESS, STATUS);
    }


      @Override
     public boolean critical()
     {
         return true;
     }

}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;



/**
 *
 * @author PerEspen
 */
public class EncoderOutOfSync extends Status
{
      //Status name for this class
    private static final String STATUS = "ENCODER_OUT_OF_SYNC";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x65;

    public EncoderOutOfSync( )
    {
        super(STATUS_ADDRESS , STATUS);
    }


      @Override
     public boolean critical()
     {
         return true;
     }

}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;


/**
 *
 * @author PerEspen
 */
public class Failure extends Status
{
      //Status name for this class
    private static final String STATUS = "FAILURE";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x67;

    public Failure()
    {
        super(STATUS_ADDRESS, STATUS);
    }


      @Override
     public boolean critical()
     {
         return true;
     }

}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

/**
 *
 * @author PerEspen
 */
public class LinearBotLimitTrigged extends Status
{
      //Status name for this class
    private static final String STATUS = "LINEAR_BOT_LIMIT_TRIGGED";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x64;

    public LinearBotLimitTrigged( )
    {
        super(STATUS_ADDRESS , STATUS);

        // set critical value to true
        super.setCritical(true);
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

import StatusListener.StatusListener;
import java.nio.ByteBuffer;
import java.util.Iterator;
import RoeRobot.Tray;
import RoeRobot.TrayRegister;

/**
 *
 * @author PerEspen
 */
public class Parameters extends Status
{

    //Status name for this class
    private static final String STATUS = "PARAMETERS";

    //ADDRESS For this status command
    private static final byte STATUS_ADDRESS = 0x70;

    private static final byte DEFAULT_BYTE_RANGE = Integer.BYTES;

    /**
     * PARAMETERS*
     */

    //Calibrated values
    private int xCalibRange = 0;
    private int yCalibRange = 0;
    private int zCalibRange = 0;
    private int numberOfTrays = 0;
    //Tray reg to put the trays in
    private TrayRegister trayReg;


    //Bools to track which calibs are updated etc
    private boolean linearCalib = false;
    private boolean elevatorCalib = false;
    private boolean send = false;

    public Parameters()
    {
        //Put superclass params
        super(STATUS_ADDRESS, STATUS);

        //Create tray reg
        trayReg = new TrayRegister();
        //Initiate the values
        linearCalib = false;
        elevatorCalib = false;
        send = false;
    }









    public static byte getCMD()
    {
        return STATUS_ADDRESS;
    }

    /*
    @Override
    public void putValue(byte[] inputVal)
    {
```

```
74          ByteBuffer bb;
75          int lenghtCnt = 0;
76          System.out.println(lenghtCnt);
77
78
79          //Checks if there are multiple values, multiple values means its both x and
            y, maybe all
80          if (inputVal.length == DEFAULT_BYTE_RANGE * 2)
81          {
82              //Copying and setting the X byte[]//
83
84              byte[] copy = new byte[DEFAULT_BYTE_RANGE];
85              System.arraycopy(inputVal, 0, copy, 0, DEFAULT_BYTE_RANGE);
86              this.setXByteArr(copy);
87              lenghtCnt = lenghtCnt + DEFAULT_BYTE_RANGE;
88
89              //Copying and setting the Y byte[]/
90              copy = new byte[DEFAULT_BYTE_RANGE];
91              System.arraycopy(inputVal, DEFAULT_BYTE_RANGE, copy, 0,
                DEFAULT_BYTE_RANGE);
92              this.setYByteArr(copy);
93              lenghtCnt = lenghtCnt + DEFAULT_BYTE_RANGE;
94
95
96          }
97             //Check if there are enough values to be Z and TRAYS value
98         else if (inputVal.length >= DEFAULT_BYTE_RANGE * 4)
99          {
100
101                 //Copying and setting the Y byte[]/
102                 byte[] copy = new byte[DEFAULT_BYTE_RANGE];
103                 System.arraycopy(inputVal, 0, copy, 0, DEFAULT_BYTE_RANGE);
104                 this.setZByteArr(copy);
105                 lenghtCnt = lenghtCnt + DEFAULT_BYTE_RANGE;
106
107                  //Copying and setting the Y byte[]//
108                  copy = new byte[DEFAULT_BYTE_RANGE];
109                     System.arraycopy(inputVal, DEFAULT_BYTE_RANGE, copy, 0,
                    DEFAULT_BYTE_RANGE);
110                  this.setYByteArr(copy);
111                  lenghtCnt = lenghtCnt + DEFAULT_BYTE_RANGE;
112
113         }
114         //Only 1 value means its only z range
115         else if (inputVal.length == DEFAULT_BYTE_RANGE)
116         {
117
118             /*Copying and setting the X byte[]//
119             byte[] copy = new byte[inputVal.length];
120             System.arraycopy(inputVal, 0, copy, 0, DEFAULT_BYTE_RANGE);
121             this.setZByteArr(copy);
122             lenghtCnt = lenghtCnt + DEFAULT_BYTE_RANGE;
123         }
124     }
125      */
126
127     /**
128      * Put the value
129      *
130      * @param inputVal
131      */
132     @Override
133     public void putValue(String[] inputVal)
134     {
135         int inputCnt = 0;
136
137         /**
138          * Check length on input value to decide what values to store*
139          */
140         //Bigger or equals to 3 means its Z coord +
141         if (inputVal.length >= 3 || inputVal.length == 1)
142         {
143            // System.out.println("if(inputVal.length >= 3)");
```

```
144
145
146                 //Save the Z position
147                 this.setzCalibRange(Integer.parseUnsignedInt(inputVal[inputCnt++]));
148
149                // System.out.println("this.setzCalibRange");
150                 int trayCounter = 0;
151                 this.setNumberOfTrays(numberOfTrays);
152
153                 //Iterate through all the values and create a tray
154                 for (int i = inputCnt; i < inputVal.length; ++i)
155                 {
156                     ++trayCounter;
157                    // System.out.println(" trayReg.addToRegister(tempTray);");
158                     Tray tempTray = new Tray(trayCounter,
                        Integer.parseUnsignedInt(inputVal[i]));
159                     trayReg.addToRegister(tempTray);
160                 }
161                 this.setNumberOfTrays(trayCounter);
162
163                 this.setElevatorCalib(true);
164             }
165             //If the value length is 2 that means it is calibrated X and Y values
166             if (inputVal.length == 2)
167             {
168                // System.out.println("if(inputVal.length == 2)");
169
170
171                 this.setxCalibRange(Integer.parseUnsignedInt(inputVal[inputCnt++]));
172                 this.setyCalibRange(Integer.parseUnsignedInt(inputVal[inputCnt]));
173
174                 this.setLinearCalib(true);
175                // System.out.println("setyCalibRange-setxCalibRange DONE");
176             }
177
178             //System.out.println("Finished put values");
179         }
180
181         /**
182          * Update all the trays with it's default coord values
183          */
184         public void updateTrays()
185         {
186             for (Iterator<Tray> iterator = trayReg.getRegisterIterator();
                iterator.hasNext();)
187             {
188                 Tray next = iterator.next();
189                next.createTrayCoords(this.getxCalibRange(), this.getyCalibRange());
190             }
191         }
192
193
194
195
196
197         public int getxCalibRange()
198         {
199             return this.xCalibRange;
200         }
201
202         public void setxCalibRange(int xCalibRange)
203         {
204             this.xCalibRange = xCalibRange;
205         }
206
207         public int getyCalibRange()
208         {
209             return this.yCalibRange;
210         }
211
212         public void setyCalibRange(int yCalibRange)
213         {
214             this.yCalibRange = yCalibRange;
```

```
215         }
216
217         public int getzCalibRange()
218         {
219             return zCalibRange;
220         }
221
222         public void setzCalibRange(int zCalibRange)
223         {
224             this.zCalibRange = zCalibRange;
225         }
226
227         public int getNumberOfTrays()
228         {
229             return numberOfTrays;
230         }
231
232         private void setNumberOfTrays(int numberOfTrays)
233         {
234             this.numberOfTrays = numberOfTrays;
235         }
236
237         public boolean isLinearCalib()
238         {
239             return linearCalib;
240         }
241
242         private void setLinearCalib(boolean linearCalib)
243         {
244             this.linearCalib = linearCalib;
245         }
246
247         public boolean isElevatorCalib()
248         {
249             return elevatorCalib;
250         }
251
252         private void setElevatorCalib(boolean elevatorCalib)
253         {
254             this.elevatorCalib = elevatorCalib;
255         }
256
257         public boolean isSend()
258         {
259             return send;
260         }
261
262         public void setSend(boolean send)
263         {
264             this.send = send;
265         }
266
267
268           public TrayRegister getTrayReg()
269         {
270             return trayReg;
271         }
272
273
274     }
275
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

/**
 *
 * @author PerEspen
 */
public class ReadyToRecieve extends Status
{
    //Status name for this class
    private static final String STATUS = "READY";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x51;

        public ReadyToRecieve( )
    {
        super(STATUS_ADDRESS , STATUS);
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;



/**
 *
 * @author PerEspen
 */
public class SafetySwitchLower extends Status
{

    //Status name for this class
    private static final String STATUS = "SAFETY_SWITCH_LOWER";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x62;

    public SafetySwitchLower( )
    {
        super(STATUS_ADDRESS, STATUS);

        // set critical value to true
        super.setCritical(true);
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

/**
 *
 * @author PerEspen
 */
public class SafetySwitchUpper extends Status
{
      //Status name for this class
    private static final String STATUS = "SAFETY_SWITCH_UPPER";
    //Address for this status
    private static final byte STATUS_ADDRESS = 0x61;

    public SafetySwitchUpper()
    {
        super(STATUS_ADDRESS, STATUS);

        // set critical value to true
        super.setCritical(true);
    }



}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

import StatusListener.StatusListener;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Status message sent from arduinos. Each object holds unique address.
 *
 * @author PerEspen
 */
public class Status
{

    // list holding listeners
    ArrayList<StatusListener> listeners;

    //Address for the status
    private final byte StatusAddress;

    //Bool to tell if its updated to the system or not
    private boolean sent = false;

    //Number of bytes if other message then address is carried
    private int nrOfBytes;

    //flag to check if status is critical
    private boolean critical = false;

    //The payload attached to this status
    private String[] value;

    //private boolean triggered;
    private final String STATUS;

    public Status(byte statusAddr, String name)
    {
        //Create the listeners array
        this.listeners = new ArrayList();
        //Set the status address
        this.StatusAddress = statusAddr;
        //Set the status name
        this.STATUS = name;
    }

    /**
     * Return the status address
     *
     * @return Return the status address
     */
    public byte getStatusAddress()
    {
        return StatusAddress;
    }

    /**
     * Return the number of bytes in the payload of this status
     *
     * @return Return the number of bytes in the payload of this status
     */
    public int getNrOfBytes()
    {
        return nrOfBytes;
    }

    //TODO: OVERRIDE AND ADD IN THE CALIB PARAM.
    /**
```

```
74          * Put the byte values where they are supposed to be. Should be overided in
75          * classes with multiple byte storage instead of only trigger bool
76          *
77          * @param val The given byte value
78          */
79         public void putValue(String[] val)
80         {
81             this.value = val;
82         }
83 
84         /**
85          * Return the value string array attached to this status
86          *
87          * @return The value string array
88          */
89         public String[] getValue()
90         {
91             return this.value;
92         }
93 
94         /**
95          * Return the string name of this status
96          *
97          * @return The string name of this status
98          */
99         public String getString()
100        {
101            return this.STATUS;
102        }
103 
104        /**
105         * Change the state of the critical variable
106         *
107         * @param critical variables new state
108         */
109        protected void setCritical(boolean critical)
110        {
111            this.critical = critical;
112        }
113 
114        /**
115         * Return a new Status instance of the object calling it
116         *
117         * @return Return a new instance of this Status - used by sub-classes
118         */
119        public Status returnNew()
120        {
121            Status returnstat = null;
122            try
123            {
124                returnstat = this.getClass().newInstance();
125            } catch (InstantiationException ex)
126            {
127                Logger.getLogger(Status.class.getName()).log(Level.SEVERE, null, ex);
128            } catch (IllegalAccessException ex)
129            {
130                Logger.getLogger(Status.class.getName()).log(Level.SEVERE, null, ex);
131            }
132 
133            return returnstat;
134        }
135 
136        /**
137         * Returns true if this status is considered as critical for function
138         *
139         * @return value of critical flag
140         */
141        public boolean critical()
142        {
143            return this.critical;
144        }
145 
146        /**
```

```
147         * Add listener to listener list
148         *
149         * @param listener to be added to list
150         */
151        public void addListener(StatusListener listener)
152        {
153            this.listeners.add(listener);
154        }
155
156        /**
157         * Notify listeners on new status
158         */
159        /**
160         * Notify listeners on busy
161         */
162        public void notifyListeners()
163        {
164            if (this.listeners != null)
165            {
166                for (StatusListener listener : listeners)
167                {
168                    listener.notifyNewStatus(this);
169                }
170            }
171        }
172
173        /**
174         * Return the is sent bool for this status
175         *
176         * @return The is sent bool for this status
177         */
178        public boolean isSent()
179        {
180            return sent;
181        }
182
183        /**
184         * Set the sent bool for this status
185         *
186         * @param sent The bool to be set
187         */
188        public void setSent(boolean sent)
189        {
190            this.sent = sent;
191        }
192    }
193
```

```java
/*
 * This class is the listener interface for the status classes.
 * It holds methods that can be called on all other classes implementing this
 * interface.
 */
package StatusListener;

import Status.Status;

/**
 *
 * @author KristianAndreLilleindset
 */
public interface StatusListener
{
    /**
     * Notify new status
     * @param status being triggered
     */
    public void notifyNewStatus(Status status);
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Status;

/**
 *
 * @author PerEspen
 */
public class Stopped extends Status
{
    //Status name for this class
    private static final String STATUS = "STOPPED";
    //The status address
    private static final byte STATUS_ADDRESS = 0x52;

    public Stopped( )
    {
        super(STATUS_ADDRESS, STATUS);
    }
}
```

H.3 Serial Communication

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package SerialCommunication;

/*
 * This class is responsible for creating a serial connection,
 * when a connection has been established a reader and a writer object
 * is created.
 */
import Commands.Commando;
import Commands.MagnetOn;
import Commands.Move;
import Status.Busy;
import Status.EMC;
import Status.ElevatorLimitTrigg;
import Status.EncoderOutOfRange;
import Status.EncoderOutOfSync;
import Status.Failure;
import Status.LinearBotLimitTrigged;
import Status.Parameters;
import Status.ReadyToRecieve;
import Status.SafetySwitchLower;
import Status.SafetySwitchUpper;
import Status.Status;
import Status.Stopped;
import StatusListener.StatusListener;
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * This class handles the opens and handles the Serial communication.
 * Commandos are added to the qeue, then sent to the desired controllers
 * By InputListener it parses incoming messages and sets the corresponding status
 * Notifies STATUS listeners if the status is new
 *
 * @author Per Espen Aarseth
 */
public class SerialCommunication extends Thread implements SerialInputListener
{


    // ********************* SERIAL VARIABLES *********************
    //Connected devices to serial get following Device-ID
    private static final String CONTROLLER_COM_ADDR_ELEVATOR = "ttyACM0";
    private static final String CONTROLLER_COM_ADDR_LINEARBOT = "ttyUSB0";
    //Device name
    private static final String CONTROLLER_STRADDR_ELEVATOR = "dev2";
    private static final String CONTROLLER_STRADDR_LINEARBOT = "dev1";



    //Commports to the controllers
    SerialJComm linearBot;
    SerialJComm elevatorBot;

    // Flag for incomming data and storage
    boolean newDataRecieved = false;
    String[] incommingData = null;
    byte[] incommingByteData = null;

    // Boolean for awaiting ack from controllers after sent
    boolean linearBotAwaitingACK = false;
    boolean elevatorBotAwaitingACK = false;

```

```
74
75
76          // ************************ COMMAND/STATUS ****************************
77
78      //Lists to keep incomming demands in queue
79      LinkedList<Commando> sendQeue;
80
81      HashMap<Byte, Status> statusMap;
82      ArrayList<Byte> statusList;
83
84      //Statuses
85      Status elevatorState;
86      Status linearBotState;
87      Parameters calibrationParams;
88
89      //Only for testing
90      boolean readyTriggered = false;
91
92
93      // list holding the classes listening to the statuses
94      private ArrayList<StatusListener> listenerList;
95
96      /**
97       * Constructor
98       */
99      public SerialCommunication()
100     {
101         //Create the send qeue
102         sendQeue = new LinkedList<Commando>();
103
104         //Open the serial ports to the given port id
105         linearBot = new SerialJComm(CONTROLLER_COM_ADDR_LINEARBOT);
106         elevatorBot = new SerialJComm(CONTROLLER_COM_ADDR_ELEVATOR);
107
108         //create the listener list
109         listenerList = new ArrayList<StatusListener>();
110
111         //Add the listeners
112         linearBot.addListener(this);
113         elevatorBot.addListener(this);
114
115         //Create the calib param status for checks
116         calibrationParams = new Parameters();
117
118     }
119
120     /**
121      * ****************** SERIAL SETUP/FUNCTIONS ************************
122      */
123     /**
124      * Method creating a connection with a serialport if one is found.
125      */
126     public synchronized void connect()
127     {
128         //Connect to the serial devices
129         elevatorBot.connect();
130         linearBot.connect();
131
132         //Start the threads
133         linearBot.start();
134         elevatorBot.start();
135     }
136
137     /**
138      * Method closing the connection with the serialport.
139      */
140     public synchronized void close()
141     {
142         // check if there is a instance of a serialport
143         if (this.elevatorBot != null)
144         {
145             elevatorBot.close();
146         }
```

```
147             // check if there is a instance of a serialport
148             if (this.linearBot != null)
149             {
150                 linearBot.close();
151             }
152 
153         }
154 
155 
156 
157         //Serial data listener function
158         @Override
159         public synchronized void serialDataAvailable(byte[] data)
160         {
161 
162             //Set the new data bool to true
163             newDataRecieved = true;
164 
165             incommingData = null;
166             //Save the incomming data
167             incommingData = fromByteToStringArr(data);
168 
169             //Print the incomming data as a string
170             String dataString = new String(data, StandardCharsets.UTF_8);
171 
172             //Check and parse data
173             if (!checkAckAndToggle(incommingData))
174             {
175                 //Parse the newly recieved data
176                 parseInputData(incommingData);
177             }
178 
179         }
180 
181         //Serial data listener function
182         @Override
183         public synchronized void serialDataAvailable(String[] data)
184         {
185             //Data on serial port has arrived, set data storage to null to "reset"
186             //the data
187             incommingData = null;
188             //Save the incomming data
189             incommingData = data;
190 
191 
192                 //Parse the newly recieved data
193                 parseInputData(incommingData);
194                 //Set the new data bool to true
195                 this.newDataRecieved = true;
196         }
197 
198 
199         /**
200          * ENUM to hold all the addresses connected to the incomming states of the
201          * arduinos
202          */
203         private enum State
204         {
205             Busy(new Busy()),
206             ReadyToRecieve(new ReadyToRecieve()),
207             Stopped(new Stopped()),
208             EMC(new EMC()),
209             SAFETY_SWITCH_UPPER(new SafetySwitchUpper()),
210             SAFETY_SWITCH_LOWER(new SafetySwitchLower()),
211             ELEV_LIMIT_TRIGG(new ElevatorLimitTrigg()),
212             LINEARBOT_LMIT_TRIGG(new LinearBotLimitTrigged()),
213             ENCODER_OUT_OF_SYNC(new EncoderOutOfSync()),
214             ENCODER_OUT_OF_RANGE(new EncoderOutOfRange()),
215             PARAMETER(new Parameters()),
216             Failure(new Failure());
217 
218 
219 
```

```
220             //Hashmap for lookup
221             private static final HashMap<Byte, State> lookup = new HashMap<Byte, State>();
222 
223             //Put the states with the accompanied value in the hashmap
224             static
225             {
226                 //Create reverse lookup hash map
227                 for (State s : State.values())
228                 {
229                     lookup.put(s.getStateValue(), s);
230                 }
231             }
232 
233             //The status
234             private Status status;
235             //
236             private State(Status status)
237             {
238                 this.status = status;
239             }
240 
241             //Return the status address
242             public byte getStateValue()
243             {
244                 return status.getStatusAddress();
245             }
246 
247             //Lookup the address to see if the status exists in the list
248             public static State get(byte address)
249             {
250                 //the reverse lookup by simply getting
251                 //the value from the lookup HsahMap.
252                 return lookup.get(address);
253             }
254 
255 
256             //Return the status
257             public Status getStatus()
258             {
259                 return this.status;
260             }
261 
262         }
263 
264         /**
265          * *************THE LOOP****************
266          */
267         @Override
268         public void run()
269         {
270             //Bool to keep the loop going
271             boolean running = true;
272 
273             //Loop to keep the thread alive
274             while (running)
275             {
276 
277                 if (getSendQSize() != 0)
278                 {    //Send the commands in the qeue
279                     sendCommand(popSendQ());
280                 }
281 
282                 //New data is recieved
283                 //this bool is set by the listener
284                 if (newDataRecieved)
285                 {
286                     //Check the new statuses and trigger if needed
287                     if (checkStatesAndTrigger(this.elevatorState, this.linearBotState))
288                     {
289                         //Reset flag, as data have been parsed
290                         newDataRecieved = false;
291                     }
292                 }
```

```
293             }
294         }
295 
296 
297         /**
298          * Synchronized method for returning queue size
299          *
300          * @return Return the size of the queue
301          */
302         private synchronized int getSendQSize()
303         {
304             return sendQeue.size();
305         }
306 
307 
308         /**
309          * Synchronized method for returning last element of queue
310          *
311          * @return Returns the last commando put into the queue
312          */
313         private synchronized Commando popSendQ()
314         {
315             return sendQeue.pop();
316         }
317 
318 
319         /**
320          * Send the data in string to both controllers
321          *
322          * @param sendString String to send
323          */
324         private void writeString(String sendString)
325         {
326             try
327             {
328                 this.linearBot.sendData(sendString.getBytes("UTF-8"));
329                 this.elevatorBot.sendData(sendString.getBytes("UTF-8"));
330             } catch (UnsupportedEncodingException ex)
331             {
332                 Logger.getLogger(SerialCommunication.class.getName()).log(Level.SEVERE,
                    null, ex);
333             }
334         }
335 
336 
337         /**
338          * Send the data in string to linear serial port
339          *
340          * @param sendString
341          */
342         private synchronized void writeStringLinear(String sendString)
343         {
344              this.linearBot.setDataToBeSent(sendString);
345         }
346 
347 
348         /**
349          * Send the data in to the string elevator port
350          *
351          * @param sendString
352          */
353         private synchronized void writeStringElevator(String sendString)
354         {
355              this.elevatorBot.setDataToBeSent(sendString);
356         }
357 
358 
359         /**
360          * Send the data in byte[] to both controllers
361          *
362          * @param sendString byte arr to send
363          */
364         private void writeBytes(byte[] sendByte)
```

```
365         {
366             this.elevatorBot.sendData(sendByte);
367             this.linearBot.sendData(sendByte);
368         }
369 
370 
371         /**
372          * Send the data in byte[] to elevator controller
373          *
374          * @param sendString byte arr to send
375          */
376         private void writeBytesElevator(byte[] sendByte)
377         {
378             this.elevatorBot.sendData(sendByte);
379         }
380 
381 
382 
383         /**
384          * Send the data in byte[] to linear controller
385          *
386          * @param sendString byte arr to send
387          */
388         private void writeBytesLinear(byte[] sendByte)
389         {
390             this.linearBot.sendData(sendByte);
391         }
392 
393         /**
394          * Return a string containing both dev-address and cmd-address
395          *
396          * @param stringAddress Device address
397          * @param cmdByte The cmd-address Byte
398          * @return Return a string with both dev-address and cmd address seperated
399          * with ", "
400          */
401         String makeCMDString(String stringDevAddress, byte cmdByte)
402         {
403             String returnString = null;
404             String cmdString = Byte.toString(cmdByte);
405             returnString = stringDevAddress + ", " + cmdString;
406 
407             return returnString;
408         }
409 
410 
411         /**
412          * Check the incomming data for ACK or NACK and set the appropriate bools
413          *
414          * @param incommingData The incomming sring[] data
415          * @return Return true if incomming data was NACK or ACK
416          */
417         private boolean checkAckAndToggle(String[] incommingData)
418         {
419             boolean returnBool = false;
420 
421             //Check for null
422             if (incommingData != null)
423             {
424                 //Save address and feedback(ACK or NACK)
425                 String addr = incommingData[0];
426                 String feedback = null;
427                 //Check for address
428                 if (addr.compareTo(CONTROLLER_STRADDR_LINEARBOT) == 0)
429                 {
430                     //Check for length
431                     if (incommingData.length > 1)
432                     {
433 
434                         //Save the data from incdata
435                         feedback = incommingData[1];
436                         //CHECK FOR ACK
437                         if (feedback.compareToIgnoreCase("1") == 0)
```

```
                    {
                        //Update bools
                        returnBool = true;
                        linearBotAwaitingACK = false;
                    } //Check for NACK
                    else if (feedback.compareToIgnoreCase("0") == 0)
                    {
                        returnBool = true;
                        linearBotAwaitingACK = false;
                    }
                }
            } else if (incommingData[0].compareTo(CONTROLLER_STRADDR_ELEVATOR) == 0)
            {
                //Check for length
                if (incommingData.length > 1)
                {
                    //Save the data from incdata
                    feedback = incommingData[1];
                    //CHECK FOR ACK
                    if (feedback.compareToIgnoreCase("1") == 0)
                    {
                        //Update bools
                        returnBool = true;
                        elevatorBotAwaitingACK = false;
                    } //Check for NACK
                    else if (feedback.compareToIgnoreCase("0") == 0)
                    {
                        elevatorBotAwaitingACK = false;
                        returnBool = true;
                    }
                }
            }
        }

        return returnBool;
    }


    /**
     * Returns a string array from an byte array
     *
     * @param byteArr Byte array to make string array from
     * @return Returns a string array from an byte array param
     */
    private String[] fromByteToStringArr(byte[] byteArr)
    {
        String newString = null;
        String[] arrString = null;
        try
        {
            newString = new String(byteArr, "UTF-8");
            arrString = newString.split(",");
        } catch (UnsupportedEncodingException ex)
        {
            Logger.getLogger(SerialCommunication.class.getName()).log(Level.SEVERE,
            null, ex);
        }
        return arrString;
    }



    /**
     * Parses the string array in the incomming data parameter. Makes the
     * appropriate
     *
     * @param incommingData The data to parse
     * @return Returns true if new data was parsed
     */
    private void parseInputData(String[] incommingData)
    {
        boolean newData = false;

```

```
510             //Check for nullpointer
511             if (incommingData != null)
512             {
513                 int arrCnt = 0;
514 
515                 //Save the device address
516                 String addrStr = incommingData[arrCnt++];
517 
518                 //Create new value str[]
519                 String[] valueStr = new String[incommingData.length - 1];
520 
521                 //Copy the values
522                 for (int i = arrCnt; i < incommingData.length; ++i)
523                 {
524                     valueStr[i - arrCnt] = incommingData[i];
525                 }
526 
527                 //Check for linear address
528                 if (addrStr.compareTo(CONTROLLER_STRADDR_LINEARBOT) == 0)
529                 {
530                     //Make state for the linearbot
531                     System.out.print("Making Linear bot state");
532                     Status tempStatus = makeState(valueStr);
533                     //Check the status
534                     if(tempStatus != null)
535                     {
536                         this.linearBotState = tempStatus;
537                         newData = true;
538                     }
539 
540 
541 
542                      System.out.println(this.linearBotState.getString());
543 
544                     //Check if it is a calibration status, and set values if so
545                     if (checkForCalibParam(this.linearBotState))
546                     {
547                         //Copy all the values
548                         String[] onlyValues = new String[valueStr.length - 1];
549                         System.arraycopy(valueStr, 1, onlyValues, 0, valueStr.length - 1);
550                         //Put the values
551                         calibrationParams.putValue(onlyValues);
552 
553                         //Check if send should be set
554                         if (calibrationParams.isElevatorCalib())
555                         {
556                             //Check if the calib has been sent
557                             //So it doesnt add listeners multiple times
558                             if (!calibrationParams.isSent())
559                             {
560                                 //Check for nullpointer
561                                 //And add the listeners to the calib param status before
                                    notifying
562                                 if (listenerList != null)
563                                 {
564                                     // add listeners to the new state
565                                     for (StatusListener listener : this.listenerList)
566                                     {
567                                         calibrationParams.addListener(listener);
568                                     }
569                                 }
570                                 calibrationParams.updateTrays();
571                                 calibrationParams.setSend(true);
572                                 newData = true;
573                             }
574 
575                         }
576                     }
577 
578                 } //Check for elevator address
579                 else if (addrStr.compareTo(CONTROLLER_STRADDR_ELEVATOR) == 0)
580                 {
581                       //Check the status
```

```
582                 Status tempStatus = makeState(valueStr);
583                  //Check the status
584                  if(tempStatus != null)
585                  {
586                      this.elevatorState = tempStatus;
587                      newData = true;
588                  }
589
590
591                  //Check if it is a calibration status, and set values if so
592                  if (checkForCalibParam(elevatorState))
593                  {
594                      //Copy all the values
595                      String[] onlyValues = new String[valueStr.length - 1];
596                      System.arraycopy(valueStr, 1, onlyValues, 0, valueStr.length - 1);
597
598                      //Put values in the calib param
599                      calibrationParams.putValue(onlyValues);
600
601                      //Check if send should be set
602                      if (calibrationParams.isLinearCalib())
603                      {
604                          //Check if the calib has been sent
605                          //So it doesnt add listeners multiple times
606                          if (!calibrationParams.isSent())
607                          {
608                              //Check for nullpointer
609                              //And add the listeners to the calib param status before
                                 notifying
610                              if (listenerList != null)
611                              {
612                                  // add listeners to the new state
613                                  for (StatusListener listener : this.listenerList)
614                                  {
615                                      calibrationParams.addListener(listener);
616                                  }
617                              }
618                              //This calibration parameter should be sent
619                              //Set the appropriate bool and update the trays
620                              newData = true;
621                              calibrationParams.updateTrays();
622                              calibrationParams.setSend(true);
623
624                          }
625                      }
626                  }
627              }
628          }
629      }
630
631      /**
632       * Check for the calibration parameter or
633       *  readyToRecieve state and trigger if both are ready
634       * Else trigger the states which is not ready
635       *
636       * @param elevatorState The elevator State
637       * @param linearBotState The linearbot state
638       * @return Returns true if status was updated, false if not
639       */
640      private boolean checkStatesAndTrigger(Status elevatorState, Status linearBotState)
641      {
642          //Return bool
643          boolean sentStatus = false;
644
645          //Safeguarding against null-pointer
646          if (elevatorState != null && linearBotState != null)
647          {
648              //CHECK FOR CALIBRATION PARAMETER STATUS
649              //Check if calibration parameter is updated and should be sent
650              if (calibrationParams.isElevatorCalib() &&
                 calibrationParams.isLinearCalib())
651              {
652                  //Check if the send bool is set in the parameter
```

```
653                 if (calibrationParams.isSend())
654                 {
655                     calibrationParams.setSend(false);
656                     calibrationParams.setSent(true);
657                     calibrationParams.notifyListeners();
658                     sentStatus = true;
659                 }
660             }
661 
662 
663                 //Check for elevator critical status
664                 if(elevatorState.critical())
665                 {
666                     //Set the current state to sent
667                     elevatorState.setSent(true);
668                     //Notify the listeners of this state
669                     elevatorState.notifyListeners();
670                     sentStatus = true;
671                 }
672                 //Check for critical linear status
673                  if(linearBotState.critical())
674                  {
675                     //Set the current state to sent
676                         linearBotState.setSent(true);
677                         //Notify the listeners of this state
678                         linearBotState.notifyListeners();
679                         sentStatus = true;
680                  }
681 
682                 //CHECK IF BOTH STATUSES ARE READY TO RECIEVE
683                 else if (this.checkForReady(elevatorState) &&
                    this.checkForReady(linearBotState))
684                 {
685                     //Set the state to sent
686                     elevatorState.setSent(true);
687                     linearBotState.setSent(true);
688                     //Notify the listeners of this state
689                     elevatorState.notifyListeners();
690                     sentStatus = true;
691 
692                 }
693 
694                 //CHECK WHICH STATE IS NOT READY
695                 //Check the elevator state
696                 else if (!this.checkForReady(elevatorState))
697                 {
698                     if (!checkForCalibParam(elevatorState))
699                     {
700                         //Set the state to sent
701                         elevatorState.setSent(true);
702                         //Notify the listeners of this state
703                         elevatorState.notifyListeners();
704                         sentStatus = true;
705                         //Check which are not ready
706                     }
707                 } else if (!this.checkForReady(linearBotState))
708                 {
709                     if (!checkForCalibParam(linearBotState))
710                     {
711                         //Set the state to sent
712                         linearBotState.setSent(true);
713                         //Notify the listeners of this state
714                         linearBotState.notifyListeners();
715                         sentStatus = true;
716                         //Check which are not ready
717                     }
718                 }
719         }
720         //Return bool to tell
721         return sentStatus;
722     }
723 
724     /**
```

```
725         * Returns true if the status is a ready to recieve status
726         *
727         * @param checkState The status to checkl
728         * @return Returns true if the status is a ready to recieve status
729         */
730        private boolean checkForReady(Status checkState)
731        {
732            boolean isReady = false;
733            if (Byte.compare(checkState.getStatusAddress(),
                State.ReadyToRecieve.getStateValue()) == 0)
734            {
735                isReady = true;
736            }
737            return isReady;
738        }
739
740        /**
741         * Return true if the status correspondes with
742         */
743        private boolean checkForCalibParam(Status checkState)
744        {
745            boolean wasCalib = false;
746            if(checkState != null)
747            {
748                  if
                    (checkState.getString().compareTo(State.PARAMETER.getStatus().getString(
                    )) == 0)
749                    {
750                wasCalib = true;
751                    }
752            }
753
754            return wasCalib;
755        }
756
757
758        /**
759         * Make a state from the given statebyte[]
760         *
761         * @param stateByte Statebyte to create state from
762         * @return Returns the created state, else null!
763         */
764        public Status makeState(String[] stateByte)
765        {
766            Status returnState = null;
767
768            //Initiate the object with 0
769            byte cmdAddr = 0;
770            //Save the cmd byte
771            if(stateByte[0].length() <= 3)
772            {
773                cmdAddr = Byte.valueOf(stateByte[0]);
774            }
775
776
777            //Get the status based on cmd address
778            State state = State.get(cmdAddr);
779
780            //Nullpointer check
781            if (state != null)
782            {
783                //Create new status based on the returned state
784                Status status = state.getStatus();
785                returnState = status.returnNew();
786
787
788                //CHECK FOR VALUES
789                if (stateByte.length > 1)
790                {
791                    //Make new byte[] to store values in
792                    String[] valueByte = new String[stateByte.length - 1];
793
794                    //Copy the array
```

```
                System.arraycopy(stateByte, 1, valueByte, 0, stateByte.length - 1);

                //Put the values
                returnState.putValue(valueByte);
            }

            //Check for nullpointer
            if (listenerList != null)
            {
                // add listeners to the new state
                for (StatusListener listener : this.listenerList)
                {
                    returnState.addListener(listener);
                }
            }
        }
        return returnState;
    }

    /**
     * Add to the sendqueue, only commands
     *
     * @param cmd Commando to be performed
     */
    public synchronized void addSendQ(Commando cmd)
    {
        sendQeue.add(cmd);
    }



    /**
     * Resize an array with only carrying information, -1 is considered as not
     * valuable information.
     *
     * @param inputArr
     * @return Return an resized array
     */
    private byte[] resizeArray(byte[] inputArr, byte resizeOption)
    {
        int length = inputArr.length;
        int cnt = 0;

        //Find the actual length of the array
        for (int i = 0; i < length; ++i)
        {
            if (Byte.compare(inputArr[i], resizeOption) != 0)
            {
                ++cnt;
            }

        }
        //Create the new byte[]
        byte[] returnByte = new byte[cnt];
        //Copy the wanted values
        System.arraycopy(inputArr, 0, returnByte, 0, cnt);
        //Return the resized byte[]
        return returnByte;
    }

    /**
     * Handles the commandos given in parameter. Tasks handled based on Commando
     * subclass.
     *
     * @param cmd The commando to perform
     */
    private void sendCommand(Commando cmd)
    {

        String elevatorString = null;
        String linearString = null;


```

```
868             //Check for move command
869             if (cmd instanceof Move)
870             {
871                 //Do the X-Y movement first and send to the controller
872                 Move cmdMove = (Move) cmd;
873 
874                 if ((cmdMove.isxMoveBool() == true) && (cmdMove.isyMoveBool() == true))
875                 {
876                     String sendString = makeCMDString(CONTROLLER_STRADDR_LINEARBOT,
                        cmdMove.getCmdAddr());
877                     String valueString = new String(String.valueOf(cmdMove.getxMove())+
                        ", " + String.valueOf(cmdMove.getyMove()));
878 
879                     sendString +=  ", " + valueString;
880 
881                     //Send the data
882                     this.writeStringLinear(sendString);
883                 }
884                 //Check for z move should be sent
885                 if ((cmdMove.iszMoveBool() == true))
886                 {
887                     String sendString = makeCMDString(CONTROLLER_STRADDR_ELEVATOR,
                        cmdMove.getCmdAddr());
888                     String valueString = String.valueOf(cmdMove.getzMove());
889 
890                     sendString += ", " + valueString;
891 
892                     //Send the data
893                     this.writeStringElevator(sendString);
894                 }
895 
896 
897             }
898             //Send command
899             else if (cmd != null)
900             {
901                 //Check if the command is for the elevator
902                 if(cmd.isForElevatorRobot())
903                 {
904                          //Make string for elevator
905                 elevatorString = makeCMDString(CONTROLLER_STRADDR_ELEVATOR,
                    cmd.getCmdAddr());
906                 //Send data and set bool
907                 if(elevatorString != null)
908                 {
909                     //Check for null value
910                     if(cmd.getValue() != null)
911                     {
912                         //Make a new string with the value
913                         String stringWithValue = makeString(CONTROLLER_STRADDR_ELEVATOR,
                            cmd.getStringCmdAddr(), cmd.getValue());
914                         this.writeStringElevator(stringWithValue);
915                     }
916                     else
917                     {
918                         //Send the command
919                         this.writeStringElevator(elevatorString);
920                     }
921                  }
922                 }
923 
924                 //Check if the command is for the linear
925                 if(cmd.isForLinearRobot())
926                 {
927                 //Send linear data and set bool
928                 linearString = makeCMDString(CONTROLLER_STRADDR_LINEARBOT,
                    cmd.getCmdAddr());
929                 //Check for null
930                 if(linearString != null)
931                 {
932                      if(cmd.getValue() != null)
933                     {
934                         //Make a new string with the value
```

```
                    String stringWithValue =
                    makeString(CONTROLLER_STRADDR_LINEARBOT, cmd.getStringCmdAddr(),
                    cmd.getValue());
                    this.writeStringLinear(stringWithValue);
                }
                else
                {
                    //Send the command
                    this.writeStringLinear(linearString);
                }
            }
            }
        }


    }

    /**
     * Return a string from the given device addr, cmd addr
     * and byte arr payload
     *
     * @param devAddr Device address in string
     * @param cmdString Command address in string
     * @param payload Byte arr payload
     * @return Return the string containing all the params divided with ","
     */
    private String makeString(String devAddr, String cmdString, byte[] payload)
    {
        return (devAddr + "," + cmdString + "," + Arrays.toString(payload));
    }


    /**
     * Return a string from the given device addr, cmd addr
     * and string arr payload. Seperated with ",".
     *
     * @param devAddr Device address
     * @param cmdString Command address
     * @param payload payload/values
     * @return
     */
     private String makeString(String devAddr, String cmdString, String[] payload)
    {
        //Create the string object
        String valueString = "";
        //Copy all the values to a string
        for(int i=0; i<payload.length; ++i)
        {
            valueString += "," + payload[i];
        }

        //Append all the string with the value at last and return it
        return (devAddr + "," + cmdString + "," + valueString);
    }



    /**
     * Put all the bytes together in a byte arr and return it
     *
     * @param devAddr The device address in byte
     * @param cmdAddr The cmd address in byte
     * @param payload The payload in byte[]
     * @return Return a complete byte[] with all sending values, returns null if
     * nothing was copied
     */
    private byte[] addBytes(byte devAddr, byte cmdAddr, byte[] payload)
    {
        byte[] totalByte = null;
        //Keep track of the next array pos
        int arrayCnt = 0;

        //Check if cmd address is present
```

```
1006            if (cmdAddr != 0)
1007            {
1008                totalByte = new byte[payload.length + 2];
1009                //Store the device address and cmd address
1010                totalByte[arrayCnt++] = devAddr;
1011                totalByte[arrayCnt++] = cmdAddr;
1012            } else
1013            {
1014                totalByte = new byte[payload.length + 1];
1015                //Store the device address and cmd address
1016                totalByte[arrayCnt++] = devAddr;
1017            }
1018
1019            //Add the whole payload
1020            if (totalByte != null)
1021            {
1022                System.arraycopy(payload, 0, totalByte, arrayCnt, payload.length);
1023            }
1024
1025            return totalByte;
1026        }
1027
1028        /**
1029         * ONLY FOR TESTING
1030         *
1031         * @return
1032         */
1033        public boolean returnTriggered()
1034        {
1035            return readyTriggered;
1036        }
1037
1038        /**
1039         * Add class as listener to statuses. listener needs to implement
1040         * StatusListener interface
1041         *
1042         * @param listener to add as listener to statuses
1043         */
1044        public void addListener(StatusListener listener)
1045        {
1046            this.listenerList.add(listener);
1047        }
1048
1049    }
1050
```

```java
/*
 * This class is the listener interface for the serial reader
 * implement method serialDataAvailable and add as listener to reader
 * object for getting notified when message is recieved.
 */
package SerialCommunication;

/**
 * This class is used for as "notify" functions with listeners
 * Between the Serial port and implementing classes
 * Raw data is put in the parameter fields
 *
 * @author kristianandrelilleindset
 */
public interface SerialInputListener
{
    /**
     * Method called by classes implementing this interface.
     */
    public void serialDataAvailable(byte[] data);
    public void serialDataAvailable(String[] data);
}


```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package SerialCommunication;

import com.fazecast.jSerialComm.SerialPort;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * This class creates and maintains the Serial port connection
 * @author Per Espen
 */
public class SerialJComm extends Thread
{

    //The serial port
    private SerialPort port;

    //The serial port name
    private String portName;


    // variable holding the desired rate of sending and receiving data
    private static final int DATA_RATE = 115200;

    //Reader and writer stream
    InputStream reader;
    OutputStream writer;

    //Buffered reader
    // variable holding the bufferedreader
    private BufferedReader input;

    // arrayList holding all of the listeners interested in the serial communication
    private final ArrayList<SerialInputListener> listeners;

    //Variable holding the data to be sent to the Arduino
    private byte[] dataToBeSent;
    private boolean dataToSend = false;

    private String[] inputStringData;
    private byte[] inputByteData;

    public SerialJComm(String portName)
    {

        //Print the port name
        getPortNames();
        this.portName = portName;

        // creating the arrayList of listeners
        this.listeners = new ArrayList<>();

    }

    /**
     * Connect to the serial port, with the portname in the constructor
     */
    public void connect()
    {
        //Find the port
        port = findPort(portName);
```

```
74          try
75          {
76              System.out.println("Opening Port. ");
77              //Wait for connection
78              Thread.sleep(500);
79              initializePort();
80              Thread.sleep(1000);
81
82          } catch (InterruptedException ex)
83          {
84              System.out.println("Could not open port. ");
85              Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE, null, ex);
86          }
87
88          //Check if port was successfully opened
89          if (this.port.isOpen())
90          {
91              System.out.println("Port is open");
92          } else
93          {
94              System.out.println("Port is NOT open");
95          }
96      }
97
98      /**
99       * Get the input and output streams from the SerialPort interface
100      */
101     private void initializePort()
102     {
103         //Opening the port and setting the buad rate
104         this.port.setBaudRate(DATA_RATE);
105
106         //Try to open the port
107         this.port.openPort();
108         port.setComPortTimeouts(port.TIMEOUT_READ_SEMI_BLOCKING, 0,0);
109
110
111         //Return the input stream
112         // creates an inputstream for reading data
113         this.input = new BufferedReader(new
            InputStreamReader(this.port.getInputStream()));
114
115
116
117     }
118
119      @Override
120     public void run()
121     {
122         //While loop to keep thread running as long as port is open
123         while (port.isOpen())
124         {
125             //Checks if data is availble to be sent
126             if (dataToSend)
127             {
128                 this.sendData();
129                 dataToSend = false;
130             }
131
132             try
133             {
134                 //read from serial port
135                 while (input.ready())
136                 {
137                     //SLEEP FOR THE ALL THE BYTES INCOMMING ON THE SERIAL TO GET IN
                        THE BUFFER BEFORE READING STARTS
138
139                     try
140                     {
141                         Thread.sleep(10);
142                     } catch (InterruptedException ex)
143                     {
144                         System.out.println("SerialJCOMM has interrupted sleep..");
```

```
145
                            Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE
                            , null, ex);
146                     }
147
148
149                     //Uses INPUT READ BUFFER to read a line until \n is in the line
150                     try
151                     {
152                         String[] dataArrString = input.readLine().split(",");
153                         System.out.println(Arrays.toString(dataArrString));
154                         notifyListeners(dataArrString);
155
156                     } catch (UnsupportedEncodingException ex)
157                     {
158
                            Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE
                            , null, ex);
159                     } catch (IOException ex)
160                     {
161                         System.out.println("SerialJCOMM port " + this.portName + "
                            got IO exception..");
162
                            Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE
                            , null, ex);
163                     }
164                 }
165             } catch (IOException ex)
166             {
167                 Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE,
                    null, ex);
168             }
169         }
170         this.close();
171     }
172
173
174
175     /**
176      * Print the given byte arr
177      */
178     private void printStringArr(String[] stringArr)
179     {
180         String inputString;
181         System.out.println("INPUT ARR");
182         int size = stringArr.length;
183         for (int i = 0; i < size; ++i)
184         {
185             System.out.println(stringArr[i]);
186         }
187     }
188
189     /**
190      * Print the given byte arr
191      */
192     private void printByteArr(byte[] byteArr)
193     {
194         String inputString;
195         try
196         {
197             inputString = new String(byteArr, "UTF-8");
198             System.out.print("Input string from reader:(");
199             System.out.println(inputString);
200         } catch (UnsupportedEncodingException ex)
201         {
202             System.out.print("Tried creating string from byte arr");
203             Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE, null, ex);
204         }
205     }
206
207     /**
208      * Print the port names connected to the SerialPort
209      *
```

```
210         * @return Print the port names connected to the SerialPort
211         */
212        private String[] getPortNames()
213        {
214            System.out.println("Finding ports");
215            SerialPort[] ports = SerialPort.getCommPorts();
216            String[] result = new String[ports.length];
217            for (int i = 0; i < ports.length; i++)
218            {
219                result[i] = ports[i].getSystemPortName();
220                System.out.println(result[i]);
221            }
222
223            return result;
224        }
225
226        /**
227         * Print the port names connected to the SerialPort
228         *
229         * @return Print the port names connected to the SerialPort
230         */
231        private SerialPort findPort(String findPort)
232        {
233            SerialPort foundPort = null;
234            SerialPort[] ports = SerialPort.getCommPorts();
235            String[] result = new String[ports.length];
236            for (int i = 0; i < ports.length; i++)
237            {
238                result[i] = ports[i].getSystemPortName();
239                if (findPort.compareTo(ports[i].getSystemPortName()) == 0)
240                {
241                    foundPort = ports[i];
242                }
243            }
244
245            //Check if the port was not found
246            //Check what the string was, and do another search on possible port name
247            if (foundPort == null)
248            {
249                //If the device has changed from ACM0 to ACM1
250                if (findPort.contentEquals("ttyACM0"))
251                {
252                    findPort = "ttyACM1";
253
254                    for (int i = 0; i < ports.length; i++)
255                    {
256                        result[i] = ports[i].getSystemPortName();
257                        if (findPort.compareTo(ports[i].getSystemPortName()) == 0)
258                        {
259                            foundPort = ports[i];
260                        }
261                    }
262
263                }
264            }
265
266            return foundPort;
267        }
268
269        /**
270         **** THE WRITER PART OF THE SERIAL PORT ****
271         */
272        /**
273         * sends the data received from the function call
274         */
275        public void sendData(byte[] bytesToSend)
276        {
277            // try to send the read data
278            try
279            {
280                //Print the data to be sent - for debugging
281                System.out.println("Writer sending data");
282                String inputString = new String(this.getDataToSend(), "UTF-8");
```

```
283                 System.out.println(inputString);
284 
285                 //Send the data
286                 this.port.writeBytes(this.getDataToSend(), this.getDataToSend().length);
287 
288             } catch (IOException ex)
289             {
290                 ex.printStackTrace();
291                 System.out.println("Serial: " + ex.toString());
292             }
293         }
294 
295         /**
296          * sends the data received from the function call
297          */
298         public synchronized void sendData(String stringToSend)
299         {
300             // try to send the read data
301             try
302             {
303                 //Set data to be sent
304                 this.setDataToBeSent(stringToSend.getBytes("UTF-8"));
305                 //Print the data to be sent - for debugging
306                 //Data to send
307                 System.out.println("Writer sending data");
308                 String inputString = new String(this.getDataToSend(), "UTF-8");
309                 System.out.println(inputString);
310 
311                 //Send the data
312                 this.port.writeBytes(this.getDataToSend(), this.getDataToSend().length);
313                 // port.flush();
314 
315             } catch (IOException ex)
316             {
317                 ex.printStackTrace();
318                 System.out.println("Serial: " + ex.toString());
319             }
320 
321         }
322 
323         /**
324          * Sends the data saved in the "data to be sent" field
325          */
326         private synchronized void sendData()
327         {
328 
329             // try to send the read data
330             try
331             {
332                 //Set data to be sent
333                 // this.setDataToBeSent(stringToSend.getBytes("UTF-8"));
334                 //Print the data to be sent - for debugging
335                 //Data to send
336                 System.out.println("Writer sending data");
337                 String inputString = new String(this.getDataToSend(), "UTF-8");
338                 System.out.println(inputString);
339 
340                 //Send the data
341                 this.port.writeBytes(this.getDataToSend(), this.getDataToSend().length);
342                 // port.flush();
343 
344             } catch (IOException ex)
345             {
346                 ex.printStackTrace();
347                 System.out.println("Serial: " + ex.toString());
348             }
349 
350         }
351         /**
352          * Set value to the data to be sent field
353          * @param dataString Data string to be sent
354          */
355         public synchronized void setDataToBeSent(String dataString)
```

```
356         {
357             // setting the start and stopbytes of the data to be sent
358             // making it easy for the Arduino to reecognize i
                this.calculator.getCalculatedData()f the message
359             // is at the beginning when it starts to receive.
360 
361             byte[] dataToSend = null;
362             try
363             {
364                 this.dataToBeSent = dataString.getBytes("UTF-8");
365             } catch (UnsupportedEncodingException ex)
366             {
367                 Logger.getLogger(SerialJComm.class.getName()).log(Level.SEVERE, null, ex);
368             }
369 
370             this.dataToSend = true;
371         }
372         /**
373         * Set the data to be sent, in the databyte byte arr
374         * @param dataByte Data to be sent
375         */
376         private synchronized void setDataToBeSent(byte[] dataByte)
377         {
378             // setting the start and stopbytes of the data to be sent
379             // making it easy for the Arduino to reecognize i
                this.calculator.getCalculatedData()f the message
380             // is at the beginning when it starts to receive.
381 
382             int byteLength = dataByte.length;
383             reziseSendData(byteLength);
384 
385             //Iterate through the incomming databyte and set it to the send byte
386             for (int i = 0; i < byteLength; ++i)
387             {
388                 this.dataToBeSent[i] = dataByte[i];
389             }
390 
391         }
392         /**
393          * Return data in data to be sent field
394          * @return byte array of the sending data
395          */
396         private synchronized byte[] getDataToSend()
397         {
398             return this.dataToBeSent;
399         }
400         /**
401          * Resize the send data with given param length
402          * @param byteLength length of byte arr
403          */
404         private void reziseSendData(int byteLength)
405         {
406             this.dataToBeSent = new byte[byteLength];
407         }
408 
409         private void resetSendData()
410         {
411             int byteLength = this.dataToBeSent.length;
412             //Iterate through the incomming databyte and set it to the send byte
413             for (int i = 0; i < byteLength; ++i)
414             {
415                 this.dataToBeSent[i] = 0;
416             }
417         }
418 
419         /**
420          * ******** LISTENERS AND NOTIFY*********
421          */
422         /**
423          * Add a listener interested in the input data to the list. listener has to
424          * implement the CalculationListener interface
425          *
426          * @param listener
```

```
427         */
428        public synchronized void addListener(SerialInputListener listener)
429        {
430            this.listeners.add(listener);
431        }
432
433        /**
434         * Method notifying notify all listeners of data now available for reading
435         * listener has to implement the CalculationListener interface
436         */
437        private synchronized void notifyListeners(String[] input)
438        {
439            if (this.listeners != null)
440            {
441                for (SerialInputListener listener : this.listeners)
442                {
443                    listener.serialDataAvailable(input);
444                }
445            }
446        }
447
448        /**
449         * Method closing the connection with the serialport.
450         */
451        public synchronized void close()
452        {
453            // check if there is a instance of a serialport
454            if (this.port != null)
455            {
456                // remove eventlisteners from the serialport
457                this.port.removeDataListener();
458                // close the connection
459                this.port.closePort();
460            }
461        }
462    }
463
```

H.4 Image Processing

```java
package ImageProcessing;

import org.opencv.core.Mat;
import org.opencv.videoio.VideoCapture;
import static org.opencv.videoio.Videoio.CAP_PROP_FRAME_HEIGHT;
import static org.opencv.videoio.Videoio.CAP_PROP_FRAME_WIDTH;

/**
 * A class handling handle camera activity, open camera, take picture
 *
 * @author Kristian Andre Lilleindset
 * @version 13-02-2018
 */
public class Camera
{

    // Which connected camera to open
    private final int camToOpen = 0;

    // Picture frame
    private final Mat frame;

    // Timestamp of last image
    private long timestamp;

    // Camera
    private final VideoCapture cam;

    // Cameras field of view (angle)
    private final int FOV = 78;



    public Camera()
    {
        boolean found = false;
        // Open a camerasource
        this.cam = new VideoCapture(camToOpen);

        // create a Mat frame variable
        this.frame = new Mat();

        // set the desired width and height of the pictures taken
        this.cam.set(CAP_PROP_FRAME_WIDTH, 1920);
        this.cam.set(CAP_PROP_FRAME_HEIGHT, 1080);
    }


    /**
     * Take a picture and return the Mat frame.
     *
     * @param cameraHeight distance between lens and surface
     * @return RoeImage with picture and properties
     */
    public RoeImage takePicture(float cameraHeight, int pictureIndex)
    {
        // return variable
        RoeImage result = new RoeImage(cameraHeight, this.FOV);

        // take picture
        this.cam.read(this.frame);
        // update timestamp for image capturing
        this.timestamp = System.currentTimeMillis();

        // add picture to result
        result.SetImage(this.frame);

        // add timestamp of captured image
        result.setTimeStamp(this.timestamp);

        // set the index of the picture captured
        result.setPictureIndex(pictureIndex);

```

```
74            // the image captured with properties
75            return result;
76        }
77    }
78
```

```java
package ImageProcessing;

/**
 * Listener for transferring RoeImage
 * @author odroid
 */
public interface ImageCaptureListener
{

    public void noitfyImageCaptured(RoeImage capturedImage);

}

```

```java
package ImageProcessing;



import RoeRobot.Coordinate;
import java.util.ArrayList;
import java.util.List;
import java.util.Stack;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import org.opencv.imgproc.Moments;

/**
 * Class handling image processing of a image.
 * Searches for roe in a image.
 *
 * returns the position of roe in the image by x-y direction.
 * @author KristianAndreLilleindset
 * @version 17-04-2018
 */

public class ImageProcessing implements Runnable
{
    // list of images to process
    private final Stack<RoeImage> processQueue;

    // list of listeners to the image processing
    private final ArrayList<ImageProcessingListener> listeners;

    // debug variable
    private final boolean debug = false;

    public ImageProcessing()
    {
        // create lists for images and listeners
        this.processQueue = new Stack();
        this.listeners = new ArrayList();
    }
    /**
     * Process image for detecting roe and update coordinates to image.
     */
    private void processImage()
    {
        // get image from processing queue
        RoeImage processingImage = this.getImageFromProcessingQueue();

        // get RGB picture from image object
        Mat rgbImage = processingImage.getImage();

        // transform from RGB to grayscale
        Mat grayImage = new Mat();


        Imgproc.cvtColor(rgbImage, grayImage, Imgproc.COLOR_BGR2GRAY);
        //Imgproc.COLOR_RGB2GRAY

        // brighten grayscale image
        Mat brightGrayImage = new Mat();
        grayImage.convertTo(brightGrayImage, -1, 5, 10);

        // binarize grayscale image
        Mat bwImage = new Mat();
        Imgproc.threshold(grayImage, bwImage, 0, 255, Imgproc.THRESH_BINARY +
        Imgproc.THRESH_OTSU);

        // create structure element for eroding and dilating
        // using circular SE for keeping correct shape of desired objects
        Mat structureElement;
```

```
72          structureElement = Imgproc.getStructuringElement(Imgproc.MORPH_ELLIPSE, new
            Size(35,35));
73
74          // erode image for removing unwanter artifacts
75          Mat erodedImage = new Mat();
76          Imgproc.erode(bwImage, erodedImage, structureElement);
77
78          // dilate image restoring objects to correct size, use same SE as
79          // eroding for getting correct size.
80          Mat dilatedImage = new Mat();
81          Imgproc.dilate(erodedImage, dilatedImage, structureElement);
82
83          // find BLOB's in image (Binary Large Objects)
84          // finds rectangle surrounding the blob
85          List<MatOfPoint> blobContours = new ArrayList();
86          Imgproc.findContours(dilatedImage, blobContours, new
            Mat(),Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
87
88          // find properties of contours
89          Moments blobProperties;
90
91          for(int  i = 0; i < blobContours.size(); i++)
92          {
93              // get properties from contour of first blob
94              blobProperties = Imgproc.moments(blobContours.get(i));
95
96              // calculate center of blob in x and y direction,
97              // origo of image at top left corner
98              int blobCenterX = (int) (blobProperties.m10/blobProperties.m00);
99              int blobCenterY = (int) (blobProperties.m01/blobProperties.m00);
100
101             // Recalculate to match robot coordinate system
102             // origo being bottom right corner
103             blobCenterX = rgbImage.width() - blobCenterX;
104             blobCenterY = rgbImage.height() - blobCenterY;
105
106             // create coordinate object
107             Coordinate blobCentroid = new Coordinate(blobCenterY, blobCenterX, 0);
108
109             // add pixel coordinate of blob centroid to image being processed
110             processingImage.addRoePositionPixel(blobCentroid);
111
112             // calculate millimeter coordinate of blob centroid to image being
                processed
113             this.pixelToMillimeterConversion(blobCentroid, processingImage);
114         }
115
116
117         System.out.println("Image processed list size " +
            processingImage.getRoePositionPixelList().size());
118
119         // notify listeners of image finished processing
120         // add processed image as parameter
121         for(ImageProcessingListener listener : this.listeners)
122         {
123             listener.notifyImageProcessed(processingImage);
124         }
125
126
127
128         //======================== DEBUG ============================//
129
130         // add visual rectangle around the blobs detected
131         Mat boundedBlobs = dilatedImage;
132         Mat boundedBlobsCentroid = dilatedImage;
133         if(debug)
134         {
135             for(int i = 0; i < blobContours.size(); i++)
136             {
137                 if (Imgproc.contourArea(blobContours.get(i)) > 1 )
138                 {
139                     Rect rect = Imgproc.boundingRect(blobContours.get(i));
140                     if (rect.height > 1)
```

```
141                     {
142                         Imgproc.rectangle(boundedBlobs, new Point(rect.x,rect.y),
                            new Point(rect.x + rect.width, rect.y + rect.height), new
                            Scalar(255,255,255));
143                         Imgproc.rectangle(boundedBlobsCentroid, new
                            Point(rect.x,rect.y), new Point(rect.x + rect.width, rect.y
                            + rect.height), new Scalar(255,255,255));
144
145                     }
146                 }
147
148                 // get properties from contour of first blob
149                 blobProperties = Imgproc.moments(blobContours.get(i));
150
151                 // calculate center of blob in x and y direction,
152                 // origo of image at top left corner
153                 int blobCenterX = (int) (blobProperties.m10/blobProperties.m00);
154                 int blobCenterY = (int) (blobProperties.m01/blobProperties.m00);
155
156                 // mark centroid of blobs
157                 Imgproc.circle(boundedBlobsCentroid, new
                    Point(blobCenterX,blobCenterY), 1, new Scalar(0,0,0));
158             }
159         }
160     }
161
162     /**
163      * Calculate distance from pixels to mm.
164      *
165      */
166     private void pixelToMillimeterConversion(Coordinate pixelCoordinate, RoeImage
        processingImage)
167     {
168         // get variables needed from the image
169         double fieldOfView = processingImage.getFieldOfView();
170         double distance = processingImage.getDistance();
171         double imageHeight = processingImage.getImage().height();
172         double imageWidth = processingImage.getImage().width();
173
174         // calculate length of diagonal of image in mm
175         double diagonalMillimeter = distance *
            Math.cos((fieldOfView/2)*(Math.PI/180)) * 2;
176
177         // calculate length of diagonal in pixels
178         double diagonalPixel = Math.sqrt(Math.pow(imageHeight, 2) +
            Math.pow(imageWidth, 2));
179
180         // calculate angle of diagonal
181         double theta = Math.atan(imageHeight/imageWidth);
182
183         // calculate width of image in millimeter
184         double imageWidthMillimeter = Math.cos(theta) * diagonalMillimeter;
185
186         // calaculate height of image in millimeter
187         double imageHeightMillimeter = Math.sin(theta) * diagonalMillimeter;
188
189         // calculate the size of a pixel in x direction in mm
190         double pixelSizeDirX = imageHeightMillimeter/imageHeight;
191
192         // calculate tje size of a pixel in y direction in mm
193         double pixelSizeDirY = imageWidthMillimeter/imageWidth;
194
195
196         // uses xDir size on y coordinate and opposite
197         // since image is rotated 90 degrees in robot
198         // calculate distance to x and y position in millimeter
199         double xPositionMillimeter = pixelCoordinate.getxCoord() * pixelSizeDirY;
200         double yPositionMillimeter = pixelCoordinate.getyCoord() * pixelSizeDirX;
201
202         // create coordinate in millimeters
203         Coordinate millimeterCoordinate = new Coordinate(xPositionMillimeter,
            yPositionMillimeter);
204
```

```
205             // add coordinate in millimeter to the RoeImage being processed
206             processingImage.addRoePositionMillimeter(millimeterCoordinate);
207         }
208 
209 
210         /**
211          * Add image to processing list.
212          *
213          * Adds image to bottom of list (FiFo)
214          *
215          * @param image to add to list
216          */
217         public synchronized void addImageToProcessingQueue(RoeImage image)
218         {
219             this.processQueue.add(image);
220         }
221 
222 
223         /**
224          * Get image from processing list.
225          * Returns image from top of list (FiFo)
226          *
227          * @return image from top of list
228          */
229         public synchronized RoeImage getImageFromProcessingQueue()
230         {
231             return this.processQueue.pop();
232         }
233 
234 
235         /**
236          * Get number of elements in processing queue
237          *
238          * @return number of elements in processing queue
239          */
240         private synchronized int nmbrOfElementsInProcessingQueue()
241         {
242             return this.processQueue.size();
243         }
244 
245 
246         /**
247          * Add listener to listener list
248          * Listener must implement ImageProcessingListener interface
249          *
250          * @param listener being added to list
251          */
252         public void addListener(ImageProcessingListener listener)
253         {
254             this.listeners.add(listener);
255         }
256 
257 
258         @Override
259         public void run()
260         {
261             //check if processing queue is empty, if not start processing image
262             while(true)
263             {
264 
265                 if(this.nmbrOfElementsInProcessingQueue() > 0)
266                 {
267                 this.processImage();
268                 }
269             }
270         }
271     }
```

```java
package ImageProcessing;

/**
 * Listener for the image processing.
 * notifyes when there are processed images in queue.
 * Listeners must implement this interface for being notifyed.
 *
 * @author KristianAndreLilleindset
 * @version 17-04-2018
 */
public interface ImageProcessingListener
{

    /**
     * Notification of image finished processed
     *
     * @param processedImage from image processing
     */
    public void notifyImageProcessed(RoeImage processedImage);
}
```

```java
package ImageProcessing;

import RoeRobot.Coordinate;
import java.util.ArrayList;
import java.util.List;
import org.opencv.core.Mat;


/**
 * Class represents image with the following properties:
 * - timestamp
 * - index of image in tray
 * -
 *
 * @author KristianAndreLilleindset
 * @version 17-04-2018
 */
public class RoeImage
{
    // image as Mat file
    private Mat image;

    // timestamp of image
    private long timestamp;

    // frame index in tray
    private int pictureIndex;

    // variable holding height of camera above surface in mm
    private final float captureHeight;

    //variable holding the Field Of View from the camera (angle)
    private final int FOV;

    // list holding position of roe detected in the image
    private final ArrayList<Coordinate> roePositionPixels;

    // list holding position of roe detecten in image in millimeters
    private final ArrayList<Coordinate> roePositionMillimeters;

    /**
     *
     * @param captureHeight distance between lens and surface captured in mm
     * @param fieldOfView field of view of camera (angle)
     */
    public RoeImage(float captureHeight, int fieldOfView)
    {
        // save the camera to surface distance
        this.captureHeight = captureHeight;
        //Set image to null
        image = null;

        // save the cameras FOV
        this.FOV = fieldOfView;

        // create list of roe positions
        roePositionPixels = new ArrayList();
        roePositionMillimeters = new ArrayList();
    }


    /**
     * Set image
     *
     * @param image added to RoeImage
     */
    public void SetImage(Mat image)
    {
        this.image = image;
    }


    /**
```

```
74          * Get image
75          *
76          * @return Mat frame image
77          */
78         public Mat getImage()
79         {
80             return this.image;
81         }
82 
83 
84         /**
85          * Set timestamp of the image captured
86          *
87          * @param timestamp
88          */
89         public void setTimeStamp(long timestamp)
90         {
91             this.timestamp = timestamp;
92         }
93 
94 
95         /**
96          * Get timestamp of image
97          *
98          * @return timestamp of image
99          */
100        public long getTimestamp()
101        {
102            return this.timestamp;
103        }
104 
105 
106        /**
107         * Add position of roe in pixels.
108         *
109         * @param coordinate of roe in pixels
110         */
111        public void addRoePositionPixel(Coordinate coordinate)
112        {
113            this.roePositionPixels.add(coordinate);
114        }
115 
116 
117        /**
118         * Get list of roe postitions in pixels.
119         *
120         * @return list of roe positions in pixels.
121         */
122        public List getRoePositionPixelList()
123        {
124            return (List) this.roePositionPixels;
125        }
126 
127 
128        /**
129         * Add position of roe in millimeter.
130         *
131         * @param coordinate of roe in millimeter
132         */
133        public void addRoePositionMillimeter(Coordinate coordinate)
134        {
135            this.roePositionMillimeters.add(coordinate);
136        }
137 
138 
139        /**
140         * Get list of roe positions in millimeter.
141         *
142         * @return list of roe positions in millimeter.
143         */
144        public List getRoePositionMillimeterList()
145        {
146            return (List) this.roePositionMillimeters;
```

```
147         }
148 
149 
150         /**
151          * Set picture index in coordinate system
152          *
153          * @param pictureIndex of picture
154          */
155         public void setPictureIndex(int pictureIndex)
156         {
157             this.pictureIndex = pictureIndex;
158         }
159 
160 
161         /**
162          * Get index of picture in coordinatesystem
163          *
164          * @return pictureIndex in coordinatesystem
165          */
166         public int getPictureIndex()
167         {
168             return this.pictureIndex;
169         }
170 
171         /**
172          * Get the cameras field of view.
173          *
174          * @return returns field of view of camera (angle)
175          */
176         public int getFieldOfView()
177         {
178             return this.FOV;
179         }
180 
181         /**
182          * Get the distance between camera and surface in mm
183          *
184          * @return distance between camera and surface in mm
185          */
186         public float getDistance()
187         {
188             return this.captureHeight;
189         }
190     }
```

H.5 Path Optimalization

```java
package TSP;

import static java.lang.Math.abs;
import RoeRobot.Coordinate;

/**
 * This class does a Genetic Algorithm
 * @author Yngve
 */
public class GAnew {

    private static final double mutationRate = 0.01;
    private static final int tournamentSize = 9;
    private static final boolean elitism = true;

    public GAnew() {
    }

    /**
     * // Create a new population witch will contain the new and improved
     * chromosomes. // Select two parants for mating. This two can be selected
     * by random or select the fittest of the initial population Evolve for
     * given number of iterations.
     *
     * @param evolutions
     * @param pop
     * @return
     */
    public Population evolvePopulation(long evolutions, Population pop, Tour
    originalTour) {
        Population newPopulation = new Population(pop.getNrOfTours());
        Coordinate startCoord = pop.getFittest().getCoordinate(0);
        Tour currentFittestTour = pop.getFittest();


        // Result for stesting.
         double[] result;
        // Boolen for detectiong no improvement.
        boolean noImprovement = false;
        int noImprovemnetRange = 0;

        int cnt = 0;

        // Keep our best individual if elitism is enabled
        int elitismOffset = 0;
        if (elitism) {
            newPopulation.saveTour(0, pop.getFittest());
            newPopulation.saveTour(1, originalTour);
            elitismOffset = 2;
        }

        int j = 0;
        while (evolutions > j++ && !noImprovement) {
            for (int i = elitismOffset; i < newPopulation.getNrOfTours(); i++) {
                // Chose pair of chromosmes for amtiong
                Tour parentOne = this.turnamentSelection(pop);
                Tour parentTwo = this.turnamentSelection(pop);
                if (parentOne.getList().contains(null) ||
                parentTwo.getList().contains(null)) {
                    System.out.println("Fuck You parrent");
                }
                // Preforme crossover with Pc possebility
                // Tour child = this.crossover(parrentOne, parrentTwo,
                parrentOne.tourSize());
                Tour childOne = this.crossoverWithStartCoord(parentOne, parentTwo,
                parentOne.tourSize(), startCoord);
                Tour childTwo = this.crossoverWithStartCoord(parentOne, parentTwo,
                parentOne.tourSize(), startCoord);
                // System.out.println(startCoord);
                // Add child to new population
                newPopulation.saveTour(i++, childOne);
                newPopulation.saveTour(i, childTwo);
            }
```

```
69  //
70              // Mutate the new population a bit to add some new genetic material
71              for (int i = 0; i < newPopulation.getNrOfTours(); i++) {
72                  mutateWithStartCoord(newPopulation.getTour(i));
73              }
74
75
76              // Update population.
77              pop = newPopulation;
78
79          }
80
81          return newPopulation;
82      }
83
84      /**
85       * Croosover Arranged Crossover Will mace a child of two parrents. It tacker
86       * a random range of coordiantes and add them to a child. The missing
87       * coordinates are set by the other parrant by checkin if the child are
88       * missing a coordinate. If the child misses a coordinate it will be plased
89       * at an empti sopot.
90       */
91      private Tour crossover(Tour parrentOne, Tour parrentTwo, int tourSize) {
92          Tour child = new Tour(tourSize);
93          // Get start and end sub tour positions for parent1's tour
94          int startCrossoverPoint = (int) (abs(Math.random() * tourSize));
95          int endCrossoverPoint = (int) (abs(Math.random() * tourSize));
96
97          // Loop and add the sub tour from parent1 to our child
98          for (int i = 0; i < tourSize; i++) {
99              // If our start position is less than the end position
100             if (startCrossoverPoint < endCrossoverPoint && i > startCrossoverPoint
                && i < endCrossoverPoint) {
101                 child.setCoordinate(i, parrentOne.getCoordinate(i));
102             } // If our start position is larger
103             else if (startCrossoverPoint > endCrossoverPoint) {
104                 if (!(i < startCrossoverPoint && i > endCrossoverPoint)) {
105                     child.setCoordinate(i, parrentOne.getCoordinate(i));
106                 }
107             }
108         }
109         // If the child have an empty spot.
110         if (child.getList().contains(null)) {
111             // Loop throug all positions in the child
112             for (int i = 0; i < child.tourSize(); i++) {
113                 // Check if the position is empty
114                 if (child.getCoordinate(i) == null) {
115                     // Loop throug all coodinates holded by parrent two
116                     for (int k = 0; k < parrentTwo.tourSize(); k++) {
117                         // Find a coodrinate which the child are missing
118                         if (!child.containsCoordinate(parrentTwo.getCoordinate(k))) {
119                             // add the missing child.
120                             child.setCoordinate(i, parrentTwo.getCoordinate(k));
121                             break;
122                         }
123                     }
124                 }
125             }
126         }
127
128         return child;
129     }
130
131     /**
132      * Croosover Arranged Crossover
133      */
134     private Tour crossoverWithStartCoord(Tour parentOne, Tour parentTwo, int
        tourSize, Coordinate startCoord) {
135         Tour child = new Tour(tourSize);
136         child.setCoordinate(0, startCoord);
137         // Get start and end sub tour positions for parent1's tour
138         int startCrossoverPoint = (int) (abs(Math.random() * tourSize - 1) + 1);
139         int endCrossoverPoint = (int) (abs(Math.random() * tourSize - 1) + 1);
```

```
140
141         // Loop and add the sub tour from parent1 to our child
142         for (int i = 1; i < tourSize; i++) {
143             // If our start position is less than the end position
144             if (startCrossoverPoint < endCrossoverPoint && i > startCrossoverPoint
                && i < endCrossoverPoint) {
145                 child.setCoordinate(i, parentOne.getCoordinate(i));
146             } // If our start position is larger
147             else if (startCrossoverPoint > endCrossoverPoint) {
148                 if (!(i < startCrossoverPoint && i > endCrossoverPoint)) {
149                     child.setCoordinate(i, parentOne.getCoordinate(i));
150                 }
151             }
152         }
153         // Loop through all parentTwo's coordinates
154         for (int i = 1; i < child.tourSize(); i++) {
155             // If the child have an empty spot.
156             if (child.getList().contains(null)) {
157                 if (child.getCoordinate(i) == null) {
158                     for (int k = 0; k < parentTwo.tourSize(); k++) {
159                         if (!child.containsCoordinate(parentTwo.getCoordinate(k))) {
160                             child.setCoordinate(i, parentTwo.getCoordinate(k));
161                             break;
162                         }
163                     }
164                 }
165             }
166
167         }
168         if (child.getList().contains(null)) {
169             System.out.println("Fuck You child");
170             this.crossoverWithStartCoord(parentOne, parentTwo, tourSize, startCoord);
171         }
172         return child;
173     }
174
175     /**
176      * Mutate a tour using swap mutation The start coordinate will ceaped ass
177      * first in list.
178      */
179     private static void mutateWithStartCoord(Tour tour) {
180         // Loop through tour cities
181         for (int tourPos1 = 1; tourPos1 < tour.tourSize(); tourPos1++) {
182             // Apply mutation rate
183             double randNr = Math.random();
184
185             if (randNr < mutationRate) {
186                 // Get a second random position in the tour
187                 int tourPos2 = (int) (abs(Math.random() * tour.tourSize() - 1) + 1);
188                 // Get the coordinates at target position in tour
189                 Coordinate coord1 = tour.getCoordinate(tourPos1);
190                 Coordinate coord2 = tour.getCoordinate(tourPos2);
191                 // Swap them around
192                 tour.setCoordinate(tourPos2, coord1);
193                 tour.setCoordinate(tourPos1, coord2);
194                 //  System.out.println("I mutated the shit out of you. ");
195             }
196         }
197     }
198
199     /**
200      * *
201      * Rouletwheel selection.
202      */
203
204
205     /**
206      * Turnement selection will retun
207      */
208     private Tour turnamentSelection(Population pop) {
209
210         // Create a tournament population
211         Population turnamentPop = new Population(this.tournamentSize);
```

```
212             // For each place in the tournament get a random candidate tour and
213             // add it
214             for (int i = 0; i < this.tournamentSize; i++) {
215                 int randCromosomId = (int) (Math.random() * pop.getNrOfTours());
216                 // Return fittest tour for turnement.
217                 turnamentPop.saveTour(i, pop.getTour(randCromosomId));
218             }
219             return turnamentPop.getFittest();
220         }
221 
222     }
223 
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package TSP;

import java.util.ArrayList;
import RoeRobot.Coordinate;

/**
 *
 * @author Yngve
 */
public class NearestNeighbors {

    public NearestNeighbors() {

    }

    public Tour NearestNeighbors(ArrayList<Coordinate> coordinates) {
        ArrayList<Coordinate> coordListCopy = coordinates;
        Tour resultTour = new Tour(coordinates.size());
        Coordinate startCoord = coordinates.get(0);
        Coordinate currentCoord = null;
        resultTour.setCoordinate(0, startCoord);
        coordListCopy.remove(0);
        //Previus Distance


        for (int j = 1; j < resultTour.tourSize(); j++) {
                  double prevDist = Double.MAX_VALUE;
            for (int i = 0; i < coordListCopy.size(); i++) {
                Coordinate toCoord = coordListCopy.get(i);
                if (!resultTour.containsCoordinate(toCoord)) {
                    double deltaX = Math.abs(toCoord.getxCoord() -
                    startCoord.getxCoord());
                    double deltaY = Math.abs(toCoord.getyCoord() -
                    startCoord.getyCoord());
                    double distance = Math.sqrt(deltaX * deltaX + deltaY * deltaY);

                    if (distance <= prevDist) {
                        prevDist = distance;
                        currentCoord = toCoord;

                    }
                }
            }
            if(!(currentCoord == null)&&!resultTour.containsCoordinate(currentCoord)){
                resultTour.setCoordinate(j, currentCoord);
                coordListCopy.remove(currentCoord);
            } else {
                System.out.println("Coorinate do not excist");
            }
        }

        return resultTour;
    }
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package TSP;

import java.util.ArrayList;
import java.util.Stack;
import RoeRobot.Coordinate;

/**
 *
 * @author Yngve
 */
public class PatternOptimalization {

    // Ready to optimice the pattern
    private boolean ready;

    // Number of populations to generate !!can not be an odd number
    private int nrOfPopulations = 400;

    // Number of populations to generate !!can not be an odd number
    private int nrOfGenerations = 10;

    // Arraylist holding on all coordinates for a specific tour.
    private ArrayList<Coordinate> coordinatList;

    /**
     * Constructor
     */
    public PatternOptimalization() {
        this.coordinatList = new ArrayList<>();

    }

    /**
     *
     * @param coordinateList
     */
    public void addCoordinates(ArrayList coordinatesList) {
        this.coordinatList.addAll(coordinatesList);
    }

    /**
     * Add an coordinate ass start coodrinat.
     *
     * @param startCoordinate
     */
    public void addStartCoordinate(Coordinate startCoordinate) {
        this.coordinatList.add(0, startCoordinate);
    }

    /**
     *
     */
    public void setNrOfPopulations(int nrOfPopulations) {
        this.nrOfPopulations = nrOfPopulations;
    }

    public ArrayList doOptimizeNearestNeighbour(double xMilimerePerSec, double
    yMilimerePerSec) {
        // NearestNeigghbors Optimizer
        ArrayList<Coordinate> fittestTourList = new ArrayList<>();
        NearestNeighbors NN = new NearestNeighbors();
        Tour NNTour = NN.NearestNeighbors(this.coordinatList);
        System.out.println(NNTour.getTotalTime(xMilimerePerSec, yMilimerePerSec));

        return fittestTourList;
    }

    /**
```

```
73          *
74          * @param xMilimerePerSec
75          * @param yMilimerePerSec
76          * @return
77          */
78         public ArrayList doOptimalization(double xMilimerePerSec, double
            yMilimerePerSec) {
79
80             Tour originalTour = new Tour(coordinatList.size());
81             ArrayList<Coordinate> fittestTourList = new ArrayList<>();
82             if (this.coordinatList.isEmpty()) {
83                 System.out.println("No coordinates added!!!!");
84             } else {
85
86                 for (int i = 0; i < coordinatList.size(); i++) {
87                     originalTour.setCoordinate(i, coordinatList.get(i));
88                 }
89                 // System.out.println("Original tour: " + originalTour);
90                 double orginalTourTime = originalTour.getTotalTime(xMilimerePerSec,
                    yMilimerePerSec);
91                 System.out.println("Original tour tot time: " + orginalTourTime + " sec");
92                 System.out.println("Original tour Total distance: " +
                    originalTour.getTotalDistance());
93
94                 // If cordinates added.
95                 // Add Start coordinate.
96                 if (!this.coordinatList.isEmpty()) {
97                     // Generate populations.
98                     long startRand = System.currentTimeMillis();
99                     System.out.println("----------Start Random opt-----------");
100                    Population population = new Population(nrOfPopulations,
                       this.coordinatList, true);
101                    fittestTourList = population.getFittest().getList();
102                    long timeRandomHasUsed = (System.currentTimeMillis() - startRand);
103                    System.out.println("Time Random Opt has used:  " + timeRandomHasUsed
                       + " milli sec");
104                    System.out.println("Random path Total distance of fittest tour: " +
                       population.getFittest().getTotalDistance());
105                    double randomPathTime =
                       population.getFittest().getTotalTime(xMilimerePerSec,
                       yMilimerePerSec);
106                    System.out.println("Random path Fittest tour total time: " +
                       randomPathTime + " sec");
107                    System.out.println("---------Start GA---------");
108                    // To mashure the tame used by the GA
109                    long startGA = System.currentTimeMillis();
110                    // Start GA
111                    GAnew ga = new GAnew();
112                    population = ga.evolvePopulation(this.nrOfGenerations, population,
                       originalTour);
113    //                for (int j = 2; j < 400; j++) {
114    //                    startGA = System.currentTimeMillis();
115    //                    population = ga.evolvePopulation((j * 10), population,
       originalTour);//400, population);
116    //                    long timeGAHasUsed = (System.currentTimeMillis() - startGA);
117    //                    double afterGATime =
       population.getFittest().getTotalTime(xMilimerePerSec, yMilimerePerSec);
118    //                    System.out.println((j * 10) + " " + (afterGATime +
       timeGAHasUsed / 1000));
119    //                }
120                    long timeGAHasUsed = (System.currentTimeMillis() - startGA);
121                    long totalOptTime = (System.currentTimeMillis() - startRand);
122                    double afterGATime =
                       population.getFittest().getTotalTime(xMilimerePerSec,
                       yMilimerePerSec);
123
124                    System.out.println("Time GA has used:  " + timeGAHasUsed + " milli
                       sec");
125                    System.out.println("-------------------------");
126                    System.out.println("Time used for otimize:  " + totalOptTime + "
                       milli sec");
127                    System.out.println("-------------------------");
128                    System.out.println("Fittest form GA tour dist: " +
```

```
                    population.getFittest().getTotalDistance());
129                 System.out.println("Removing time from fittest GA tour: " +
                    afterGATime + " sec");
130                 System.out.println("Total time GA uses: " + (afterGATime +
                    timeGAHasUsed / 1000) + " sec.      Process time + removing time");
131                 System.out.println("\n" + "--------RESULT---------");
132                 System.out.println("Time between linitial tour and Random pattern:
                    " + ((orginalTourTime - randomPathTime - (timeRandomHasUsed /
                    1000))) + " sec");
133                 System.out.println("Random pattern improvement form Orginal:  " +
                    (((orginalTourTime - randomPathTime - (timeRandomHasUsed / 1000)) /
                    orginalTourTime) * 100) + " %");
134                 System.out.println("-------------------");
135
136                 System.out.println("Time between linitial tour and GA:  " +
                    ((orginalTourTime - afterGATime - (timeGAHasUsed / 1000))) + " sec");
137                 System.out.println("GA improvement form Orginal:  " +
                    (((orginalTourTime - afterGATime - (timeGAHasUsed / 1000)) /
                    orginalTourTime) * 100) + " %");
138                 System.out.println("Time between Random pattern and GA:  " +
                    (randomPathTime - afterGATime) + " sec");
139                 System.out.println("-------------------------");
140
141                 //System.out.println(population.getFittest());
142                 this.coordinatList.clear();
143             }
144         }
145         System.out.println("New List returned");
146
147
148         return fittestTourList;
149     }
150
151     /**
152      *
153      * @return
154      */
155     public synchronized boolean isReady() {
156         return ready;
157     }
158
159     /**
160      *
161      * @param ready
162      */
163     public synchronized void setReady(boolean ready) {
164         this.ready = ready;
165     }
166
167 }
168
```

```java

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package TSP;

import java.util.ArrayList;
import java.util.List;

/**
 * The Population will keep order of all the tours (chromosomes) The population
 * class will be able to - Add a tour to the list of tours. - Retrun a tour with
 * a spesific index number - Retrun the fittest tour. (the fittets tour is the
 * tour with the distace closest to the optimum solution. - Retrun the size of
 * the population.
 *
 * @author Yngve
 */
public class Population {

    // list of tours (chromosomes) in the population.
    private Tour[] tours;

    public Population(int populationSize, ArrayList coordList, boolean initialise) {
        this.tours = new Tour[populationSize];
        if (initialise) {
            for (int index = 0; index <= populationSize - 1; index++) {
                Tour tour = new Tour(coordList, true);
                this.tours[index] = tour;
            }
        }
    }

    /**
     * Generate empty population.
     *
     * @param populationSize
     */
    public Population(int populationSize) {
        this.tours = new Tour[populationSize];
    }

    /**
     * Adds a tour to the list of tours.
     *
     * @param tour
     */
    public void addTour(Tour tour) {
        this.tours[this.tours.length + 1] = tour;
    }

    // Saves a tour
    public void saveTour(int index, Tour tour) {
        tours[index] = tour;
    }

    /**
     * Get tour from the list of tours.
     *
     * @param index of the tour to return.
     * @return Tour
     */
    public Tour getTour(int index) {
        return this.tours[index];
    }

    /**
     * Get fittest tour in the population.
     *
     * @return fittest tour in the population.
     */
```

```
74      public Tour getFittest() {
75          Tour fittestTour = this.getTour(0);
76          for (int i = 0; i <= this.tours.length - 1; i++) {
77              if (fittestTour.getFitness() < this.getTour(i).getFitness()) {
78                  fittestTour = this.getTour(i);
79              }
80          }
81          return fittestTour;
82      }
83
84      /**
85       * Get fittest tour in the population.
86       *
87       * @return fittest tour in the population.
88       */
89      public Tour getSecoundFittest() {
90          Tour fittestTour = this.getTour(0);
91          Tour secoundFittestTour = this.getTour(0);
92          for (int i = 0; i <= this.tours.length - 1; i++) {
93              if (fittestTour.getFitness() < this.getTour(i).getFitness()) {
94                  fittestTour = this.getTour(i);
95              }
96          }
97          for (int i = 0; i <= this.tours.length - 1; i++) {
98              if (secoundFittestTour.getFitness() > fittestTour.getFitness() &&
                    secoundFittestTour.getFitness() < this.getTour(i).getFitness()) {
99                  secoundFittestTour = this.getTour(i);
100             }
101         }
102         return secoundFittestTour;
103     }
104
105     // List all tour ass string.
106     public String listAllTours() {
107         String allToursString = "There is no tours";
108         if (this.tours.length != 0) {
109             allToursString = "List of tours: \n";
110             for (int index = 0; index <= this.tours.length - 1; index++) {
111                 allToursString = allToursString +
                        this.tours[index].destinationsToString() + " " +
                        this.tours[index].getTotalDistance() + "\n";
112             }
113         }
114         return allToursString;
115     }
116
117     /**
118      * Get the number of tour in population.
119      *
120      *
121      */
122     public int getNrOfTours() {
123         return this.tours.length;
124     }
125
126 }
127
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package TSP;

import RoeRobot.Coordinate;
import java.util.ArrayList;
import java.util.Collections;

/**
 * Tour will represent one possible tour option. This tour it ganerated by
 * random The tour is be able to: - Store all the Position - Return a Position
 * with a spesifix index. - Add a Position to the tour. - Calculate thje
 * fittness for the tour example (1/totDistance) - Return the fittness of the
 * tour - Retrun number of Position in tour. -
 *
 * @author Yngve
 */
public class Tour {

    // Arraylist containing all destinations.
    private ArrayList<Coordinate> coordinates;
    private double fitness = 0;
    private double totalDistance = 0;
    private double totalTime = 0;

    /**
     * The constructor adds all the destinations to the list of destinations and
     * Shuffeles them randomly
     *
     * @param coordList
     * @param fillWithDestinations
     */
    public Tour(ArrayList coordList, boolean fillWithDestinations) {
        this.coordinates = new ArrayList<>();
        if (fillWithDestinations) {
            this.coordinates.addAll(coordList);
            this.shuffleDestinations();
            this.calcTotalDist();
        } else {
            for (int i = 0; i < coordList.size(); i++) {
                coordinates.add(null);
            }
        }
    }

    /**
     * Create empty tour
     */
    public Tour(int tourSize) {
        this.coordinates = new ArrayList<>();
        for (int i = 0; i < tourSize; i++) {
            coordinates.add(null);
        }
    }

    /**
     * Randomly shuffla all destinations except the start destination
     */
    private void shuffleDestinations() {
        // find start destinatio
        Coordinate dest = this.coordinates.get(0);
        this.coordinates.remove(dest);
        // Randomly reorder the tour
        Collections.shuffle(this.coordinates);
        this.coordinates.add(0, dest);
    }

    /**
     * Get the fitness of the tour.
     *
```

```
74          * @return fitness of the tour.
75          */
76         public double getFitness() {
77             this.calcTotalDist();
78             this.fitness = 0;
79             this.fitness = (1 / this.totalDistance);
80             return this.fitness;
81         }
82 
83         /**
84          * Get the totalt distance of the tour
85          *
86          * @return totalt distance of the tour
87          */
88         public double getTotalDistance() {
89             this.calcTotalDist();
90             return this.totalDistance;
91         }
92 
93         public double getTotalTime(double xMilimerePerSec, double yMilimerePerSec) {
94             this.calcTotalTime(xMilimerePerSec, yMilimerePerSec);
95             return this.totalTime;
96         }
97 
98         /**
99          * Get Start Coordiante
100         */
101        public Coordinate getStartCoordinate() {
102            return this.coordinates.get(0);
103        }
104 
105        /**
106         * Sets a city in a certain position within a tour
107         *
108         * @param tourPosition
109         * @param coordinate
110         */
111        public void setCoordinate(int tourPosition, Coordinate coordinate) {
112            this.coordinates.set(tourPosition, coordinate);
113            // If the tours been altered we need to reset the fitness and distance
114        }
115 
116 
117        // Get number of cities on our tour
118        public int tourSize() {
119            return this.coordinates.size();
120        }
121 
122        /**
123         * @param coodrinateId
124         * @return Coordinate with id given id number
125         */
126        public Coordinate getCoordinate(int coodrinateId) {
127            return this.coordinates.get(coodrinateId);
128        }
129 
130        // Check if the tour contains a city
131        public boolean containsCoordinate(Coordinate coordinte) {
132            return this.coordinates.contains(coordinte);
133        }
134 
135        public String destinationsToString() {
136            String destinationsString = "There are no destinations";
137            if (!this.coordinates.isEmpty()) {
138                destinationsString = "";
139                for (Coordinate destination : this.coordinates) {
140                    destinationsString = destinationsString + destination.toString();
141                }
142            }
143            return destinationsString;
144        }
145 
146        public ArrayList<Coordinate> getList() {
```

```
147             return this.coordinates;
148         }
149 
150         /**
151          * Calculate the totalt distance of the tour
152          */
153         private void calcTotalDist() {
154             this.totalDistance = 0;
155             for (int i = 0; i < this.coordinates.size() - 1; i++) {
156                 double fromX = this.coordinates.get(i).getxCoord();
157                 double fromY = this.coordinates.get(i).getyCoord();
158                 if (this.coordinates.contains(null)) {
159                     System.out.println("I Found Waldo");
160                 }
161                 double toX = this.coordinates.get(i + 1).getxCoord();
162                 double toY = this.coordinates.get(i + 1).getyCoord();
163                 double deltaX = Math.abs(toX - fromX);
164                 double deltaY = Math.abs(toY - fromY);
165 
166                 this.totalDistance = this.totalDistance + Math.sqrt(deltaX * deltaX +
                    deltaY * deltaY);
167             }
168         }
169 
170         /**
171          * Calculate total time of tour
172          */
173         private void calcTotalTime(double xMilimerePerSec, double yMilimerePerSec) {
174             this.totalTime = 0;
175             for (int i = 0; i < this.coordinates.size() - 1; i++) {
176                 double fromX = this.coordinates.get(i).getxCoord();
177                 double fromY = this.coordinates.get(i).getyCoord();
178                 if (this.coordinates.contains(null)) {
179                     System.out.println("I Found Waldo");
180                 }
181                 double toX = this.coordinates.get(i + 1).getxCoord();
182                 double toY = this.coordinates.get(i + 1).getyCoord();
183                 double deltaX = Math.abs(toX - fromX);
184                 double deltaY = Math.abs(toY - fromY);
185                 // To time
186                 double deltaTX = deltaX / xMilimerePerSec;
187                 double deltaTY = deltaY / yMilimerePerSec;
188 
189                 this.totalTime = (this.totalTime + Math.sqrt(deltaTX * deltaTX + deltaTY
                    * deltaTY));
190             }
191         }
192 
193         /**
194          *
195          * @return
196          */
197         @Override
198         public String toString() {
199             String geneString = " ";
200             for (int i = 0; i < this.coordinates.size(); i++) {
201                 geneString += this.getCoordinate(i).toString() + "\n ";
202             }
203             return geneString;
204         }
205 
206     }
207 
```

H.6 HMI

```java
package GPIO;

import com.pi4j.io.gpio.*;
import com.pi4j.io.gpio.event.GpioPinDigitalStateChangeEvent;
import com.pi4j.io.gpio.event.GpioPinListenerDigital;
import com.pi4j.platform.Platform;
import com.pi4j.platform.PlatformAlreadyAssignedException;
import com.pi4j.platform.PlatformManager;
import com.pi4j.util.CommandArgumentParser;
import RoeRobot.RoeRobotFasade;

/**
 * GPIO Pins for HMI
 * @author Yngve
 */
public class GPIO_HMI {

    // Input for buttons.
    private GpioPinDigitalInput startBtn;
    private GpioPinDigitalInput stopBtn;
    private GpioPinDigitalInput emergencyBtn;

    // Outpust for indicator lamps.
    private GpioPinDigitalOutput runningLamp;
    private GpioPinDigitalOutput faultLamp;

    // The roe analyser fasade
    RoeRobotFasade roeFasade;

    // GPIO litsener
    GpioPinListenerDigital startLitsener;
    GpioPinListenerDigital stopLitsener;
    GpioPinListenerDigital emergencyLitsener;

    // GPIO controller.
    private GpioController gpio;

    /**
     * Construcor
     *
     * @param roeFasade
     * @throws PlatformAlreadyAssignedException
     */
    public GPIO_HMI(RoeRobotFasade roeFasade) throws
    PlatformAlreadyAssignedException {
        this.roeFasade = roeFasade;

        PlatformManager.setPlatform(Platform.ODROID);

        //Create gpio controller
        this.gpio = GpioFactory.getInstance();

        // by default we will use gpio pin #01; however, if an argument
        // has been provided, then lookup the pin by address
        // Inputs for Start, stop and em stop buttons.
        Pin startBtnPin = CommandArgumentParser.getPin(OdroidXU4Pin.class,
        OdroidXU4Pin.GPIO_02);
        Pin stopBtnPin = CommandArgumentParser.getPin(OdroidXU4Pin.class,
        OdroidXU4Pin.GPIO_07);
        Pin emergencyBtnPin = CommandArgumentParser.getPin(OdroidXU4Pin.class,
        OdroidXU4Pin.GPIO_03);
        PinPullResistance pull =
        CommandArgumentParser.getPinPullResistance(PinPullResistance.PULL_UP);  //
        default pin pull resistance if no pull argument found

        // Outputs for indecator leds.
        Pin runningLampPin = CommandArgumentParser.getPin(OdroidXU4Pin.class,
        OdroidXU4Pin.GPIO_22);
        Pin faultLampPin = CommandArgumentParser.getPin(OdroidXU4Pin.class,
        OdroidXU4Pin.GPIO_26);

        // provision gpio pin as an input pin
        this.startBtn = gpio.provisionDigitalInputPin(startBtnPin, "Start button",
```

```
            pull);
66          this.stopBtn = gpio.provisionDigitalInputPin(stopBtnPin, "Start button",
            pull);
67          this.emergencyBtn = gpio.provisionDigitalInputPin(emergencyBtnPin, "Start
            button", pull);
68
69          // provision gpio pin as an output pin
70          this.runningLamp = gpio.provisionDigitalOutputPin(runningLampPin, "Running
            lamp: ");
71          this.faultLamp = gpio.provisionDigitalOutputPin(faultLampPin, "Fault lamp: ");
72          // Create listeners.
73          this.createLitseners();
74
75      }
76
77      /**
78       * Create litseners.
79       *
80       */
81      private void createLitseners() {
82          // create GPIO startLitsener
83          this.startLitsener = (GpioPinDigitalStateChangeEvent event) -> {
84              this.startSequens();
85              // display pin state on console
86              System.out.println(" --> GPIO PIN STATE CHANGE: " + event.getPin() + " =
                " + event.getState());
87          };
88          this.stopLitsener = (GpioPinDigitalStateChangeEvent event) -> {
89              this.stopSequens();
90              // display pin state on console
91              System.out.println(" --> GPIO PIN STATE CHANGE: " + event.getPin() + " =
                " + event.getState());
92          };
93          this.emergencyLitsener = (GpioPinDigitalStateChangeEvent event) -> {
94              this.emegencySequens();
95              // display pin state on console
96              System.out.println(" --> GPIO PIN STATE CHANGE: " + event.getPin() + " =
                " + event.getState());
97          };
98
99          // add buttonst to the startLitsener.
100         gpio.addListener(this.startLitsener, this.startBtn);
101         gpio.addListener(this.stopLitsener, this.stopBtn);
102         gpio.addListener(this.emergencyLitsener, this.emergencyBtn);
103     }
104
105     public void startSequens() {
106         this.roeFasade.startCycle();
107     }
108
109     public void stopSequens() {
110         this.roeFasade.stopCycle();
111     }
112
113     public void emegencySequens() {
114
115     }
116
117 }
118
```

```java
/*
 * THE GUI ROBOT GUI
 */
package GUI;

import ImageProcessing.ImageCaptureListener;
import ImageProcessing.RoeImage;
import RoeRobot.RoeRobotFasade;
import java.awt.Color;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import javax.imageio.ImageIO;
import javax.swing.table.TableColumn;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.videoio.VideoCapture;
import RoeRobot.Tray;
import RoeRobot.TrayProcessedListener;

/**
 * THE ROE ROBOT GUI
 *
 * @author Kristoffer
 */
public final class RoeBot extends javax.swing.JFrame  implements
ImageCaptureListener, TrayProcessedListener{

    /**
     * for dynamic panel
     */
    GridBagLayout layout = new GridBagLayout(); //setting the layout for dynamic
    panel
    GridBagConstraints c = new GridBagConstraints();
    RackAllClosed rackClosed;
    RackBottomOpen rackBottomOpen;
    RackMiddleOpen rackMiddleOpen;
    RackTopOpen rackTopOpen;
    RoeRobotFasade roeBotFasade;

    /**
     * for the camera capturing
     */
    DaemonThread daemonThread;
    private DaemonThread myThread = null;
    int count = 0;
    VideoCapture webSource = null;
    Mat frame = new Mat();
    MatOfByte mem = new MatOfByte();
    Loading loading = new Loading();


    /**
     * variables used for retrieving input
     */
    private int operationInterval;
    private int redLightVal;
    private int greenLightVal;
    private int blueLightVal;

    private int deadRoeInTray;

    //The current tray
    Tray workingTray;

    /**
     * Get image from camera when captured
     * @param capturedImage
```

```
72          */
73         @Override
74         public void noitfyImageCaptured(RoeImage capturedImage)
75         {
76             this.frame = capturedImage.getImage();
77         }
78 
79         /**
80          * Get information from tray after pictures are processed
81          * @param capturedImage
82          */
83         @Override
84         public void noitfyProcessingDone(Tray workingTray)
85         {
86             this.workingTray = workingTray;
87             this.updateTrayTable();
88 
89         }
90 
91 
92 
93 
94 
95         /**
96          * DaemonThread Class. Does not prevent JVM from exiting when the program
97          * finishes, but the thread is still running.
98          *
99          */
100        class DaemonThread implements Runnable {
101
102            protected volatile boolean runnable = false;
103
104            @Override
105            public void run() {
106                synchronized (this) {
107                    while (runnable) {
108                        if (webSource.grab()) {
109                            try {
110                                webSource.retrieve(frame);
111
112                                Imgcodecs.imencode(".bmp", frame, mem);
113                                //Highgui.imencode(".bmp", frame, mem);
114                                Image im = ImageIO.read(new
                                    ByteArrayInputStream(mem.toArray()));
115                                BufferedImage buff = (BufferedImage) im;
116                                Graphics g = CameraPanel.getGraphics();
117
118                                if (g.drawImage(buff, 0, 0, getWidth(), getHeight() -
                                    150, 0, 0, buff.getWidth(), buff.getHeight(), null)) {
119                                    if (runnable == false) {
120                                        System.out.println("Going to wait()");
121                                        this.wait();
122                                    }
123                                }
124
125                            } catch (Exception ex) {
126                                System.out.println("Error");
127                            }
128                        }
129                    }
130                }
131            }
132        }
133
134
135        /**
136         * Creates new form RoeBot
137         * @param roeBotFasade
138         */
139        public RoeBot(RoeRobotFasade roeBotFasade) {
140            initComponents(); //initializing components. work as a connection between
                GUI Editor and JAVA.
141
```

```
142             PanelReady.setVisible(false); //hiding the Ready panel to calibration button
                is pushed
143             NumberOfSearches.setVisible(false);
144             this.setNumberOfSearchesButton.setEnabled(false);
145             errorMessageSetTraysLabel.setVisible(false);
146
147             //Load the GUI windows
148             rackClosed = new RackAllClosed();
149             rackBottomOpen = new RackBottomOpen();
150             rackMiddleOpen = new RackMiddleOpen();
151             rackTopOpen = new RackTopOpen();
152
153             //Set the fasade
154             this.roeBotFasade = roeBotFasade;
155
156             gridBagConstraints();
157             //this.lightReg = lightReg;
158         }
159
160         private void gridBagConstraints() {
161             DynamicPanelCameraAndRack.setLayout(layout);
162             c.gridx = 0;
163             c.gridy = 0;
164             DynamicPanelCameraAndRack.add(rackClosed, c);
165             c.gridx = 0;
166             c.gridy = 0;
167             DynamicPanelCameraAndRack.add(CameraPanel, c);
168             c.gridx = 0;
169             c.gridy = 0;
170             DynamicPanelCameraAndRack.add(rackBottomOpen, c);
171             c.gridx = 0;
172             c.gridy = 0;
173             DynamicPanelCameraAndRack.add(rackTopOpen, c);
174             c.gridx = 0;
175             c.gridy = 0;
176             DynamicPanelCameraAndRack.add(rackMiddleOpen, c);
177         }
178
179         /**
180          * This method is called from within the constructor to initialize the form.
181          * WARNING: Do NOT modify this code. The content of this method is always
182          * regenerated by the Form Editor.
183          */
184         @SuppressWarnings("unchecked")
185         // <editor-fold defaultstate="collapsed" desc="Generated
            Code">//GEN-BEGIN:initComponents
186         private void initComponents() {
187
188             jLayeredPane1 = new javax.swing.JLayeredPane();
189             PanelCalibration = new javax.swing.JPanel();
190             btnCalibrate = new javax.swing.JButton();
191             pleaseCalibrateToContinueLabel = new javax.swing.JLabel();
192             lblRoeBot1 = new javax.swing.JLabel();
193             jLabel1 = new javax.swing.JLabel();
194             jLabel3 = new javax.swing.JLabel();
195             jLabel4 = new javax.swing.JLabel();
196             jLabel5 = new javax.swing.JLabel();
197             jLabel6 = new javax.swing.JLabel();
198             NumberOfSearches = new javax.swing.JPanel();
199             textFieldSetSearches = new javax.swing.JTextField();
200             setNumberOfSearchesButton = new javax.swing.JButton();
201             errorMessageSetTraysLabel = new javax.swing.JLabel();
202             PanelReady = new javax.swing.JPanel();
203             tgbSearchSystem = new javax.swing.JToggleButton();
204             tgbEmergencyStop = new javax.swing.JToggleButton();
205             btnReCalibrate = new javax.swing.JButton();
206             jScrollPane1 = new javax.swing.JScrollPane();
207             UpdateTable = new javax.swing.JTable();
208             lblRoeBot = new javax.swing.JLabel();
209             tgbLivePhoto = new javax.swing.JToggleButton();
210             btnLightRegulations = new javax.swing.JButton();
211             DynamicPanelCameraAndRack = new javax.swing.JPanel();
212             CameraPanel = new javax.swing.JPanel();
```

```
213             jTgbPause = new javax.swing.JToggleButton();
214
215             setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
216
217             jLayeredPane1.setMaximumSize(new java.awt.Dimension(1460, 700));
218             jLayeredPane1.setMinimumSize(new java.awt.Dimension(1460, 700));
219
220             PanelCalibration.setMaximumSize(new java.awt.Dimension(1460, 700));
221             PanelCalibration.setMinimumSize(new java.awt.Dimension(1460, 700));
222             PanelCalibration.setPreferredSize(new java.awt.Dimension(1460, 700));
223
224             btnCalibrate.setBackground(new java.awt.Color(255, 255, 255));
225             btnCalibrate.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
226             btnCalibrate.setText("Calibrate");
227             btnCalibrate.addActionListener(new java.awt.event.ActionListener() {
228                 public void actionPerformed(java.awt.event.ActionEvent evt) {
229                     btnCalibrateActionPerformed(evt);
230                 }
231             });
232
233             pleaseCalibrateToContinueLabel.setFont(new java.awt.Font("Verdana", 2, 12));
                // NOI18N
234             pleaseCalibrateToContinueLabel.setText("Please calibrate to continue");
235
236             lblRoeBot1.setFont(new java.awt.Font("Segoe UI Black", 3, 48)); // NOI18N
237             lblRoeBot1.setText("Welcome to the RoeBot");
238
239             jLabel1.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
240             jLabel1.setText("A project by: ");
241
242             jLabel3.setText("Yngve Bratthaug");
243
244             jLabel4.setText("Per Espen Aarseth");
245
246             jLabel5.setText("Kristian Andre Lilleindset");
247
248             jLabel6.setText("Kristoffer Hildrestrand");
249
250             textFieldSetSearches.setText("Operation interaval (min)");
251             textFieldSetSearches.addActionListener(new java.awt.event.ActionListener() {
252                 public void actionPerformed(java.awt.event.ActionEvent evt) {
253                     textFieldSetSearchesActionPerformed(evt);
254                 }
255             });
256
257             setNumberOfSearchesButton.setText("Start");
258             setNumberOfSearchesButton.addActionListener(new
                java.awt.event.ActionListener() {
259                 public void actionPerformed(java.awt.event.ActionEvent evt) {
260                     setNumberOfSearchesButtonActionPerformed(evt);
261                 }
262             });
263
264             errorMessageSetTraysLabel.setBackground(new java.awt.Color(255, 51, 51));
265             errorMessageSetTraysLabel.setForeground(new java.awt.Color(255, 51, 51));
266             errorMessageSetTraysLabel.setText("Please enter 1, 2 or 3");
267
268             javax.swing.GroupLayout NumberOfSearchesLayout = new
                javax.swing.GroupLayout(NumberOfSearches);
269             NumberOfSearches.setLayout(NumberOfSearchesLayout);
270             NumberOfSearchesLayout.setHorizontalGroup(
271
                    NumberOfSearchesLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
                    nt.LEADING)
272                     .addGroup(NumberOfSearchesLayout.createSequentialGroup()
273                         .addGap(9, 9, 9)
274                         .addComponent(textFieldSetSearches,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 138,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
275                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
276                         .addComponent(setNumberOfSearchesButton,
                            javax.swing.GroupLayout.DEFAULT_SIZE, 141, Short.MAX_VALUE)
277                         .addContainerGap())
```

```
278                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    NumberOfSearchesLayout.createSequentialGroup()
279                     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
280                     .addComponent(errorMessageSetTraysLabel)
281                     .addGap(105, 105, 105))
282             );
283             NumberOfSearchesLayout.setVerticalGroup(
284 
                    NumberOfSearchesLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
                    nt.LEADING)
285                 .addGroup(NumberOfSearchesLayout.createSequentialGroup()
286                     .addGap(29, 29, 29)
287 
                        .addGroup(NumberOfSearchesLayout.createParallelGroup(javax.swing.Group
                        Layout.Alignment.BASELINE)
288                         .addComponent(textFieldSetSearches,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 48,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
289                         .addComponent(setNumberOfSearchesButton,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 48,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
290                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        8, Short.MAX_VALUE)
291                     .addComponent(errorMessageSetTraysLabel,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 22,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
292             );
293 
294             javax.swing.GroupLayout PanelCalibrationLayout = new
                javax.swing.GroupLayout(PanelCalibration);
295             PanelCalibration.setLayout(PanelCalibrationLayout);
296             PanelCalibrationLayout.setHorizontalGroup(
297 
                    PanelCalibrationLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
                    nt.LEADING)
298                 .addGroup(PanelCalibrationLayout.createSequentialGroup()
299                     .addContainerGap(489, Short.MAX_VALUE)
300 
                        .addGroup(PanelCalibrationLayout.createParallelGroup(javax.swing.Group
                        Layout.Alignment.LEADING)
301                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                            PanelCalibrationLayout.createSequentialGroup()
302                             .addComponent(lblRoeBot1,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 642,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
303                             .addGap(329, 329, 329))
304                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                            PanelCalibrationLayout.createSequentialGroup()
305 
                                .addGroup(PanelCalibrationLayout.createParallelGroup(javax.swi
                                ng.GroupLayout.Alignment.LEADING)
306                                 .addComponent(btnCalibrate,
                                    javax.swing.GroupLayout.PREFERRED_SIZE, 244,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
307                                 .addGroup(PanelCalibrationLayout.createSequentialGroup()
308 
                                    .addGroup(PanelCalibrationLayout.createParallelGroup(j
                                    avax.swing.GroupLayout.Alignment.TRAILING)
309                                     .addComponent(NumberOfSearches,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
310                                     .addComponent(pleaseCalibrateToContinueLabel,
                                        javax.swing.GroupLayout.PREFERRED_SIZE, 280,
                                        javax.swing.GroupLayout.PREFERRED_SIZE))
311                                 .addGap(230, 230, 230)
312 
                                    .addGroup(PanelCalibrationLayout.createParallelGroup(j
                                    avax.swing.GroupLayout.Alignment.LEADING)
313                                     .addComponent(jLabel1,
                                        javax.swing.GroupLayout.PREFERRED_SIZE, 96,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                    .addComponent(jLabel3)
                                    .addComponent(jLabel4)
                                    .addComponent(jLabel5)
                                    .addComponent(jLabel6))))
                        .addGap(146, 146, 146))))
        );
        PanelCalibrationLayout.setVerticalGroup(

            PanelCalibrationLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
            nt.LEADING)
            .addGroup(PanelCalibrationLayout.createSequentialGroup()
                .addGap(29, 29, 29)
                .addComponent(lblRoeBot1, javax.swing.GroupLayout.PREFERRED_SIZE,
                117, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(27, 27, 27)
                .addComponent(pleaseCalibrateToContinueLabel,
                javax.swing.GroupLayout.PREFERRED_SIZE, 58,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(50, 50, 50)
                .addComponent(btnCalibrate, javax.swing.GroupLayout.PREFERRED_SIZE,
                114, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                137, Short.MAX_VALUE)

                .addGroup(PanelCalibrationLayout.createParallelGroup(javax.swing.Group
                Layout.Alignment.LEADING)
                    .addGroup(PanelCalibrationLayout.createSequentialGroup()
                        .addComponent(jLabel1,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 40,
                        javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel3)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel4)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel5)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel6))
                    .addComponent(NumberOfSearches,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(44, 44, 44))
        );

        lblRoeBot1.getAccessibleContext().setAccessibleName("Welcome to RoeBot");

        PanelReady.setMaximumSize(new java.awt.Dimension(1460, 700));
        PanelReady.setMinimumSize(new java.awt.Dimension(1460, 700));
        PanelReady.setPreferredSize(new java.awt.Dimension(1460, 700));

        tgbSearchSystem.setBackground(new java.awt.Color(51, 255, 0));
        tgbSearchSystem.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        tgbSearchSystem.setText("Search");
        tgbSearchSystem.setMaximumSize(new java.awt.Dimension(160, 80));
        tgbSearchSystem.setMinimumSize(new java.awt.Dimension(160, 80));
        tgbSearchSystem.setName(""); // NOI18N
        tgbSearchSystem.setPreferredSize(new java.awt.Dimension(160, 80));
        tgbSearchSystem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                tgbSearchSystemActionPerformed(evt);
            }
        });

        tgbEmergencyStop.setBackground(new java.awt.Color(255, 0, 0));
```

```
365             tgbEmergencyStop.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
366             tgbEmergencyStop.setText("Stop");
367             tgbEmergencyStop.setMaximumSize(new java.awt.Dimension(160, 80));
368             tgbEmergencyStop.setMinimumSize(new java.awt.Dimension(160, 80));
369             tgbEmergencyStop.setPreferredSize(new java.awt.Dimension(160, 80));
370             tgbEmergencyStop.addActionListener(new java.awt.event.ActionListener() {
371                 public void actionPerformed(java.awt.event.ActionEvent evt) {
372                     tgbEmergencyStopActionPerformed(evt);
373                 }
374             });
375
376             btnReCalibrate.setBackground(new java.awt.Color(255, 255, 255));
377             btnReCalibrate.setText("Recalibrate");
378             btnReCalibrate.addActionListener(new java.awt.event.ActionListener() {
379                 public void actionPerformed(java.awt.event.ActionEvent evt) {
380                     btnReCalibrateActionPerformed(evt);
381                 }
382             });
383
384             UpdateTable.setModel(new javax.swing.table.DefaultTableModel(
385                 new Object [][] {
386
387                 },
388                 new String [] {
389                     "Tray number", "Removed roes"
390                 }
391             ) {
392                 Class[] types = new Class [] {
393                     java.lang.Integer.class, java.lang.Integer.class
394                 };
395
396                 public Class getColumnClass(int columnIndex) {
397                     return types [columnIndex];
398                 }
399             });
400             UpdateTable.getTableHeader().setReorderingAllowed(false);
401             jScrollPane1.setViewportView(UpdateTable);
402
403             lblRoeBot.setFont(new java.awt.Font("Segoe UI Black", 3, 48)); // NOI18N
404             lblRoeBot.setText("RoeBot");
405
406             tgbLivePhoto.setBackground(new java.awt.Color(255, 255, 255));
407             tgbLivePhoto.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
408             tgbLivePhoto.setText("Live photo");
409             tgbLivePhoto.addActionListener(new java.awt.event.ActionListener() {
410                 public void actionPerformed(java.awt.event.ActionEvent evt) {
411                     tgbLivePhotoActionPerformed(evt);
412                 }
413             });
414
415             btnLightRegulations.setBackground(new java.awt.Color(255, 255, 255));
416             btnLightRegulations.setText("Light Regulation");
417             btnLightRegulations.addActionListener(new java.awt.event.ActionListener() {
418                 public void actionPerformed(java.awt.event.ActionEvent evt) {
419                     btnLightRegulationsActionPerformed(evt);
420                 }
421             });
422
423             javax.swing.GroupLayout CameraPanelLayout = new
                javax.swing.GroupLayout(CameraPanel);
424             CameraPanel.setLayout(CameraPanelLayout);
425             CameraPanelLayout.setHorizontalGroup(
426
                    CameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
                    ADING)
427                 .addGap(0, 573, Short.MAX_VALUE)
428             );
429             CameraPanelLayout.setVerticalGroup(
430
                    CameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
                    ADING)
431                 .addGap(0, 429, Short.MAX_VALUE)
432             );
```

```
433
434         javax.swing.GroupLayout DynamicPanelCameraAndRackLayout = new
            javax.swing.GroupLayout(DynamicPanelCameraAndRack);
435         DynamicPanelCameraAndRack.setLayout(DynamicPanelCameraAndRackLayout);
436         DynamicPanelCameraAndRackLayout.setHorizontalGroup(
437
                DynamicPanelCameraAndRackLayout.createParallelGroup(javax.swing.GroupLayou
                t.Alignment.LEADING)
438             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                DynamicPanelCameraAndRackLayout.createSequentialGroup()
439                 .addContainerGap(38, Short.MAX_VALUE)
440                 .addComponent(CameraPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
441                 .addContainerGap())
442         );
443         DynamicPanelCameraAndRackLayout.setVerticalGroup(
444
                DynamicPanelCameraAndRackLayout.createParallelGroup(javax.swing.GroupLayou
                t.Alignment.LEADING)
445             .addGroup(DynamicPanelCameraAndRackLayout.createSequentialGroup()
446                 .addContainerGap()
447                 .addComponent(CameraPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
448                 .addContainerGap())
449         );
450
451         jTgbPause.setBackground(new java.awt.Color(255, 255, 0));
452         jTgbPause.setFont(new java.awt.Font("TakaoPGothic", 1, 14)); // NOI18N
453         jTgbPause.setText("Pause");
454         jTgbPause.setToolTipText("");
455         jTgbPause.addActionListener(new java.awt.event.ActionListener() {
456             public void actionPerformed(java.awt.event.ActionEvent evt) {
457                 jTgbPauseActionPerformed(evt);
458             }
459         });
460
461         javax.swing.GroupLayout PanelReadyLayout = new
            javax.swing.GroupLayout(PanelReady);
462         PanelReady.setLayout(PanelReadyLayout);
463         PanelReadyLayout.setHorizontalGroup(
464
                PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
                DING)
465             .addGroup(PanelReadyLayout.createSequentialGroup()
466                 .addGap(74, 74, 74)
467
                    .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                    .Alignment.TRAILING)
468                     .addComponent(DynamicPanelCameraAndRack,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
469                     .addGroup(PanelReadyLayout.createSequentialGroup()
470                         .addComponent(tgbSearchSystem,
                            javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
471                         .addGap(39, 39, 39)
472                         .addComponent(jTgbPause,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 165,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
473                         .addGap(27, 27, 27)
474                         .addComponent(tgbEmergencyStop,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 160,
                            javax.swing.GroupLayout.PREFERRED_SIZE)))
475                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                    245, Short.MAX_VALUE)
476
                    .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                    .Alignment.LEADING)
477                     .addGroup(PanelReadyLayout.createSequentialGroup()
478                         .addComponent(btnLightRegulations,
```

```
                        javax.swing.GroupLayout.PREFERRED_SIZE, 190,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(90, 90, 90)
                        .addComponent(btnReCalibrate,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 172,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jScrollPane1,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(65, 65, 65))
            .addGroup(PanelReadyLayout.createSequentialGroup()
                .addGap(293, 293, 293)
                .addComponent(tgbLivePhoto, javax.swing.GroupLayout.PREFERRED_SIZE,
                151, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(210, 210, 210)
                .addComponent(lblRoeBot, javax.swing.GroupLayout.PREFERRED_SIZE,
                224, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                Short.MAX_VALUE))
        );
        PanelReadyLayout.setVerticalGroup(

            PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
            DING)
            .addGroup(PanelReadyLayout.createSequentialGroup()
                .addContainerGap()

                .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                .Alignment.BASELINE)
                    .addComponent(lblRoeBot, javax.swing.GroupLayout.PREFERRED_SIZE,
                    117, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(tgbLivePhoto,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 75,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                .Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    PanelReadyLayout.createSequentialGroup()
                        .addComponent(DynamicPanelCameraAndRack,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(115, 115, 115))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    PanelReadyLayout.createSequentialGroup()
                        .addComponent(jScrollPane1,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 350,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(90, 90, 90)

                        .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.Gro
                        upLayout.Alignment.LEADING)

                            .addGroup(PanelReadyLayout.createParallelGroup(javax.swing
                            .GroupLayout.Alignment.BASELINE)
                                .addComponent(tgbEmergencyStop,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 81,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(tgbSearchSystem,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jTgbPause,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 80,
                                javax.swing.GroupLayout.PREFERRED_SIZE))

                            .addGroup(PanelReadyLayout.createParallelGroup(javax.swing
                            .GroupLayout.Alignment.BASELINE)
                                .addComponent(btnLightRegulations,
```

```
                                javax.swing.GroupLayout.PREFERRED_SIZE, 74,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(btnReCalibrate,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 70,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(9, 9, 9))))
        );

        jLayeredPane1.setLayer(PanelCalibration,
        javax.swing.JLayeredPane.DEFAULT_LAYER);
        jLayeredPane1.setLayer(PanelReady, javax.swing.JLayeredPane.DEFAULT_LAYER);

        javax.swing.GroupLayout jLayeredPane1Layout = new
        javax.swing.GroupLayout(jLayeredPane1);
        jLayeredPane1.setLayout(jLayeredPane1Layout);
        jLayeredPane1Layout.setHorizontalGroup(

            jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
            .addGroup(jLayeredPane1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(PanelCalibration,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE))

            .addGroup(jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
                .addGroup(jLayeredPane1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(PanelReady,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)))
        );
        jLayeredPane1Layout.setVerticalGroup(

            jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
            .addComponent(PanelCalibration,
            javax.swing.GroupLayout.Alignment.TRAILING,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGroup(jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                jLayeredPane1Layout.createSequentialGroup()
                    .addComponent(PanelReady,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())))
        );

        javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        pack();
```

```
552         }// </editor-fold>//GEN-END:initComponents
553
554         /**
555          * TODO: call Loading and loading image is going to circulate to finished
556          * calibrating
557          *
558          * @param evt
559          */
560         private void btnCalibrateActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_btnCalibrateActionPerformed
561             btnCalibrate.setBackground(Color.white);
562             loading.setVisible(true);
563             NumberOfSearches.setVisible(true);
564             btnCalibrate.setVisible(false);
565             pleaseCalibrateToContinueLabel.setVisible(false);
566             roeBotFasade.doCalibrate();
567
568             //TODO: if calibration done, then set Loading not visible. and opens
569             //setSearches-panel, and then you can put in number of searches.
570             //the "ready"-panel with number of found trays information.
571             //PanelReady.setVisible(true);
572             //PanelCalibration.setVisible(false);
573         }//GEN-LAST:event_btnCalibrateActionPerformed
574
575         /**
576          * TOGGLEBUTTON, will start the system and cycle it.
577          *
578          * @param evt
579          */
580         private void tgbSearchSystemActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_tgbSearchSystemActionPerformed
581             tgbSearchSystem.setBackground(Color.GREEN);
582             if (tgbSearchSystem.isSelected()) {
583                 roeBotFasade.startCycle();
584                 tgbSearchSystem.setText("DestROEing");
585                 tgbEmergencyStop.setText("Stop");
586             }
587
588         }//GEN-LAST:event_tgbSearchSystemActionPerformed
589
590
591         private void tgbEmergencyStopActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_tgbEmergencyStopActionPerformed
592             System.out.println("Stop pressed!");
593             tgbEmergencyStop.setVisible(true);
594             tgbEmergencyStop.setBackground(Color.red);
595
596             if (tgbSearchSystem.isSelected()) {
597                 tgbEmergencyStop.setVisible(true);
598                 roeBotFasade.stopRobot();
599                 tgbEmergencyStop.setText("STOPPED!");
600                 tgbSearchSystem.setText("Search");
601             }
602         }//GEN-LAST:event_tgbEmergencyStopActionPerformed
603
604         private void btnReCalibrateActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_btnReCalibrateActionPerformed
605             // TODO add your handling code here:
606             roeBotFasade.doCalibrate();
607         }//GEN-LAST:event_btnReCalibrateActionPerformed
608
609         /**
610          * TOGGLE BUTTON, activating the photo if pushed. Else rack system will
611          * appear.
612          *
613          * @param evt toggleButton
614          */
615         private void tgbLivePhotoActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_tgbLivePhotoActionPerformed
616             //TODO: Change tgbLivePhoto.isSelected() to when it is two or three racks
                that show
617             //right photo.
618
```

```
619             if (tgbLivePhoto.isSelected()) {
620                 rackClosed.getClosedRack().setVisible(false);
621                 CameraPanel.setVisible(true);
622                 tgbLivePhoto.setText("Rack Update");
623                 webSource = new VideoCapture(0);
624                 myThread = new DaemonThread();
625                 Thread t = new Thread(myThread);
626                 t.setDaemon(true);
627                 myThread.runnable = true;
628                 t.start();
629             } else {
630                 tgbLivePhoto.setText("Live Photo");
631                 //panelCamera.setVisible(false);
632                 CameraPanel.setVisible(false);
633                 rackClosed.getClosedRack().setVisible(true);
634                 myThread.runnable = false;
635                 webSource.release();
636                 Thread t = new Thread(myThread);
637                 t.setDaemon(false);
638                 myThread.runnable = false;
639                 t.interrupt();
640             }
641         }//GEN-LAST:event_tgbLivePhotoActionPerformed
642 
643 
644         private void btnLightRegulationsActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_btnLightRegulationsActionPerformed
645             LightRegulations lightRegulator = new LightRegulations();
646             // display window
647             lightRegulator.setVisible(true);
648 
649             // get values from the window
650             this.redLightVal = lightRegulator.getRedLightValue();
651             this.greenLightVal = lightRegulator.getGreenLightValue();
652             this.blueLightVal = lightRegulator.getBlueLightValue();
653 
654 
655             // Update the fasade with new values
656             this.roeBotFasade.regulateLights(this.redLightVal, this.greenLightVal,
                this.blueLightVal);
657             System.out.println("endra farger til:"+ redLightVal +" , "+ greenLightVal +
                " , "+blueLightVal);
658 
659         }//GEN-LAST:event_btnLightRegulationsActionPerformed
660 
661         private void setNumberOfSearchesButtonActionPerformed(java.awt.event.ActionEvent
            evt) {//GEN-FIRST:event_setNumberOfSearchesButtonActionPerformed
662 
663 
664 
665 
666                 this.roeBotFasade.setSearchInterval(this.operationInterval);
667                 this.loading.setVisible(false);
668                 this.PanelCalibration.setVisible(false);
669                 this.PanelReady.setVisible(true);
670                 this.setNumberOfSearchesButton.setEnabled(true);
671                 this.UpdateTable.setVisible(true);
672 
673 
674         }//GEN-LAST:event_setNumberOfSearchesButtonActionPerformed
675 
676         private void textFieldSetSearchesActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_textFieldSetSearchesActionPerformed
677 
678             boolean numberInput = false;
679             //Wait for numberinput
680             while(!numberInput)
681                 {
682             String input = textFieldSetSearches.getText();
683             //if 2 or 3 is typed
684             //then choose right image in dynamic panel
685             if (input != null)
686             {
```

```
687                 try{
688                 this.operationInterval = Integer.parseInt(input);
689                 numberInput = true;
690                 }
691                 catch(NumberFormatException ex)
692                 {
693                     System.out.println("må vårrå ett tall");
694                     numberInput = false;
695                 }
696                 this.setNumberOfSearchesButton.setEnabled(true);
697             }
698             else
699             {
700                 this.setNumberOfSearchesButton.setEnabled(false);
701             }
702             }
703         }//GEN-LAST:event_textFieldSetSearchesActionPerformed
704 
705         private void jTgbPauseActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_jTgbPauseActionPerformed
706             //
707             if (jTgbPause.isSelected()) {
708                 System.out.println("Pause is pressed");
709                 // check the state of the pause button, pause or continue
710                 if (jTgbPause.getText().equals("Pause")) {
711                     System.out.println("Pause is Button");
712                     // pause robot, change button to continue button
713                     jTgbPause.setText("Continue");
714                     jTgbPause.setBackground(Color.green);
715                     roeBotFasade.pauseRobot();
716                 }
717                 //
718                 else if (jTgbPause.getText().equals("Continue")) {
719                     System.out.println("Continue is Button");
720                     // continue robot, change button state to pause
721                     this.roeBotFasade.continueRobot();
722                     jTgbPause.setText("Pause");
723                     jTgbPause.setBackground(Color.yellow);
724                 }
725             }
726 
727         }//GEN-LAST:event_jTgbPauseActionPerformed
728 
729 
730         private void updateTrayTable() {
731             TableColumn coloumn;
732             Component comp;
733 
734         }
735 
736 
737 
738         // Variables declaration - do not modify//GEN-BEGIN:variables
739         private javax.swing.JPanel CameraPanel;
740         private javax.swing.JPanel DynamicPanelCameraAndRack;
741         private javax.swing.JPanel NumberOfSearches;
742         public javax.swing.JPanel PanelCalibration;
743         public javax.swing.JPanel PanelReady;
744         public javax.swing.JTable UpdateTable;
745         private javax.swing.JButton btnCalibrate;
746         public javax.swing.JButton btnLightRegulations;
747         private javax.swing.JButton btnReCalibrate;
748         private javax.swing.JLabel errorMessageSetTraysLabel;
749         private javax.swing.JLabel jLabel1;
750         private javax.swing.JLabel jLabel3;
751         private javax.swing.JLabel jLabel4;
752         private javax.swing.JLabel jLabel5;
753         private javax.swing.JLabel jLabel6;
754         private javax.swing.JLayeredPane jLayeredPane1;
755         private javax.swing.JScrollPane jScrollPane1;
756         public javax.swing.JToggleButton jTgbPause;
757         private javax.swing.JLabel lblRoeBot;
758         private javax.swing.JLabel lblRoeBot1;
```

```
759         private javax.swing.JLabel pleaseCalibrateToContinueLabel;
760         private javax.swing.JButton setNumberOfSearchesButton;
761         private javax.swing.JTextField textFieldSetSearches;
762         public javax.swing.JToggleButton tgbEmergencyStop;
763         private javax.swing.JToggleButton tgbLivePhoto;
764         public javax.swing.JToggleButton tgbSearchSystem;
765         // End of variables declaration//GEN-END:variables
766     }
767
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package GUI;

import javax.swing.JLabel;

/**
 * Loading fish
 * @author Laptop1
 */
public class Loading extends javax.swing.JFrame {

    /**
     * Creates new form Loading
     */
    public Loading() {
        initComponents();
        setLocation(460, 190);
        setAlwaysOnTop(true);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        labelLoadingFish = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        labelLoadingFish.setIcon(new
        javax.swing.ImageIcon(getClass().getResource("/mainFrame/images/giphy.gif")));
         // NOI18N
        labelLoadingFish.setToolTipText("");

        javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(labelLoadingFish, javax.swing.GroupLayout.PREFERRED_SIZE,
            476, javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(labelLoadingFish, javax.swing.GroupLayout.PREFERRED_SIZE,
            302, javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
        (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
        default look and feel.
         * For details see
         http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
```

```
65          try {
66              for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
67                  if ("Nimbus".equals(info.getName())) {
68                      javax.swing.UIManager.setLookAndFeel(info.getClassName());
69                      break;
70                  }
71              }
72          } catch (ClassNotFoundException ex) {
73
                java.util.logging.Logger.getLogger(Loading.class.getName()).log(java.util.
                logging.Level.SEVERE, null, ex);
74          } catch (InstantiationException ex) {
75
                java.util.logging.Logger.getLogger(Loading.class.getName()).log(java.util.
                logging.Level.SEVERE, null, ex);
76          } catch (IllegalAccessException ex) {
77
                java.util.logging.Logger.getLogger(Loading.class.getName()).log(java.util.
                logging.Level.SEVERE, null, ex);
78          } catch (javax.swing.UnsupportedLookAndFeelException ex) {
79
                java.util.logging.Logger.getLogger(Loading.class.getName()).log(java.util.
                logging.Level.SEVERE, null, ex);
80          }
81          //</editor-fold>
82
83          /* Create and display the form */
84          java.awt.EventQueue.invokeLater(new Runnable() {
85              public void run() {
86                  new Loading().setVisible(true);
87              }
88          });
89      }
90
91      // Variables declaration - do not modify//GEN-BEGIN:variables
92      private javax.swing.JLabel labelLoadingFish;
93      // End of variables declaration//GEN-END:variables
94  }
95
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package GUI;

import javax.swing.JLabel;

/**
 * Show all trays closed in rack
 * @author Kristoffer
 */
public class RackAllClosed extends javax.swing.JPanel {

    /**
     * Creates new form RackAllClosed
     */
    public RackAllClosed() {
        initComponents();
    }
    public JLabel getClosedRack()
    {
        return rackClosedLabel;
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        rackClosedLabel = new javax.swing.JLabel();

        rackClosedLabel.setIcon(new
        javax.swing.ImageIcon(getClass().getResource("/mainFrame/images/rack-system-th
        ree-allclosed.jpg"))); // NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(rackClosedLabel)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(rackClosedLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
    }// </editor-fold>//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel rackClosedLabel;
    // End of variables declaration//GEN-END:variables
}

```

```java
package GUI;

import javax.swing.JLabel;

/**
 * Show middle rack open
 * @author Laptop1
 */
public class RackMiddleOpen extends javax.swing.JPanel {

    /**
     * Creates new form RackMiddleOpen
     */
    public RackMiddleOpen() {
        initComponents();
    }

    public JLabel getMiddleOpenRack() {
        return jLabel1;
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();

        jLabel1.setIcon(new
        javax.swing.ImageIcon(getClass().getResource("/mainFrame/images/rack-system-th
        ree-middleOpen.jpg"))); // NOI18N
        jLabel1.setPreferredSize(new java.awt.Dimension(400, 375));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 247,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
    }// </editor-fold>//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
}

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package GUI;

import javax.swing.JLabel;

/**
 * Show top tray open
 * @author Laptop1
 */
public class RackTopOpen extends javax.swing.JPanel {

    /**
     * Creates new form RackTopOpen
     */
    public RackTopOpen() {
        initComponents();
    }

    public JLabel getTopOpenRack() {
        return jLabel1;
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();

        jLabel1.setIcon(new
        javax.swing.ImageIcon(getClass().getResource("/mainFrame/images/rack-system-th
        ree-topOpen.jpg"))); // NOI18N
        jLabel1.setPreferredSize(new java.awt.Dimension(400, 375));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 251,
            Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
    }// </editor-fold>//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
}

```

```java
package GUI;

import javax.swing.JLabel;

/**
 * Show bottom tray open in rack
 * @author Kristoffer
 */
public class RackBottomOpen extends javax.swing.JPanel {

    /**
     * Creates new form RackBottomOpen
     */
    public RackBottomOpen() {
        initComponents();
    }

    public JLabel getBottomOpenRack()
    {
        return jLabel1;
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();

        jLabel1.setIcon(new
        javax.swing.ImageIcon(getClass().getResource("/mainFrame/images/rack-system-th
        ree-bottomOpen.jpg"))); // NOI18N
        jLabel1.setPreferredSize(new java.awt.Dimension(400, 375));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 249,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
    }// </editor-fold>//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
}

```

```java
/*
 * THE GUI ROBOT GUI
 */
package GUI;

import ImageProcessing.ImageCaptureListener;
import ImageProcessing.RoeImage;
import RoeRobot.RoeRobotFasade;
import java.awt.Color;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import javax.imageio.ImageIO;
import javax.swing.table.TableColumn;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.videoio.VideoCapture;
import RoeRobot.Tray;
import RoeRobot.TrayProcessedListener;

/**
 * THE ROE ROBOT GUI
 *
 * @author Kristoffer
 */
public final class RoeBot extends javax.swing.JFrame  implements
ImageCaptureListener, TrayProcessedListener{

    /**
     * for dynamic panel
     */
    GridBagLayout layout = new GridBagLayout(); //setting the layout for dynamic
    panel
    GridBagConstraints c = new GridBagConstraints();
    RackAllClosed rackClosed;
    RackBottomOpen rackBottomOpen;
    RackMiddleOpen rackMiddleOpen;
    RackTopOpen rackTopOpen;
    RoeRobotFasade roeBotFasade;

    /**
     * for the camera capturing
     */
    DaemonThread daemonThread;
    private DaemonThread myThread = null;
    int count = 0;
    VideoCapture webSource = null;
    Mat frame = new Mat();
    MatOfByte mem = new MatOfByte();
    Loading loading = new Loading();


    /**
     * variables used for retrieving input
     */
    private int operationInterval;
    private int redLightVal;
    private int greenLightVal;
    private int blueLightVal;

    private int deadRoeInTray;

    //The current tray
    Tray workingTray;

    /**
     * Get image from camera when captured
     * @param capturedImage
```

```
72          */
73         @Override
74         public void noitfyImageCaptured(RoeImage capturedImage)
75         {
76             this.frame = capturedImage.getImage();
77         }
78 
79         /**
80          * Get information from tray after pictures are processed
81          * @param capturedImage
82          */
83         @Override
84         public void noitfyProcessingDone(Tray workingTray)
85         {
86             this.workingTray = workingTray;
87             this.updateTrayTable();
88 
89         }
90 
91 
92 
93 
94 
95         /**
96          * DaemonThread Class. Does not prevent JVM from exiting when the program
97          * finishes, but the thread is still running.
98          *
99          */
100        class DaemonThread implements Runnable {
101
102            protected volatile boolean runnable = false;
103
104            @Override
105            public void run() {
106                synchronized (this) {
107                    while (runnable) {
108                        if (webSource.grab()) {
109                            try {
110                                webSource.retrieve(frame);
111
112                                Imgcodecs.imencode(".bmp", frame, mem);
113                                //Highgui.imencode(".bmp", frame, mem);
114                                Image im = ImageIO.read(new
                                    ByteArrayInputStream(mem.toArray()));
115                                BufferedImage buff = (BufferedImage) im;
116                                Graphics g = CameraPanel.getGraphics();
117
118                                if (g.drawImage(buff, 0, 0, getWidth(), getHeight() -
                                    150, 0, 0, buff.getWidth(), buff.getHeight(), null)) {
119                                    if (runnable == false) {
120                                        System.out.println("Going to wait()");
121                                        this.wait();
122                                    }
123                                }
124
125                            } catch (Exception ex) {
126                                System.out.println("Error");
127                            }
128                        }
129                    }
130                }
131            }
132        }
133
134
135        /**
136         * Creates new form RoeBot
137         * @param roeBotFasade
138         */
139        public RoeBot(RoeRobotFasade roeBotFasade) {
140            initComponents(); //initializing components. work as a connection between
                GUI Editor and JAVA.
141
```

```
142             PanelReady.setVisible(false); //hiding the Ready panel to calibration button
                is pushed
143             NumberOfSearches.setVisible(false);
144             this.setNumberOfSearchesButton.setEnabled(false);
145             errorMessageSetTraysLabel.setVisible(false);
146
147             //Load the GUI windows
148             rackClosed = new RackAllClosed();
149             rackBottomOpen = new RackBottomOpen();
150             rackMiddleOpen = new RackMiddleOpen();
151             rackTopOpen = new RackTopOpen();
152
153             //Set the fasade
154             this.roeBotFasade = roeBotFasade;
155
156             gridBagConstraints();
157             //this.lightReg = lightReg;
158         }
159
160         private void gridBagConstraints() {
161             DynamicPanelCameraAndRack.setLayout(layout);
162             c.gridx = 0;
163             c.gridy = 0;
164             DynamicPanelCameraAndRack.add(rackClosed, c);
165             c.gridx = 0;
166             c.gridy = 0;
167             DynamicPanelCameraAndRack.add(CameraPanel, c);
168             c.gridx = 0;
169             c.gridy = 0;
170             DynamicPanelCameraAndRack.add(rackBottomOpen, c);
171             c.gridx = 0;
172             c.gridy = 0;
173             DynamicPanelCameraAndRack.add(rackTopOpen, c);
174             c.gridx = 0;
175             c.gridy = 0;
176             DynamicPanelCameraAndRack.add(rackMiddleOpen, c);
177         }
178
179         /**
180          * This method is called from within the constructor to initialize the form.
181          * WARNING: Do NOT modify this code. The content of this method is always
182          * regenerated by the Form Editor.
183          */
184         @SuppressWarnings("unchecked")
185         // <editor-fold defaultstate="collapsed" desc="Generated
            Code">//GEN-BEGIN:initComponents
186         private void initComponents() {
187
188             jLayeredPane1 = new javax.swing.JLayeredPane();
189             PanelCalibration = new javax.swing.JPanel();
190             btnCalibrate = new javax.swing.JButton();
191             pleaseCalibrateToContinueLabel = new javax.swing.JLabel();
192             lblRoeBot1 = new javax.swing.JLabel();
193             jLabel1 = new javax.swing.JLabel();
194             jLabel3 = new javax.swing.JLabel();
195             jLabel4 = new javax.swing.JLabel();
196             jLabel5 = new javax.swing.JLabel();
197             jLabel6 = new javax.swing.JLabel();
198             NumberOfSearches = new javax.swing.JPanel();
199             textFieldSetSearches = new javax.swing.JTextField();
200             setNumberOfSearchesButton = new javax.swing.JButton();
201             errorMessageSetTraysLabel = new javax.swing.JLabel();
202             PanelReady = new javax.swing.JPanel();
203             tgbSearchSystem = new javax.swing.JToggleButton();
204             tgbEmergencyStop = new javax.swing.JToggleButton();
205             btnReCalibrate = new javax.swing.JButton();
206             jScrollPane1 = new javax.swing.JScrollPane();
207             UpdateTable = new javax.swing.JTable();
208             lblRoeBot = new javax.swing.JLabel();
209             tgbLivePhoto = new javax.swing.JToggleButton();
210             btnLightRegulations = new javax.swing.JButton();
211             DynamicPanelCameraAndRack = new javax.swing.JPanel();
212             CameraPanel = new javax.swing.JPanel();
```

```
213             jTgbPause = new javax.swing.JToggleButton();
214
215             setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
216
217             jLayeredPane1.setMaximumSize(new java.awt.Dimension(1460, 700));
218             jLayeredPane1.setMinimumSize(new java.awt.Dimension(1460, 700));
219
220             PanelCalibration.setMaximumSize(new java.awt.Dimension(1460, 700));
221             PanelCalibration.setMinimumSize(new java.awt.Dimension(1460, 700));
222             PanelCalibration.setPreferredSize(new java.awt.Dimension(1460, 700));
223
224             btnCalibrate.setBackground(new java.awt.Color(255, 255, 255));
225             btnCalibrate.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
226             btnCalibrate.setText("Calibrate");
227             btnCalibrate.addActionListener(new java.awt.event.ActionListener() {
228                 public void actionPerformed(java.awt.event.ActionEvent evt) {
229                     btnCalibrateActionPerformed(evt);
230                 }
231             });
232
233             pleaseCalibrateToContinueLabel.setFont(new java.awt.Font("Verdana", 2, 12));
                // NOI18N
234             pleaseCalibrateToContinueLabel.setText("Please calibrate to continue");
235
236             lblRoeBot1.setFont(new java.awt.Font("Segoe UI Black", 3, 48)); // NOI18N
237             lblRoeBot1.setText("Welcome to the RoeBot");
238
239             jLabel1.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
240             jLabel1.setText("A project by: ");
241
242             jLabel3.setText("Yngve Bratthaug");
243
244             jLabel4.setText("Per Espen Aarseth");
245
246             jLabel5.setText("Kristian Andre Lilleindset");
247
248             jLabel6.setText("Kristoffer Hildrestrand");
249
250             textFieldSetSearches.setText("Operation interaval (min)");
251             textFieldSetSearches.addActionListener(new java.awt.event.ActionListener() {
252                 public void actionPerformed(java.awt.event.ActionEvent evt) {
253                     textFieldSetSearchesActionPerformed(evt);
254                 }
255             });
256
257             setNumberOfSearchesButton.setText("Start");
258             setNumberOfSearchesButton.addActionListener(new
                java.awt.event.ActionListener() {
259                 public void actionPerformed(java.awt.event.ActionEvent evt) {
260                     setNumberOfSearchesButtonActionPerformed(evt);
261                 }
262             });
263
264             errorMessageSetTraysLabel.setBackground(new java.awt.Color(255, 51, 51));
265             errorMessageSetTraysLabel.setForeground(new java.awt.Color(255, 51, 51));
266             errorMessageSetTraysLabel.setText("Please enter 1, 2 or 3");
267
268             javax.swing.GroupLayout NumberOfSearchesLayout = new
                javax.swing.GroupLayout(NumberOfSearches);
269             NumberOfSearches.setLayout(NumberOfSearchesLayout);
270             NumberOfSearchesLayout.setHorizontalGroup(
271
                    NumberOfSearchesLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
                    nt.LEADING)
272                     .addGroup(NumberOfSearchesLayout.createSequentialGroup()
273                         .addGap(9, 9, 9)
274                         .addComponent(textFieldSetSearches,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 138,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
275                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
276                         .addComponent(setNumberOfSearchesButton,
                            javax.swing.GroupLayout.DEFAULT_SIZE, 141, Short.MAX_VALUE)
277                         .addContainerGap())
```

```
278                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    NumberOfSearchesLayout.createSequentialGroup()
279                     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
280                     .addComponent(errorMessageSetTraysLabel)
281                     .addGap(105, 105, 105))
282             );
283             NumberOfSearchesLayout.setVerticalGroup(
284 
                    NumberOfSearchesLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
                    nt.LEADING)
285                 .addGroup(NumberOfSearchesLayout.createSequentialGroup()
286                     .addGap(29, 29, 29)
287 
                        .addGroup(NumberOfSearchesLayout.createParallelGroup(javax.swing.Group
                        Layout.Alignment.BASELINE)
288                         .addComponent(textFieldSetSearches,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 48,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
289                         .addComponent(setNumberOfSearchesButton,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 48,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
290                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        8, Short.MAX_VALUE)
291                     .addComponent(errorMessageSetTraysLabel,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 22,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
292             );
293 
294             javax.swing.GroupLayout PanelCalibrationLayout = new
                javax.swing.GroupLayout(PanelCalibration);
295             PanelCalibration.setLayout(PanelCalibrationLayout);
296             PanelCalibrationLayout.setHorizontalGroup(
297 
                    PanelCalibrationLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
                    nt.LEADING)
298                 .addGroup(PanelCalibrationLayout.createSequentialGroup()
299                     .addContainerGap(489, Short.MAX_VALUE)
300 
                        .addGroup(PanelCalibrationLayout.createParallelGroup(javax.swing.Group
                        Layout.Alignment.LEADING)
301                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                            PanelCalibrationLayout.createSequentialGroup()
302                             .addComponent(lblRoeBot1,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 642,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
303                             .addGap(329, 329, 329))
304                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                            PanelCalibrationLayout.createSequentialGroup()
305 
                                .addGroup(PanelCalibrationLayout.createParallelGroup(javax.swi
                                ng.GroupLayout.Alignment.LEADING)
306                                 .addComponent(btnCalibrate,
                                    javax.swing.GroupLayout.PREFERRED_SIZE, 244,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
307                                 .addGroup(PanelCalibrationLayout.createSequentialGroup()
308 
                                    .addGroup(PanelCalibrationLayout.createParallelGroup(j
                                    avax.swing.GroupLayout.Alignment.TRAILING)
309                                     .addComponent(NumberOfSearches,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
310                                     .addComponent(pleaseCalibrateToContinueLabel,
                                        javax.swing.GroupLayout.PREFERRED_SIZE, 280,
                                        javax.swing.GroupLayout.PREFERRED_SIZE))
311                                 .addGap(230, 230, 230)
312 
                                    .addGroup(PanelCalibrationLayout.createParallelGroup(j
                                    avax.swing.GroupLayout.Alignment.LEADING)
313                                     .addComponent(jLabel1,
                                        javax.swing.GroupLayout.PREFERRED_SIZE, 96,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                    .addComponent(jLabel3)
                                    .addComponent(jLabel4)
                                    .addComponent(jLabel5)
                                    .addComponent(jLabel6))))
                        .addGap(146, 146, 146))))
        );
        PanelCalibrationLayout.setVerticalGroup(

            PanelCalibrationLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
            nt.LEADING)
            .addGroup(PanelCalibrationLayout.createSequentialGroup()
                .addGap(29, 29, 29)
                .addComponent(lblRoeBot1, javax.swing.GroupLayout.PREFERRED_SIZE,
                117, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(27, 27, 27)
                .addComponent(pleaseCalibrateToContinueLabel,
                javax.swing.GroupLayout.PREFERRED_SIZE, 58,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(50, 50, 50)
                .addComponent(btnCalibrate, javax.swing.GroupLayout.PREFERRED_SIZE,
                114, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                137, Short.MAX_VALUE)

                .addGroup(PanelCalibrationLayout.createParallelGroup(javax.swing.Group
                Layout.Alignment.LEADING)
                    .addGroup(PanelCalibrationLayout.createSequentialGroup()
                        .addComponent(jLabel1,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 40,
                        javax.swing.GroupLayout.PREFERRED_SIZE)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel3)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel4)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel5)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
                        LATED)
                        .addComponent(jLabel6))
                    .addComponent(NumberOfSearches,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(44, 44, 44))
        );

        lblRoeBot1.getAccessibleContext().setAccessibleName("Welcome to RoeBot");

        PanelReady.setMaximumSize(new java.awt.Dimension(1460, 700));
        PanelReady.setMinimumSize(new java.awt.Dimension(1460, 700));
        PanelReady.setPreferredSize(new java.awt.Dimension(1460, 700));

        tgbSearchSystem.setBackground(new java.awt.Color(51, 255, 0));
        tgbSearchSystem.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        tgbSearchSystem.setText("Search");
        tgbSearchSystem.setMaximumSize(new java.awt.Dimension(160, 80));
        tgbSearchSystem.setMinimumSize(new java.awt.Dimension(160, 80));
        tgbSearchSystem.setName(""); // NOI18N
        tgbSearchSystem.setPreferredSize(new java.awt.Dimension(160, 80));
        tgbSearchSystem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                tgbSearchSystemActionPerformed(evt);
            }
        });

        tgbEmergencyStop.setBackground(new java.awt.Color(255, 0, 0));
```

```
365             tgbEmergencyStop.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
366             tgbEmergencyStop.setText("Stop");
367             tgbEmergencyStop.setMaximumSize(new java.awt.Dimension(160, 80));
368             tgbEmergencyStop.setMinimumSize(new java.awt.Dimension(160, 80));
369             tgbEmergencyStop.setPreferredSize(new java.awt.Dimension(160, 80));
370             tgbEmergencyStop.addActionListener(new java.awt.event.ActionListener() {
371                 public void actionPerformed(java.awt.event.ActionEvent evt) {
372                     tgbEmergencyStopActionPerformed(evt);
373                 }
374             });
375
376             btnReCalibrate.setBackground(new java.awt.Color(255, 255, 255));
377             btnReCalibrate.setText("Recalibrate");
378             btnReCalibrate.addActionListener(new java.awt.event.ActionListener() {
379                 public void actionPerformed(java.awt.event.ActionEvent evt) {
380                     btnReCalibrateActionPerformed(evt);
381                 }
382             });
383
384             UpdateTable.setModel(new javax.swing.table.DefaultTableModel(
385                 new Object [][] {
386
387                 },
388                 new String [] {
389                     "Tray number", "Removed roes"
390                 }
391             ) {
392                 Class[] types = new Class [] {
393                     java.lang.Integer.class, java.lang.Integer.class
394                 };
395
396                 public Class getColumnClass(int columnIndex) {
397                     return types [columnIndex];
398                 }
399             });
400             UpdateTable.getTableHeader().setReorderingAllowed(false);
401             jScrollPane1.setViewportView(UpdateTable);
402
403             lblRoeBot.setFont(new java.awt.Font("Segoe UI Black", 3, 48)); // NOI18N
404             lblRoeBot.setText("RoeBot");
405
406             tgbLivePhoto.setBackground(new java.awt.Color(255, 255, 255));
407             tgbLivePhoto.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
408             tgbLivePhoto.setText("Live photo");
409             tgbLivePhoto.addActionListener(new java.awt.event.ActionListener() {
410                 public void actionPerformed(java.awt.event.ActionEvent evt) {
411                     tgbLivePhotoActionPerformed(evt);
412                 }
413             });
414
415             btnLightRegulations.setBackground(new java.awt.Color(255, 255, 255));
416             btnLightRegulations.setText("Light Regulation");
417             btnLightRegulations.addActionListener(new java.awt.event.ActionListener() {
418                 public void actionPerformed(java.awt.event.ActionEvent evt) {
419                     btnLightRegulationsActionPerformed(evt);
420                 }
421             });
422
423             javax.swing.GroupLayout CameraPanelLayout = new
                javax.swing.GroupLayout(CameraPanel);
424             CameraPanel.setLayout(CameraPanelLayout);
425             CameraPanelLayout.setHorizontalGroup(
426
                    CameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
                    ADING)
427                 .addGap(0, 573, Short.MAX_VALUE)
428             );
429             CameraPanelLayout.setVerticalGroup(
430
                    CameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
                    ADING)
431                 .addGap(0, 429, Short.MAX_VALUE)
432             );
```

```
433
434         javax.swing.GroupLayout DynamicPanelCameraAndRackLayout = new
            javax.swing.GroupLayout(DynamicPanelCameraAndRack);
435         DynamicPanelCameraAndRack.setLayout(DynamicPanelCameraAndRackLayout);
436         DynamicPanelCameraAndRackLayout.setHorizontalGroup(
437
                DynamicPanelCameraAndRackLayout.createParallelGroup(javax.swing.GroupLayou
                t.Alignment.LEADING)
438             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                DynamicPanelCameraAndRackLayout.createSequentialGroup()
439                 .addContainerGap(38, Short.MAX_VALUE)
440                 .addComponent(CameraPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
441                 .addContainerGap())
442         );
443         DynamicPanelCameraAndRackLayout.setVerticalGroup(
444
                DynamicPanelCameraAndRackLayout.createParallelGroup(javax.swing.GroupLayou
                t.Alignment.LEADING)
445             .addGroup(DynamicPanelCameraAndRackLayout.createSequentialGroup()
446                 .addContainerGap()
447                 .addComponent(CameraPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
448                 .addContainerGap())
449         );
450
451         jTgbPause.setBackground(new java.awt.Color(255, 255, 0));
452         jTgbPause.setFont(new java.awt.Font("TakaoPGothic", 1, 14)); // NOI18N
453         jTgbPause.setText("Pause");
454         jTgbPause.setToolTipText("");
455         jTgbPause.addActionListener(new java.awt.event.ActionListener() {
456             public void actionPerformed(java.awt.event.ActionEvent evt) {
457                 jTgbPauseActionPerformed(evt);
458             }
459         });
460
461         javax.swing.GroupLayout PanelReadyLayout = new
            javax.swing.GroupLayout(PanelReady);
462         PanelReady.setLayout(PanelReadyLayout);
463         PanelReadyLayout.setHorizontalGroup(
464
                PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
                DING)
465             .addGroup(PanelReadyLayout.createSequentialGroup()
466                 .addGap(74, 74, 74)
467
                    .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                    .Alignment.TRAILING)
468                     .addComponent(DynamicPanelCameraAndRack,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
469                     .addGroup(PanelReadyLayout.createSequentialGroup()
470                         .addComponent(tgbSearchSystem,
                            javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
471                         .addGap(39, 39, 39)
472                         .addComponent(jTgbPause,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 165,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
473                         .addGap(27, 27, 27)
474                         .addComponent(tgbEmergencyStop,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 160,
                            javax.swing.GroupLayout.PREFERRED_SIZE)))
475                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                    245, Short.MAX_VALUE)
476
                    .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                    .Alignment.LEADING)
477                     .addGroup(PanelReadyLayout.createSequentialGroup()
478                         .addComponent(btnLightRegulations,
```

```
                        javax.swing.GroupLayout.PREFERRED_SIZE, 190,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(90, 90, 90)
                        .addComponent(btnReCalibrate,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 172,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jScrollPane1,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(65, 65, 65))
            .addGroup(PanelReadyLayout.createSequentialGroup()
                .addGap(293, 293, 293)
                .addComponent(tgbLivePhoto, javax.swing.GroupLayout.PREFERRED_SIZE,
                151, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(210, 210, 210)
                .addComponent(lblRoeBot, javax.swing.GroupLayout.PREFERRED_SIZE,
                224, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                Short.MAX_VALUE))
        );
        PanelReadyLayout.setVerticalGroup(

            PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
            DING)
            .addGroup(PanelReadyLayout.createSequentialGroup()
                .addContainerGap()

                .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                .Alignment.BASELINE)
                    .addComponent(lblRoeBot, javax.swing.GroupLayout.PREFERRED_SIZE,
                    117, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(tgbLivePhoto,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 75,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.GroupLayout
                .Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    PanelReadyLayout.createSequentialGroup()
                        .addComponent(DynamicPanelCameraAndRack,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(115, 115, 115))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    PanelReadyLayout.createSequentialGroup()
                        .addComponent(jScrollPane1,
                        javax.swing.GroupLayout.PREFERRED_SIZE, 350,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(90, 90, 90)

                        .addGroup(PanelReadyLayout.createParallelGroup(javax.swing.Gro
                        upLayout.Alignment.LEADING)

                            .addGroup(PanelReadyLayout.createParallelGroup(javax.swing
                            .GroupLayout.Alignment.BASELINE)
                                .addComponent(tgbEmergencyStop,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 81,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(tgbSearchSystem,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jTgbPause,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 80,
                                javax.swing.GroupLayout.PREFERRED_SIZE))

                            .addGroup(PanelReadyLayout.createParallelGroup(javax.swing
                            .GroupLayout.Alignment.BASELINE)
                                .addComponent(btnLightRegulations,
```

```
                                javax.swing.GroupLayout.PREFERRED_SIZE, 74,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(btnReCalibrate,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 70,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(9, 9, 9))))
        );

        jLayeredPane1.setLayer(PanelCalibration,
        javax.swing.JLayeredPane.DEFAULT_LAYER);
        jLayeredPane1.setLayer(PanelReady, javax.swing.JLayeredPane.DEFAULT_LAYER);

        javax.swing.GroupLayout jLayeredPane1Layout = new
        javax.swing.GroupLayout(jLayeredPane1);
        jLayeredPane1.setLayout(jLayeredPane1Layout);
        jLayeredPane1Layout.setHorizontalGroup(

            jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
            .addGroup(jLayeredPane1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(PanelCalibration,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE))

            .addGroup(jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
                .addGroup(jLayeredPane1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(PanelReady,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)))
        );
        jLayeredPane1Layout.setVerticalGroup(

            jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
            .addComponent(PanelCalibration,
            javax.swing.GroupLayout.Alignment.TRAILING,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGroup(jLayeredPane1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                jLayeredPane1Layout.createSequentialGroup()
                    .addComponent(PanelReady,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())))
        );

        javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        pack();
```

```
552         }// </editor-fold>//GEN-END:initComponents
553
554         /**
555          * TODO: call Loading and loading image is going to circulate to finished
556          * calibrating
557          *
558          * @param evt
559          */
560         private void btnCalibrateActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_btnCalibrateActionPerformed
561             btnCalibrate.setBackground(Color.white);
562             loading.setVisible(true);
563             NumberOfSearches.setVisible(true);
564             btnCalibrate.setVisible(false);
565             pleaseCalibrateToContinueLabel.setVisible(false);
566             roeBotFasade.doCalibrate();
567
568             //TODO: if calibration done, then set Loading not visible. and opens
569             //setSearches-panel, and then you can put in number of searches.
570             //the "ready"-panel with number of found trays information.
571             //PanelReady.setVisible(true);
572             //PanelCalibration.setVisible(false);
573         }//GEN-LAST:event_btnCalibrateActionPerformed
574
575         /**
576          * TOGGLEBUTTON, will start the system and cycle it.
577          *
578          * @param evt
579          */
580         private void tgbSearchSystemActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_tgbSearchSystemActionPerformed
581             tgbSearchSystem.setBackground(Color.GREEN);
582             if (tgbSearchSystem.isSelected()) {
583                 roeBotFasade.startCycle();
584                 tgbSearchSystem.setText("DestROEing");
585                 tgbEmergencyStop.setText("Stop");
586             }
587
588         }//GEN-LAST:event_tgbSearchSystemActionPerformed
589
590
591         private void tgbEmergencyStopActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_tgbEmergencyStopActionPerformed
592             System.out.println("Stop pressed!");
593             tgbEmergencyStop.setVisible(true);
594             tgbEmergencyStop.setBackground(Color.red);
595
596             if (tgbSearchSystem.isSelected()) {
597                 tgbEmergencyStop.setVisible(true);
598                 roeBotFasade.stopRobot();
599                 tgbEmergencyStop.setText("STOPPED!");
600                 tgbSearchSystem.setText("Search");
601             }
602         }//GEN-LAST:event_tgbEmergencyStopActionPerformed
603
604         private void btnReCalibrateActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_btnReCalibrateActionPerformed
605             // TODO add your handling code here:
606             roeBotFasade.doCalibrate();
607         }//GEN-LAST:event_btnReCalibrateActionPerformed
608
609         /**
610          * TOGGLE BUTTON, activating the photo if pushed. Else rack system will
611          * appear.
612          *
613          * @param evt toggleButton
614          */
615         private void tgbLivePhotoActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_tgbLivePhotoActionPerformed
616             //TODO: Change tgbLivePhoto.isSelected() to when it is two or three racks
                that show
617             //right photo.
618
```

```
619             if (tgbLivePhoto.isSelected()) {
620                 rackClosed.getClosedRack().setVisible(false);
621                 CameraPanel.setVisible(true);
622                 tgbLivePhoto.setText("Rack Update");
623                 webSource = new VideoCapture(0);
624                 myThread = new DaemonThread();
625                 Thread t = new Thread(myThread);
626                 t.setDaemon(true);
627                 myThread.runnable = true;
628                 t.start();
629             } else {
630                 tgbLivePhoto.setText("Live Photo");
631                 //panelCamera.setVisible(false);
632                 CameraPanel.setVisible(false);
633                 rackClosed.getClosedRack().setVisible(true);
634                 myThread.runnable = false;
635                 webSource.release();
636                 Thread t = new Thread(myThread);
637                 t.setDaemon(false);
638                 myThread.runnable = false;
639                 t.interrupt();
640             }
641         }//GEN-LAST:event_tgbLivePhotoActionPerformed
642 
643 
644         private void btnLightRegulationsActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_btnLightRegulationsActionPerformed
645             LightRegulations lightRegulator = new LightRegulations();
646             // display window
647             lightRegulator.setVisible(true);
648 
649             // get values from the window
650             this.redLightVal = lightRegulator.getRedLightValue();
651             this.greenLightVal = lightRegulator.getGreenLightValue();
652             this.blueLightVal = lightRegulator.getBlueLightValue();
653 
654 
655             // Update the fasade with new values
656             this.roeBotFasade.regulateLights(this.redLightVal, this.greenLightVal,
                this.blueLightVal);
657             System.out.println("endra farger til:"+ redLightVal +" , "+ greenLightVal +
                " , "+blueLightVal);
658 
659         }//GEN-LAST:event_btnLightRegulationsActionPerformed
660 
661         private void setNumberOfSearchesButtonActionPerformed(java.awt.event.ActionEvent
            evt) {//GEN-FIRST:event_setNumberOfSearchesButtonActionPerformed
662 
663 
664 
665 
666                 this.roeBotFasade.setSearchInterval(this.operationInterval);
667                 this.loading.setVisible(false);
668                 this.PanelCalibration.setVisible(false);
669                 this.PanelReady.setVisible(true);
670                 this.setNumberOfSearchesButton.setEnabled(true);
671                 this.UpdateTable.setVisible(true);
672 
673 
674         }//GEN-LAST:event_setNumberOfSearchesButtonActionPerformed
675 
676         private void textFieldSetSearchesActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_textFieldSetSearchesActionPerformed
677 
678             boolean numberInput = false;
679             //Wait for numberinput
680             while(!numberInput)
681                 {
682             String input = textFieldSetSearches.getText();
683             //if 2 or 3 is typed
684             //then choose right image in dynamic panel
685             if (input != null)
686             {
```

```
687                 try{
688                 this.operationInterval = Integer.parseInt(input);
689                 numberInput = true;
690                 }
691                 catch(NumberFormatException ex)
692                 {
693                     System.out.println("må vårrå ett tall");
694                     numberInput = false;
695                 }
696                 this.setNumberOfSearchesButton.setEnabled(true);
697             }
698             else
699             {
700                 this.setNumberOfSearchesButton.setEnabled(false);
701             }
702             }
703         }//GEN-LAST:event_textFieldSetSearchesActionPerformed
704 
705         private void jTgbPauseActionPerformed(java.awt.event.ActionEvent evt)
            {//GEN-FIRST:event_jTgbPauseActionPerformed
706             //
707             if (jTgbPause.isSelected()) {
708                 System.out.println("Pause is pressed");
709                 // check the state of the pause button, pause or continue
710                 if (jTgbPause.getText().equals("Pause")) {
711                     System.out.println("Pause is Button");
712                     // pause robot, change button to continue button
713                     jTgbPause.setText("Continue");
714                     jTgbPause.setBackground(Color.green);
715                     roeBotFasade.pauseRobot();
716                 }
717                 //
718                 else if (jTgbPause.getText().equals("Continue")) {
719                     System.out.println("Continue is Button");
720                     // continue robot, change button state to pause
721                     this.roeBotFasade.continueRobot();
722                     jTgbPause.setText("Pause");
723                     jTgbPause.setBackground(Color.yellow);
724                 }
725             }
726 
727         }//GEN-LAST:event_jTgbPauseActionPerformed
728 
729 
730         private void updateTrayTable() {
731             TableColumn coloumn;
732             Component comp;
733 
734         }
735 
736 
737 
738         // Variables declaration - do not modify//GEN-BEGIN:variables
739         private javax.swing.JPanel CameraPanel;
740         private javax.swing.JPanel DynamicPanelCameraAndRack;
741         private javax.swing.JPanel NumberOfSearches;
742         public javax.swing.JPanel PanelCalibration;
743         public javax.swing.JPanel PanelReady;
744         public javax.swing.JTable UpdateTable;
745         private javax.swing.JButton btnCalibrate;
746         public javax.swing.JButton btnLightRegulations;
747         private javax.swing.JButton btnReCalibrate;
748         private javax.swing.JLabel errorMessageSetTraysLabel;
749         private javax.swing.JLabel jLabel1;
750         private javax.swing.JLabel jLabel3;
751         private javax.swing.JLabel jLabel4;
752         private javax.swing.JLabel jLabel5;
753         private javax.swing.JLabel jLabel6;
754         private javax.swing.JLayeredPane jLayeredPane1;
755         private javax.swing.JScrollPane jScrollPane1;
756         public javax.swing.JToggleButton jTgbPause;
757         private javax.swing.JLabel lblRoeBot;
758         private javax.swing.JLabel lblRoeBot1;
```

```
759         private javax.swing.JLabel pleaseCalibrateToContinueLabel;
760         private javax.swing.JButton setNumberOfSearchesButton;
761         private javax.swing.JTextField textFieldSetSearches;
762         public javax.swing.JToggleButton tgbEmergencyStop;
763         private javax.swing.JToggleButton tgbLivePhoto;
764         public javax.swing.JToggleButton tgbSearchSystem;
765         // End of variables declaration//GEN-END:variables
766     }
767
```

Appendix I

Arduino Code

I.1 Horizontal System Code

```
#include <Stepper.h>
#include <Wire.h>
#include <FastLED.h>

//Debug flag
bool debug = false;
bool calibTest = false;
bool cal = false;

/******** Stepper ***********/
/******** IO ***********/
#define MOTOR_ENABLE 8
/******** X ***********/
#define X_MOTOR_STEP 4
#define X_MOTOR_DIR 7
#define X_HOME_SWITCH A2
#define X_OUTER_SWITCH A3

/******** Y ***********/
#define Y_MOTOR_STEP 3
#define Y_MOTOR_DIR 6
#define Y_HOME_SWITCH A1
#define Y_OUTER_SWITCH A0

/******** MAGNET ***********/
#define MAGNET_PIN 9
#define MAGNET_ENDSTOP A6

/******** DIRECTIONS ***********/
#define DEC_DIR -1
#define INC_DIR 1

/******** STEPPER SPESIFICATIONS ***********/
#define DRIVER_MICROSTEPS_FACTOR  4 // Factor
#define STEP_PER_REVOLUTION 200
#define BELT_PRESENT_PITCH 2//mm
#define X_PULLEY_TOOTH 20
#define Y_PULLEY_TOOTH 16

//Calculate step per mm
#define X_STEP_PER_MM ((STEP_PER_REVOLUTION*DRIVER_MICROSTEPS_FACTOR) / (BELT_PRESENT_PITCH * X_PULLEY_TOOTH))
#define Y_STEP_PER_MM ((STEP_PER_REVOLUTION*DRIVER_MICROSTEPS_FACTOR) / (BELT_PRESENT_PITCH * Y_PULLEY_TOOTH))


/******** STOP INTERRUPT PIN ***********/
#define INTERRUPT_PIN 2

/******** I2C Communication ***********/
#define ANSWERSIZE 5
#define ADDRESS 0x03

/************** Serial Communication *************/
#define DEVICE_NAME "dev1"

/******** LED ***********/
#define LED_PIN     11
#define NUM_LEDS    42 // 42 on the robot
#define LED_TYPE    WS2811
#define COLOR_ORDER GRB

CRGB leds[NUM_LEDS];
CRGBPalette16 currentPalette;

short ledRGB[] = {0, 0, 0};

//Default stepper speed
int stepSpeed[] = {120 , 120};

//Travelled distance
int travelledDistance;

```

```
72  //Structure to hold position
73  typedef struct Pos {
74    int x;
75    int y;
76  } Pos;
77  //The different positions to use throughout
78  Pos currentPos;
79  Pos maxPos;
80  Pos newPos;
81
82  //Limit switch booleans
83  boolean xHomeSwitch = false;
84  boolean xAwaySwitch = false;
85  boolean yHomeSwitch = false;
86  boolean yAwaySwitch = false;
87
88  //Interrupt debounce
89  static unsigned long last_interrupt_time = 0;
90  unsigned long interrupt_time = 0;
91
92
93  //Set steppers
94  Stepper motorX(STEP_PER_REVOLUTION * DRIVER_MICROSTEPS_FACTOR, X_MOTOR_STEP,
    X_MOTOR_DIR);
95  Stepper motorY(STEP_PER_REVOLUTION * DRIVER_MICROSTEPS_FACTOR, Y_MOTOR_STEP,
    Y_MOTOR_DIR);
96
97  //--------------------------------ENUMS-------------------------------
98  //ENUMS to keep track of what has been sent to the arduino, and what it should do.
99  //The state of the arduino
100 enum state {
101   busy = 0x50,
102   readyToRecieve = 0x51,
103   stopped = 0x52,
104   emc = 0x60,
105   fail = 0x67,
106   linearBotLimitTrigged = 0x64,
107   parameter = 0x70
108 };
109
110 /**
111  * *COMMANDS  * the Master can send to the slave.
112
113     IF AN COMMAND IS ADDED I NEEDS TO BE ADDED TO CommandCheck.
114 */
115 enum command {
116   //Main loop states
117   idle = 0x01,
118   failure = 0x03,
119
120   // Genral Cmd
121   moveRobotTo = 0x05,
122   moveRobot = 0x06,
123   stopRobot = 0x07,
124   doCalibrate = 0x10,
125   turnOnLight = 0x11,
126   changeColor = 0x12,
127   discoLight = 0x13,
128   findTray  = 0x14,
129   changeVelocity = 0x20,
130   changeAcceleration = 0x21,
131   magnetOn = 0x22,
132   magnetOff = 0x23,
133
134   // Request command
135   retrieveState = 0x30,
136   calibParam = 0x31
137 };
138
139 //ENUM For acknowledgement  og negative-acknowledgement
140 const byte ACK = 1;
141 const byte NACK = 0;
142 //enum acknowlagement {ACK = 1, NACK = 0};
```

```
143  //ENUM for color on leds
144  enum RGBEnum {R = 0, G = 1, B = 2};
145  //ENUM for axiz
146  enum axisEnum {X_AXIS = 0, Y_AXIS = 1};
147
148  //Create enums
149  enum state inState;             //To keep track of which STATE the arduino currently
     is in. Busy when doing stuff, ready/idle when not. EMC, encoder failure, ETC...
150  enum command mainCommand = idle;       //Switch case command for the main loop
151  enum command recieveCommand;    //Switch case command for the RECIEVE command,
     reading storing values to be done later.
152  //enum command checkCommand;
153
154
155
156
157
158
159  void setup() {
160    //Start the serial port
161    Serial.begin(115200);
162
163    if (debug)
164    {
165      //Set default positions
166      currentPos.x = 0;
167      currentPos.y = 0;
168      maxPos.x = 5000;
169      maxPos.y = 3000;
170      newPos.y = 0;
171      newPos.x = 0;
172    }
173
174    // IO setup.
175    pinMode(MOTOR_ENABLE, OUTPUT);
176    /******** X ***********/
177    pinMode(X_MOTOR_STEP, OUTPUT);
178    pinMode(X_MOTOR_DIR, OUTPUT);
179    pinMode(X_HOME_SWITCH, INPUT_PULLUP);
180    pinMode(X_OUTER_SWITCH, INPUT_PULLUP);
181    /******** Y ***********/
182    pinMode(Y_MOTOR_STEP, OUTPUT);
183    pinMode(Y_MOTOR_DIR, OUTPUT);
184    pinMode(Y_HOME_SWITCH, INPUT_PULLUP);
185    pinMode(Y_OUTER_SWITCH, INPUT_PULLUP);
186    /******** MAGNET ***********/
187    pinMode(MAGNET_PIN, OUTPUT);
188    pinMode(MAGNET_ENDSTOP, INPUT);
189    pinMode(13, OUTPUT);
190    /******** ENABLE SIGNAL ***********/
191    pinMode(MOTOR_ENABLE, OUTPUT);
192
193
194    /******** INTERRUPT PIN ***********/
195    //Initialize the safety interrupts
196    //When the limit switches are reached(pressed)
197    pinMode(INTERRUPT_PIN, INPUT_PULLUP);
198    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), stopMotors, LOW);
199
200    /******** LED ***********/
201    FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
     TypicalLEDStrip );
202
203
204    if (debug) {
205      Serial.println(F("Setup complete"));
206    }
207
208    //Set main command to idle
209    mainCommand = idle;
210  }
211  //--------------------------------------------------------------------------------
     -------------------
```

```
212
213
214   void loop()
215   {
216
217     receiveSerialEvent();
218     //Switch command for the main loop
219
      //-------------------------------------------------------------------------------
      ----------------
220     switch (mainCommand)
221     {
222       // Idle state.
223       case idle:
224         if (debug) {
225           Serial.println(F("Main: Idle"));
226         }
227         delay(5); //delay(10)
228          updateStatusAndNotify(readyToRecieve);
229         //Enable motors
230         disableMotors();
231         //Set state to idle
232         break;
233
234
235       //-------------------------------------------------------------
236       case moveRobotTo:
237
238         //Set state to busy
239          updateStatusAndNotify(busy);
240         //Do a check to see if robot needs to move
241         //Make movestep check endstop in between every step, and then return true if
          moving was succesfull without hitting endstops
242
243         //Check if robot needs to move of if the stop interrupt has been trigged.
244         if (!robotInPosition()) // Returns true if the robot is in position
245         {
246
247           //Move the robot in desired steps. IF this is false, then it was not
            possible to move
248           if (!moveStep(newPos.x, newPos.y))
249           {
250             mainCommand = failure;
251           }
252         } else {
253           mainCommand = idle;
254         }
255         break;
256
257
258
259
260    //-----------------------------------------------------
261
262       //Calibrate the robot
263       case doCalibrate:
264         if (debug) {
265           Serial.println(F("Main: Do Calibration "));
266         }
267
268         //Update status to busy
269         updateStatusAndNotify(busy);
270         // Do the calibration
271         calibration();
272         //If testing
273         if (calibTest) {
274           Serial.println("Reached outer limit");
275           Serial.print("Max pos X");
276           Serial.println(maxPos.x);
277           Serial.print("Max pos Y");
278           Serial.println(maxPos.y);
279         }
280
```

```
281         // Move to senter of area
282         newPos.x = (maxPos.x / 2);
283         newPos.y = (maxPos.y / 2);
284         mainCommand = moveRobotTo;
285         break;
286 
287 
288 
289 
290   //-----------------------------------------------------
291 
292       //Turns magnet ON
293       case magnetOn:
294         if (debug) {
295           Serial.println(F("Main: magnetOn"));
296         }
297        //Turn magnet on
298         magnetON();
299         mainCommand = idle;
300         break;
301 
302 
303 
304 
305   //-----------------------------------------------------
306 
307       //Turns magnet OFF
308       case magnetOff:
309         if (debug) {
310           Serial.println(F("Main: magnetOff"));
311         }
312        // updateStatusAndNotify(busy);
313         magnetOFF();
314         mainCommand = idle;
315         break;
316 
317 
318 
319 
320   //-----------------------------------------------------
321 
322       //Turns magnet OFF
323       case findTray:
324         if (debug) {
325           Serial.println(F("Main: find tray "));
326         }
327         updateStatusAndNotify(busy);
328 
329         if (!moveToTray()) {
330           //TODO: removed for testing
331          // mainCommand = failure;
332          mainCommand = idle;
333         } else {
334           mainCommand = idle;
335         }
336         break;
337 
338 
339 
340 
341   //-----------------------------------------------------
342       // stop
343       case stopRobot:
344         if (debug) {
345           Serial.println(F("Main: stopRobot"));
346         }
347 
348         //Set state to in EMC
349         disableMotors();
350         updateStatusAndNotify(stopped);
351         break;
352 
353 
```

```
354
355
356   //-------------------------------------------------
357       //Failure
358       case failure:
359         if (debug) {
360           Serial.println(F("Main: Failure"));
361         }
362         //Enable motors
363         disableMotors();
364         // Sends state update
365         updateStatusAndNotify(checkFailure());
366
367         //Set to idle after failure update
368         mainCommand = idle;
369         break;
370
371
372
373
374       //-------------------------------------------------------------------------
375       //Set the command for calibrating the robot.
376       case changeColor:
377         if (debug) {
378           Serial.println(F("Main: Change colors on led"));
379         }
380         //Set the led values retrieved from serial input
381         setLedColor(ledRGB[R], ledRGB[G], ledRGB[B]);
382         mainCommand = idle;
383         break;
384
385
386
387
388       //-------------------------------------------------------------------------
389       //Set the command for calibrating the robot.
390       case discoLight:
391         if (debug) {
392           Serial.println(F("Main: Disco mode"));
393         }
394         updateStatusAndNotify(busy); // When in an opperation state is busy.
395         ledDiscoShow();
396         break;
397
398
399
400
401      //-------------------------------------------------------------------------
402       default:
403         if (debug) {
404           Serial.println(F("In default"));
405         }
406         break;
407
          //-------------------------------------------------------------------------
          --------------------
408     }
409   }
410
411
412
413
```

```
1
2   /**
3      method for callibrating the distances in the x and y axes, takes a pointer
4      to array as input for returning the number of steps to the call.
5   */
6   //TODO: Remember to check DIR_LEFT & RIGHT
7   void calibration()
8   {
9
10    //Enable motors
11    enableMotors();
12    // declaring states for calibration
13    const int FIND_ZERO = 0;
14    const int FIND_AREA = 1;
15
16    // setting state to go to home position
17    int state = FIND_ZERO;
18
19    // declaring calibration variable
20    bool calibrated = false;
21
22    // run calibration until calibration is complete
23    while (!calibrated)
24    {
25      switch (state)
26      {
27        // motors run until they reach their homepositions (0,0)
28        case FIND_ZERO:
29
30          // run x-axis motor until home position is reached
31          if (ATDConverter(X_HOME_SWITCH))
32          {
33
34            moveXMotor(DEC_DIR);
35          }
36
37          // run y-axis motor until home position is reached
38          if (ATDConverter(Y_HOME_SWITCH))
39          {
40            moveYMotor(DEC_DIR);
41          }
42          // if x and y motor is at their home positions update axis positions to 0,
43          // and switch to away state
44          if (!ATDConverter(X_HOME_SWITCH) && !ATDConverter(Y_HOME_SWITCH))
45          {
46            currentPos.x = 0;
47            currentPos.y = 0;
48
49            //Set state value
50            state = FIND_AREA;
51            if (calibTest) {
52              Serial.println("Reached home");
53            }
54          }
55          break;
56
57
58        // motors run until they reach their away positions(x,y)
59        case FIND_AREA:
60          // run x-axis motor and count steps until away position is reached
61          if (ATDConverter(X_OUTER_SWITCH))
62          {
63            moveXMotor(INC_DIR);
64          }
65
66          // run y-axis motor and count stepil away position is reached
67          if (ATDConverter(Y_OUTER_SWITCH))
68          {
69            moveYMotor(INC_DIR);
70          }
71
72
73          // if x and y motor is at their away positions set calibration variable to
```

```
            true
            if (!ATDConverter(X_OUTER_SWITCH) && !ATDConverter(Y_OUTER_SWITCH))
            {
              //Calibration done
              calibrated = true;
              maxPos.x = currentPos.x;
              maxPos.y = currentPos.y;

              //Enable motors
              disableMotors();
              break;
            }
        }
      }
    }

```

```
/**
   FUNCTIONS TO CONTROL THE MOTORS
*/

void enableMotors()
{
  digitalWrite(MOTOR_ENABLE, LOW);
}

void disableMotors()
{
  digitalWrite(MOTOR_ENABLE, HIGH);
}
```

```
/**
   Slope of a line can be expressed as deltaX/deltaY. A number of steps on Y equals
   a number of steps on X.
   This will help us step along Y and adds deltaX to a counter, when the counter >=Y
   steps a number on X.'
   The motors will toggle simultaniously to the newX and newY.

   RETURNS: This function returns true if it was able to move in the required
   directions
*/

boolean moveStep(int newX, int newY) {
  //Enable motors
  enableMotors();
  int s = 0;
  float calcSteep = 0;
  int deltaX = abs(newX - currentPos.x); //distance to move
  int deltaY = abs(newY - currentPos.y);

  //Return bool to indicate if the steppers still need to move to reach position
  boolean moving = true;

  /**** CALCULATION ****/
  //Check if X should move at all
  if (deltaX != 0 && deltaY != 0)
  {
    if (deltaX == deltaY)
    {
      //Check if moving in wanted direction is possible
      if (moveXPossible(dirX()))
      {
        moveXMotor(dirX()); //onestep and direction
      } else {
        moving = false;
      }
      if (moveYPossible(dirY()))
      {
        moveYMotor(dirY()); //onestep and direction
      } else {
        moving = false;
      }
    }

    //Check if x should move more then Y
    else if (deltaX > deltaY)
    {
     //Calculate the slope
      calcSteep = deltaX / deltaY;
      //Round up the step
      s = round(calcSteep);
      /****Do the moving******/
      //MOVE X
      for (int i = 0; ((i < abs(dirX()*s)) && moving); i++)
      {
        //Check if moving in wanted direction is possible
        if (moveXPossible(dirX()))
        {
          moveXMotor(dirX());
        }
        else
        {
          moving = false;
        }
      }
      // MOVE Y
      //Check if moving in wanted direction is possible
      if (moveYPossible(dirY()))
      {
        moveYMotor(dirY()); //moves one step and increases currentpos
      }
      else
      {
        moving = false;
```

```
71          }
72        }
73
74        //Check if Y should move more then X
75        else if (deltaY > deltaX)
76        {
77          /**** CALCULATION ****/
78          calcSteep = deltaY / deltaX;
79          s = round(calcSteep);
80
81          /****Do the moving******/
82          //MOVE Y
83          for (int i = 0; ((i < abs(dirY()*s)) && moving); i++)
84          {
85            //Check if moving in wanted direction is possible
86            if (moveYPossible(dirY()))
87            {
88              moveYMotor(dirY());
89            }
90            else
91            {
92              moving = false;
93            }
94          }
95
96          // MOVE X
97          //Check if moving in wanted direction is possible
98          if (moveXPossible(dirX()))
99          {
100           moveXMotor(dirX());
101         }
102         else
103         {
104           moving = false;
105         }
106       }
107     }
108
109
110     //IF one of the steppers have reached position
111     //Check if one of the steppers still need to move
112     //Check stepper X
113     if (deltaX != 0 && deltaY == 0)
114     {
115       //MOVE X
116       //Check if moving in wanted direction is possible
117       if (moveXPossible(dirX()))
118       {
119         moveXMotor(dirX());
120       }
121       else
122       {
123         moving = false;
124       }
125     }
126
127
128     //Check if stepper Y needs to move
129     if (deltaY != 0 && deltaX == 0)
130     {
131       //MOVE Y
132       //Check if moving in wanted direction is possible
133       if (moveYPossible(dirY()))
134       {
135         moveYMotor(dirY());
136       }
137       else
138       {
139         moving = false;
140       }
141     }
142     //Return the bool which indicated if position is reached or not
143     return moving;
```

```
144  }
145
146  /*****************************************************/
147
148  //Move the left motor
149  /*
150     Moves the X motor in the wanted direction, with step speed. And increases the
     current positoin
151  */
152  void  moveXMotor(int dir)
153  {
154    motorX.step(dir);
155    motorX.setSpeed(stepSpeed[X_AXIS]);
156    increaseXStep(dir);
157  }
158
159  //Move the right motor
160  void  moveYMotor(int dir)
161  {
162    motorY.step(dir);
163    motorY.setSpeed(stepSpeed[Y_AXIS]);
164    increaseYStep(dir);
165  }
166
167
168  /**
169     Increase the step in left direction with one
170  */
171  void increaseXStep(int dir)
172  {
173    currentPos.x = (currentPos.x + dir);
174  }
175  void increaseYStep(int dir)
176  {
177    currentPos.y = (currentPos.y + dir);
178  }
179
180
181
182
183  /**
184    Return if true if possible to move in the wanted direction
185  */
186  boolean moveXPossible(int dir)
187  {
188    boolean possible = true;
189
190    if (inState == stopped) {
191      boolean possible = false;
192    } else {
193      boolean possible = true;
194
195      //Check if limit switches are activated
196      if (limitTrigged())
197      {
198        //Check the home end stop
199        if (xHomeSwitch)
200        {
201          //Check all the direction, set appropriate bool value.
202          //True moving in opposite direction is wanted
203          if (dir == DEC_DIR)
204          {
205            possible = false;
206          }
207          else if (dir == INC_DIR)
208          {
209            possible = true;
210          }
211        }
212        //Check the away end stop
213        //And determine if moving in the desired direction is possible
214        else if (xAwaySwitch)
215        {
```

```
216           if (dir == DEC_DIR)
217           {
218             possible = true;
219           }
220           else if (dir == INC_DIR)
221           {
222             possible = false;
223           }
224         }
225       }
226     }
227
228     return possible;
229   }
230
231
232   /**
233     Return if true its possible to move in the wanted direction
234   */
235   boolean moveYPossible(int dir)
236   {
237     boolean possible = true;
238
239     if (inState == stopped)
240     {
241       boolean possible = false;
242     }
243     else
244     {
245       //Check if limit switches are activated
246       if (limitTrigged())
247       {
248         //Check the home end stop
249         if (yHomeSwitch)
250         {
251           //Check all the direction, set appropriate bool value.
252           //True moving in opposite direction is wanted
253           if (dir == DEC_DIR)
254           {
255             possible = false;
256           }
257           else if (dir == INC_DIR)
258           {
259             possible = true;
260           }
261         }
262         //Check the away end stop
263         //And determine if moving in the desired direction is possible
264         else if (yAwaySwitch)
265         {
266           if (dir == DEC_DIR)
267           {
268             possible = true;
269           }
270           else if (dir == INC_DIR)
271           {
272             possible = false;
273           }
274         }
275       }
276     }
277     return possible;
278   }
279
280
281
282
283   /*Find tray fuction moves the robot i posetive Y direction (Increase)
284     until the tray is found.
285     The Tray is found when the MAGNET_ENDSTOP turns low.
286     Returns true if tray is found and false if endstop it detected.
287     if endstop detekted inState is changed enstopTrigged state by  the checkFailure();
288   */
```

```
289
290   boolean moveToTray() {
291     //Enable motors
292     enableMotors();
293     bool trayFound = false;
294    // printEndstopsString();
295     while (!limitTrigged() && ATDConverter(MAGNET_ENDSTOP)) {
296       moveYMotor(INC_DIR);
297       if (!ATDConverter(MAGNET_ENDSTOP)) {
298         trayFound = true;
299       }
300       if (limitTrigged()) {
301         trayFound = false;
302       }
303     }
304     return trayFound;
305   }
306
307
308
```

```
1
2   /*Different calculations for the RoeBot*/
3
4   int calculateDistance(int X0, int X1, int Y0, int Y1) {
5     long calculatedDistance;
6     int deltaX = X1 - X0;
7     int deltaY = Y1 - Y0;
8     calculatedDistance = sqrt(sq(deltaX) + sq(deltaY));
9     return calculatedDistance;
10  }
11  /*****************************************************/
12  int calculateQuotient(int X0, int X1, int Y0, int Y1, int Z0, int Z1) {
13    long calculatedQuotient;
14    int deltaX = X1 - X0;
15    int deltaY = Y1 - Y0;
16    //int deltaZ = Z1-Z0;
17    calculatedQuotient = (deltaX / deltaY);
18    return calculatedQuotient;
19  }
20
21  /** Calculates mm input to step output for given axis.
22  */
23  int MMToSteps(float millimeter, int axis) {
24    int steps = 0;
25    if (axis == X_AXIS) {
26      steps = X_STEP_PER_MM * millimeter;
27    } else if (axis == Y_AXIS) {
28      steps = Y_STEP_PER_MM * millimeter;
29    }
30    return steps;
31  }
32
33  /** Calculates step input to mm output for given axis.
34  */
35  float stepsToMM(int steps, int axis) {
36    float mm = 0;
37    if (axis == X_AXIS) {
38      mm = (steps / X_STEP_PER_MM) ;
39    } else if (axis == Y_AXIS) {
40      mm = (steps / Y_STEP_PER_MM);
41    }
42    return mm;
43  }
44
45
46  /*****************************************************/
47  float stepsPerMM(int stepsPerRev, float driverMicrosteps, int belt_pitch, int
    pulley_teeth) {
48    float result;
49    result = ((stepsPerRev * driverMicrosteps) / (belt_pitch * pulley_teeth));
50    return result;
51  }
52
53  float MMPerSteps(int belt_pitch, int pulley_teeth, int stepsPerRev, float
    driverMicrosteps) {
54    float result;
55    result = ((belt_pitch * pulley_teeth) / (stepsPerRev * driverMicrosteps));
56    return result;
57  }
58  /*****************************************************/
59  //Find the direction for x movement
60  int dirX() {
61    int directionX;
62
63    if (currentPos.x > newPos.x)
64    {
65      directionX = DEC_DIR;
66    }
67    else if (currentPos.x < newPos.x)
68    {
69      directionX = INC_DIR;
70    }
71    else if (currentPos.x == newPos.x)
```

```
    {
      directionX = 0;
    }
    return directionX;
  }

  /*****************************************************/
  //Find the direction for y movement
  int dirY() {
    int directionY;
    if (currentPos.y > newPos.y)
    {
      directionY = DEC_DIR;
    }
    else if (currentPos.y < newPos.y)
    {
      directionY = INC_DIR;
    }
    else if (currentPos.y == newPos.y)
    {
      directionY = 0;
    }
    return directionY;
  }
```

```
// Converts an analog innput value to an bool.
// To be used for digital switches connected to analog inputs. (Analog to digital converter) .
bool ATDConverter(short value) {
  bool val = false;
  if (analogRead(value) >= 512) {
    val = true;
  } else {
    val = false;
  }
  return val;
}

// Prints the status og all endstops.
void printEndstopsString() {
  Serial.print("Magnet Endstop:  ");
  Serial.println(ATDConverter(MAGNET_ENDSTOP));
  Serial.print("X home:  ");
  Serial.println(ATDConverter(X_HOME_SWITCH));
  Serial.print("X Away:  ");
  Serial.println(ATDConverter(X_OUTER_SWITCH));
  Serial.print("Y home:  ");
  Serial.println(ATDConverter(Y_HOME_SWITCH));
  Serial.print("Y Away:  ");
  Serial.println(ATDConverter(Y_OUTER_SWITCH));

}


```

```
/***  LED CONTROL  ***/

//Set the leds to disco
void ledDiscoShow()
{
  //
  static uint8_t startIndex = 0;
  startIndex = startIndex + 1; /* motion speed */
  uint8_t colorIndex = startIndex;
  // Pre generated show for led strips.
  CRGBPalette16 currentPalette = RainbowColors_p;

  // Brightness for the ledds.
  uint8_t brightness = 30;

  for ( int i = 0; i < NUM_LEDS; i++) {
    leds[i] = ColorFromPalette( currentPalette, colorIndex, brightness, LINEARBLEND);
    colorIndex += 3;
  }
  FastLED.show();
}


// Chnage the color off all leds to the given spectre
void setLedColor(short red, short green, short blue) {

  for (int i = 0; i <= NUM_LEDS; i++) {
    leds[i] = CRGB (red, green, blue);
  }
  if (debug) {
    Serial.println("Led color changed to");
    Serial.print("Red:  ");
    Serial.println(red);
    Serial.print("Green:  ");
    Serial.println(green);
    Serial.print("Blue:  ");
    Serial.println(blue);
  }
  FastLED.show();
}
```

```
// Turns on the magnet.
void magnetON() {
  digitalWrite(MAGNET_PIN, HIGH);
}

// Turns on or off the magnet.
void magnetOFF() {
  digitalWrite(MAGNET_PIN, LOW);
}

```

```
/*** All the safety functions ***/



//Set the
void stopMotors() {
  mainCommand = stopRobot;
}









/**
   Checks the limit switches and sets the appropriate bools
*/
bool limitTrigged()
{
  bool limitIsTrigged = false;

  //Check the X home switch
  if (!digitalRead(X_HOME_SWITCH))
  {
    xHomeSwitch = true;
    limitIsTrigged = true;
  }
  else
  {
    xHomeSwitch = false;
  }

  //Check the X outer switch
  if (!digitalRead(X_OUTER_SWITCH))
  {
    xAwaySwitch = true;
    limitIsTrigged = true;
  }
  else
  {
    xAwaySwitch = false;
  }

  //Check the Y Home switch
  if (!digitalRead(Y_HOME_SWITCH))
  {
    yHomeSwitch = true;
    limitIsTrigged = true;
  }
  else
  {
    yHomeSwitch = false;
  }

  //Check the Y outer switch
  if (!digitalRead(Y_OUTER_SWITCH))
  {
    yAwaySwitch = true;
    limitIsTrigged = true;
  }
  else
  {
    yAwaySwitch = false;
  }


  return limitIsTrigged;
}
```

```
/**
  This function checks the Serial input buffer.

   It contains an switch case holding on all possible commands and requests.
  This devise will only read and handle data ment for it by compering the indata
  with the DEVICE_NAME. If the data isnt for this device the buffer will be
flushed,

**/


void receiveSerialEvent() {

  //Check if wire is ready to be read from/incomming bits
  if (Serial.available() > 0)
  {
    //Check for the device ID
    if (Serial.find(DEVICE_NAME)) {
      if (debug) {
       Serial.println("This massage is for me!!");
      }

      //Nr of bytes in this response
      short nrOfInts = 3;
      int response[nrOfInts] = {0, 0, 0};
      int cnt = 0;
      // Reads incomming command
     int inCommand = Serial.parseInt(); // Secound Byte represents an command.

      // Serial.write(inCommand);
      // Updates the commands
      recieveCommand = inCommand;


     //--------------------RECIVE SWITCH CASE----------------------
      //Switch command to perform the tasks given by address
      switch (recieveCommand)
      {
        //-----------------
        //Command for moving the robot
        //TODO: READ INCOMMING BYTES AND STORE THEM IN DESIRED VARIABLES  (SOLVED)
        case moveRobotTo:
          if (debug) {
           Serial.println(F("*****DO MOVEROBOT TO RECOGNIZED*****"));
          }


         /***Incomming values parsing****/
         //Set the new incomming positions
```

```
          newPos.x = MMToSteps(Serial.parseFloat(), X_AXIS);
          newPos.y = MMToSteps(Serial.parseFloat(), Y_AXIS);
           if (debug) {
            Serial.println(X_STEP_PER_MM);
            Serial.println(Y_STEP_PER_MM);
            Serial.print(F("New x pos in step: " ));
            Serial.println( newPos.x );
            Serial.print(F("New y pos in step: " ));
            Serial.println( newPos.y );
           }
          mainCommand = moveRobotTo;
           break;


//-------------------------------------------------------------------------
        // Recived cmd for calibrating the robot.
        case stopRobot:
          if (debug) {
           Serial.println(F("**STOP ROBOT***"));
          }
          //sendInt(ACK); // Send an Acknowledge
          mainCommand = stopRobot;
          break;
        //-----------------


        //Moves in Y dir to tray endstop is low
        case findTray:
          if (debug) {
           Serial.println(F("***** DO FIND TRAY *****"));
          }
          //sendInt(ACK); // Send an Acknowledge
          mainCommand = findTray;

          break;






//-------------------------------------------------------------------------
        // Recived cmd for calibrating the robot.
        case doCalibrate:
          if (debug) {
           Serial.println(F("**DO CALIBRATE RECOGNIZED***"));
          }
          //sendInt(ACK); // Send an Acknowledge
          mainCommand = doCalibrate;
```

```
          break;
        //-----------------


//-------------------------------------------------------------------------
        // Recived cmd for calibrating the robot.
        case magnetOn:
          if (debug) {
           Serial.println("*****DO MAGNET NO*****");
          }
          //sendInt(ACK); // Send an Acknowledge
          mainCommand = magnetOn;
          break;




     //-------------------------------------------------------------------------
        // Recived cmd for calibrating the robot.
        case magnetOff:
          if (debug) {
           Serial.println("*****DO MAGNET NO*****");
          }
          //sendInt(ACK); // Send an Acknowledge
          mainCommand = magnetOff;
          break;



//-------------------------------------------------------------------------------
----------
        //Chnage color command receved.
        case changeColor:
          if (debug) {
           Serial.println("*****DO CANGE THE COLOR ON THE LEDS*****");
          }

          //sendInt(ACK); // Send an Acknowledge
          for (int ledColorInt = R; ledColorInt <= B; ledColorInt++)
          {
           ledRGB[ledColorInt] = Serial.parseInt();
          }
          mainCommand = changeColor;
          break;



//-------------------------------------------------------------------
```

```
        //Chnage color command receved.
        case discoLight:
          if (debug) {
           Serial.println("*****DO CANGE THE COLOR ON THE LEDS*****");
          }
         // sendInt(ACK); // Send an Acknowledge
          mainCommand = discoLight;
          break;



//------------------------------------------------------------------------
        //Changing velocity command receved
        case changeVelocity:
          if ( debug) {
           Serial.println("***** CANGE VELOCETY  *****");
          }
         // sendInt(ACK); // Send an Acknowledge

          stepSpeed[X_AXIS] = Serial.parseInt();
          stepSpeed[Y_AXIS] = Serial.parseInt();

          if (debug) {
            Serial.print("X velocety: ");
            Serial.println(stepSpeed[X_AXIS]);
            Serial.print("Y velocety: ");
            Serial.println(stepSpeed[Y_AXIS]);
          }
          break;


        /**** REQUEST EVENTS ****/
       //------------------------------------------------------------------------
          // Retrive state command recived.
        case retrieveState:
          if (debug) {
           Serial.println("*******RETRIVE STATE*******:");
          }
         // sendInt(ACK); // Send an Acknowledge
          sendInt(inState);
          break;

//------------------------------------------------------------------------
        //Send the calibrated parameters
        case calibParam:
          if (debug) {
           Serial.println("********CALIB PARAM *******");
          }
         // sendInt(ACK); // Send an Acknowledge
```

```
          //***Add the values******
          response[cnt++] =  parameter;
          response[cnt++] =  stepsToMM(maxPos.x, X_AXIS);
          response[cnt++] =  stepsToMM(maxPos.y, Y_AXIS);
          sendIntegers(response, nrOfInts);
          break;



//--------------------------------------------------------------------------
        //Set the command for calibrating the robot.
        default:
          if (debug) {
           Serial.println("*****DAFAULT CASE*****");
           Serial.println("Invalid or not found command");
          }
         // sendInt(NACK); // Send an Negative-Acknowledge
          break;
          //-----------------
      }
    } else {
      Serial.flush();
    }
  }
}
```

```
/*Function responsible for sending data over the serial port. */




// Sends one byte over and serial communication using string.
// Uses devise name in front and devide the byte witn an , as delimiter
// writes to the serial port.
void sendInt(int intToSend) {
  //Make the string, with device ID, int and ending
  String stringToSend = DEVICE_NAME;
  stringToSend = stringToSend + ",";
  stringToSend = stringToSend + intToSend;
  stringToSend = stringToSend + "\n";



   // Make char pointer and dynamically allocate the memory for all the chars the
pointer have access to
  char* cString = (char*) malloc(sizeof(char) * (stringToSend.length() + 1));
  //Use toChar function to split all the char's in the string. They will be
allocated to the memory the pointer has control of.
 stringToSend.toCharArray(cString, stringToSend.length() + 1);

  //Send the chars to the serial port
 Serial.write(cString);
  // Free the allocated memory.
  free(cString);

  if (debug) {
    Serial.println();
  }
}




// Sends multiple integers over the buss line.
void sendIntegers(int integers[], int arrayLength) {

  String stringToSend = DEVICE_NAME;
  //Iterate the integer array and add them to a String.
  for (int i = 0; i <  arrayLength; i++) {
    stringToSend = stringToSend + "," + integers[i];
  }
  //End the string
```

```
  stringToSend = stringToSend + "\n";



    //Give the char pointer the memory for all the chars it should have control
over
  char* cString = (char*) malloc(sizeof(char) * (stringToSend.length() + 1));
  //Use toChar function to split all the char's in the string. They will be
allocated to the memory the pointer has control of.
  stringToSend.toCharArray(cString, stringToSend.length() + 1);
  //Send the chars to the serial port
 Serial.write(cString);

  // Free the allocated memory.
  free(cString);
  if (debug) {
    Serial.println();
  }
}
```

I.2 Vertical System Code

```
#include <Stepper.h>
#include <Wire.h>


//debugging bools
bool cal = false;
// Debug over 1 for normal debug,
// Debug = 1, calibration debug
// Debug = 2, move case debug
// Debug = 3 for steps
int debug = false;


/******************** STEPPER VARIABLES ***********************/
#define MOTOR_BREAK 7
#define PUMP 6
//Left motor
#define LEFT_MOTOR_STEP 13
#define LEFT_MOTOR_DIR 12
#define LEFT_MOTOR_ENABLE 11
#define LEFT_BOTTOM_SWITCH 28
#define LEFT_TOP_SWITCH 26
#define LEFT_DIR_UP 1
#define LEFT_DIR_DOWN -1
//Right motor
#define RIGHT_MOTOR_STEP 10
#define RIGHT_MOTOR_DIR 9
#define RIGHT_MOTOR_ENABLE 8
#define RIGHT_BOTTOM_SWITCH 36
#define RIGHT_TOP_SWITCH 35
#define RIGHT_DIR_UP -1
#define RIGHT_DIR_DOWN 1

/******** STOP INTERRUPT PIN ***********/
#define STOP_INTERRUPT_PIN 2

//Safety IR
#define TOP_IR_RELAY 18
#define BOTTOM_IR_RELAY 19

/******** STEPPER SPESIFICATIONS ***********/
#define STEP_PER_REVOLUTION 200 // Stepps per rev on the motor (FULL STEP)
#define DRIVER_MICROSTEPS_FACTOR 4 // Micro stepp factor.
#define PULLEY_RADIUS 12.16
#define BELT_PITCH_RADIUS 12.74
#define PULLER_CIRCUMFERENCE (2*BELT_PITCH_RADIUS)*PI
#define BELT_PRESENT_PITCH 5 //mm
#define PULLEY_TOOTH 16 // Number of tooths on pulley.

//Variable to hold steps pr mm
#define STEP_PER_MM ((STEP_PER_REVOLUTION * DRIVER_MICROSTEPS_FACTOR * 4) /
(PULLER_CIRCUMFERENCE))


// Falgs size in Z diraction measured in mm
#define FLAG_SIZE 25 //mm
//Default steps recognized as flags
const int flagDetectionSteps = FLAG_SIZE * STEP_PER_MM;
// 5% of the falag size to be used for safy barriere
const int safetyFactorOffset = flagDetectionSteps * 2; // Multiply with safety offset;



/************** Serial Communication *************/
#define DEVICE_NAME "dev2"

/************** The maximun number of trays posible for this system *************/
#define MAX_NR_OF_TRAYS 10



//Default stepper speed
int stepSpeed = 200;
```

```
73  //Save traveled distance here
74  int travelledDistance;
75
76
77  int barrierCounter = 0;
78  int barrierCounterTop = 0;
79  int barrierCounterBottom = 0;
80
81
82
83  /********************** STRUCTURES *************************/
84
85  //Structure to hold positions
86  typedef struct Pos {
87    int zLeft;
88    int zRight;
89  } Pos;
90
91
92  /**
93    Struct to represent tray
94  */
95  typedef struct Tray
96  {
97    Pos zPos;
98    int trayNumber;
99  } Tray;
100
101 /**
102    Struct to hold all the trays
103 */
104 typedef struct TrayRegister
105 {
106   int nrOfTrays;
107   Tray trayList[MAX_NR_OF_TRAYS];
108 } TrayRegister;
109
110 //Create tray register
111 TrayRegister trayReg;
112
113
114 //The position variables
115 Pos currentPos;
116 Pos maxPos;
117 Pos newPos;
118 Pos lastFlagPos;
119
120
121 /************ SWITCH variables ************/
122 //Limit switch booleans
123 boolean leftBottomSwitch = false;
124 boolean leftTopSwitch = false;
125 boolean rightBottomSwitch = false;
126 boolean rightTopSwitch = false;
127 //Safety barrier
128 boolean topSafetyBarrier = false;
129 boolean bottomSafetyBarrier = false;
130 //Bool to keep track if the brake is on or not
131 boolean brake = false;
132
133
134 //Set up the motors
135 Stepper motor_left(STEP_PER_REVOLUTION * DRIVER_MICROSTEPS_FACTOR, LEFT_MOTOR_STEP,
    LEFT_MOTOR_DIR);
136 Stepper motor_right(STEP_PER_REVOLUTION * DRIVER_MICROSTEPS_FACTOR,
    RIGHT_MOTOR_STEP, RIGHT_MOTOR_DIR);
137
138
139 //-------------------------------ENUMS-----------------------------------------------
140 //ENUMS to keep track of what has been sent to the arduino, and what it should do.
141 //The state of the arduino
142 enum state {
143   busy = 0x50,
```

```
144     readyToRecieve = 0x51,
145     emc = 0x60,
146     fail = 0x67,
147     encoderOutOfSync = 0x65,
148     elevatorLimitTrigg = 0x63,
149     stopped = 0x52,
150     safetySwitchLower = 0x62,
151     safetySwitchUpper = 0x61,
152     parameter = 0x70
153   };
154
155   /**
156    * *COMMANDS* the Master can send to the slave.
157   */
158   enum command {
159     moveRobot = 0x05,
160      doSuction = 0x06,
161     stopRobot = 0x07,
162     doCalibrate = 0x10,
163     changeVelocity = 0x20,
164     changeAcceleration = 0x21,
165    // vacuumOn = 0x24,
166    // vacuumOff = 0x25,
167     retrieveState = 0x30,
168     calibParam = 0x31,
169
170     //Main loop states
171     idle = 0x01,
172     EMC = 0x02,
173     failure = 0x03
174   };
175
176
177   /**
178      The enum for deciding the moving conditions
179   */
180   enum moveState
181   {
182     normalMove = 0,
183     safetyMoving = 1,
184     endStopMoving = 2,
185     moveNotPossible = 3,
186     doneMoving = 4
187   };
188
189
190   //ENUM For acknowledgement  og negative-acknowledgement
191   enum acknowlagement {ACK = 1, NACK = 0};
192
193
194   //Create enums
195   enum state inState;             //To keep track of which STATE the arduino currently
      is in. Busy when doing stuff, ready/idle when not. EMC, encoder failure, ETC...
196   enum command mainCommand;       //Switch case command for the main loop
197   enum command recieveCommand;    //Switch case command for the RECIEVE command,
      reading storing values to be done later.
198   enum moveState movingState;
199   //enum command checkCommand;
200
201
202
203   int lastPrintStep;
204
205
206
207
208
209
210   //------------------------------SETUP----------------------------------------------------
      -------------------
211
212   void setup() {
213
```

```
214     // Start Serial communication
215     Serial.begin(115200);
216 
217     //Debug
218     if (debug > 0)
219     {
220       currentPos.zLeft = 0;
221       currentPos.zRight = 0;
222       Serial.println(stepsToMM(currentPos.zLeft));
223     }
224     if (!debug) {
225       //Setup the tray register
226       trayReg.nrOfTrays = 0;
227     }
228 
229     //Set the pin modes
230     pinMode(LEFT_BOTTOM_SWITCH, INPUT);
231     pinMode(LEFT_TOP_SWITCH, INPUT);
232     pinMode(RIGHT_BOTTOM_SWITCH, INPUT);
233     pinMode(RIGHT_TOP_SWITCH, INPUT);
234     pinMode(MOTOR_BREAK, OUTPUT);
235     pinMode(PUMP, OUTPUT);
236 
237     //Safery barrier sensors
238     pinMode(TOP_IR_RELAY, INPUT_PULLUP);
239     pinMode(BOTTOM_IR_RELAY, INPUT_PULLUP);
240 
241 
242     /******** INTERRUPT PIN ***********/
243     pinMode(STOP_INTERRUPT_PIN, INPUT_PULLUP);
244     attachInterrupt(digitalPinToInterrupt(STOP_INTERRUPT_PIN), stopMotors, FALLING);
245 
246     turnOnBrake();
247     mainCommand = idle;
248 
249   }
250 
251 
252 
253 
254   //--------------------------------------------------------------------------------
      --------------
255   void loop()
256   {
257 
258 
259     // Recives an event for serial com
260     receiveSerialEvent();
261 
262 
263 
264     //Switch command for the
265 
      //--------------------------------------------------------------------------------
      ----------------
266     switch (mainCommand)
267     {
268 
        //--------------------------------------------------------------------------------
        ------------------
269       //The arduino has nothing to do, and is therefore in idle
270       case idle:
271       //Check debug flag
272         if (debug) {
273           Serial.println(F("Main: Idle"));
274           delay(1000);
275         }
276 
277         turnOnBrakeDisableMotors();
278         delay(5);
279         updateStatusAndNotify(readyToRecieve);
280         break;
281 
```

```
282
283       case moveRobot:
284          //Check debug flag
285         if (debug) {
286           Serial.println(F("Main: move robot"));
287         }
288         //Set state to busy
289         updateStatusAndNotify(busy);
290
291         /*If endstop is reached check if it can continue to move
292         Check which endstop is reached, and which direction it can move
293         If it cant move in the preferred direction(set to by move)
294         Switch state to endstopReached, and bypass the moveRobot command
295         Do the moving*/
296
297         //Switch case for controlling the moving
298         switch (movingState)
299         {
300           case normalMove:
301             if ((!safetyBarrier()) && !limitTrigged())
302             {
303               //Check if should continue moving, also moves while checking if
304               //Do normal moving
305               if (moveStep(newPos))
306               {
307                 //Moving normally so reset the barrier counter
308                 updateCntBottomBarrier();
309                 updateCntTopBarrier();
310               }
311               //Done moving
312               else
313               {
314                 movingState = doneMoving;
315               }
316
317             }
318             //One of the end stops are triggered
319             else if (limitTrigged())
320             {
321               movingState = endStopMoving;
322             }
323             //The safety barrier is trigged
324             else if (safetyBarrier())
325             {
326               movingState = safetyMoving;
327             }
328             break;
329
330
331           //Moving while the safety barrier is trigged
332           /*Cases that occur while moving when safety is trigged:
333             1. End-stop sensors can be triggered, direction has to be checked before
                moving then
334             2. Can go outside of pre-determined safety steps -> Send to failure/not
                possible to move
335             3. While inside of pre-determined safety steps -> check if safety barrier
                is still active, it its not. Then the "obstacle" was a flag, and save the
                z positions to a new tray /-> Continues to move if needed.
336           */
337           case safetyMoving:
338             //Maybe add check of which sensor is hit - should be added when
339             //Safety barrier is hit
340
341             //Check if endstop is hit and is direction safe!
342             if (limitTrigged() && !isDirectionSafe(newPos))
343             {
344               if (debug == 2)
345               {
346                 Serial.println("Direction not safe");
347               }
348               sendInt(NACK);
349               movingState = moveNotPossible;
350             }
```

```
351
352             //Check if the safety margins are still OK
353             else if (!counterExceeded())
354             {
355               //Check if safetybarrier is still activated, if its not, it means the
                  robot is barrier outside any danger zone
356               if (!safetyBarrier())
357               {
358                 if (debug == 2)
359                 {
360                   Serial.println("SafetyBarrier exit'ed");
361                 }
362
363                 //Return to normal moving
364                 movingState = normalMove;
365               }
366               //Default steps are not overrided yet, so continue moving
367               else if (moveStep(newPos))
368               {
369                 updateCntBottomBarrier();
370                 updateCntTopBarrier();
371               }
372               //Done moving
373               else
374               {
375                 movingState = doneMoving;
376               }
377
378             }
379             //BarrierCounter exceeded
380             else
381             {
382               //Set moving state to not possible
383               movingState = moveNotPossible;
384             }
385             break;
386
387           //End stops are triggered
388           case endStopMoving:
389             if (debug == 2)
390             {
391               Serial.println("EndstopMoving");
392             }
393             //check if endstop is still active
394             if (limitTrigged())
395             {
396               //Check direction
397               if (isDirectionSafe(newPos))
398               {
399                 if (debug == 2)
400                 {
401                   Serial.println("dir safe - moving");
402                 }
403
404                 //Do moving and check if moving is done
405                 if (!moveStep(newPos))
406                 {
407                   if (debug == 2)
408                   {
409                     Serial.println("moving done");
410                   }
411
412                   movingState = doneMoving;
413                 }
414               }
415               //Moving is not possible because of direction
416               else
417               {
418                 if (debug == 2)
419                 {
420                   Serial.println("dir unsafe");
421                 }
422
```

```
423                 movingState = moveNotPossible;
424               }
425             }
426             //Move normally
427             else
428             {
429               movingState = normalMove;
430             }
431             break;
432
433           //Not possible to move - SafetyMoving has exceed move range or try to move
              in same direction as the triggered end stop
434           case moveNotPossible:
435             if (debug == 2)
436             {
437               Serial.println("MoveNotPossible");
438             }
439             mainCommand = failure;
440             break;
441
442           //The position has been reached
443           case doneMoving:
444             if (debug == 2)
445             {
446               Serial.println("doneMoving");
447             }
448             turnOnBrake();
449             mainCommand = idle;
450             break;
451
452           default:
453             break;
454         }
455         break;
456
457
458
        //-------------------------------------------------------------------------------
        ------------------
459       //Calibrate the robot
460       case doCalibrate:
461         if (debug) {
462           Serial.println(F("Main: Do calibrate"));
463         }
464         //Set state
465         updateStatusAndNotify(busy);
466
467         //Calibrate
468         calibrationWithSafety();
469         //Set to idle
470         mainCommand = idle;
471         break;
472
473
474
475      //-----------------------------------------------------------------------
476       // Turn Off vacuum.
477       case doSuction:
478         if (debug) {
479           Serial.println("Main: DO SUCTION OFF *****");
480         }
481
482         updateStatusAndNotify(busy);
483         turnVacuumOnOff(1000);
484         mainCommand = idle;
485         break;
486
487
488
489
490
        //-------------------------------------------------------------------------------
        ------------------
```

```
491       //The arduino has recieved an STOP command
492       case stopRobot:
493         if (debug) {
494           Serial.println(F("Main: Stopped"));
495           Serial.println(F("Fucntion not made"));
496         }
497         turnOnBrakeDisableMotors();
498
499         // Update Status and notify
500         updateStatusAndNotify(stopped);
501
502         break;
503
504
505
          //-------------------------------------------------------------------------------
          ------------------
506       //The is in failure mode has recieved an STOP command
507       case failure:
508         if (debug) {
509           Serial.println(F("Failure"));
510         }
511
512         // Update Status and notify
513         updateStatusAndNotify(checkFailure());
514
515         turnOnBrakeDisableMotors();
516         if (debug) {
517           Serial.println("DELAY IS USED FOR HOLDING THE MOTOR UNTIL THE BRAKE IS SAFLY
              TURND ON");
518         }
519         delay(200);
520         mainCommand = idle;
521         break;
522
523
          //-------------------------------------------------------------------------------
          ------------------
524       //No command is recognized. The arduino should never reside in this case..
525       default:
526         if (debug == 3)
527         {
528           Serial.println("Command not recognised");
529           Serial.println("In default");
530         }
531         break;
532     }
533
534
535
        //-------------------------------------------------------------------------------
        ----------------
536
537   }
538
539
540
541
542
543
544
545
```

```
1
2   /**
3      Clibrate the Elevator.
4      Safety barriers is used.
5
6      Returns true if calibration was succesful, false if not
7   */
8   bool calibrationWithSafety()
9   {
10
11    // Turns on motors and disable the brakes
12    turnOnMotorsDisableBrakes();
13    bool calibrationFinished = true;
14
15    //states for moving
16    const int FIND_BOTTOM = 0;
17    const int FIND_TOP = 1;
18    const int FIND_FLAGS = 2;
19    const int FAILURE = 3;
20    const int DONE = 4;
21    bool moving = true;
22
23    //Calib pos
24    /* Pos calibPos;
25      calibPos.zLeft = currentPos.zLeft;
26      calibPos.zRight = currentPos.zRight;
27    */
28    int barrierCounterLeft = 0;
29    int barrierCounterRight = 0;
30    //setting state to go to bottom
31    int state = FIND_BOTTOM;
32
33
34    //Keep in loop until either finished or error
35    while (moving)
36    {
37      //running this state until it it is completed
38      switch (state)
39      {
40
41        case FIND_BOTTOM:
42          //finding (0,0) position, which is bottom
43          if (digitalRead(LEFT_BOTTOM_SWITCH))
44          {
45            //IF safetybarrier is activated, check the barrier counter
46            if (safetyBarrierTop() || safetyBarrierBottom())
47            {
48              if (barrierCounterLeft < (flagDetectionSteps + safetyFactorOffset) &&
                !counterExceeded())
49              {
50                moveLeftMotor(LEFT_DIR_DOWN);
51                ++barrierCounterLeft;
52              }
53              else
54              {
55                state = FAILURE;
56              }
57            }
58            //No problem moving
59            else
60            {
61              if (debug == 1) {
62                Serial.println("Moving LEFT with safety OK");
63              }
64
65              moveLeftMotor(LEFT_DIR_DOWN);
66              barrierCounterLeft = 0;
67            }
68          }
69
70          if (digitalRead(RIGHT_BOTTOM_SWITCH))
71          {
72            //IF safetybarrier is activated, check the barrier counter
```

```
73            if (safetyBarrierTop() || safetyBarrierBottom())
74            {
75              //Check the safety counters
76              if ((barrierCounterRight < (flagDetectionSteps + safetyFactorOffset)) &&
                !counterExceeded())
77              {
78                moveRightMotor(RIGHT_DIR_DOWN);
79                ++barrierCounterRight;
80              }
81              else
82              {
83                state = FAILURE;
84              }
85            }
86            //No problem moving
87            else
88            {
89              if (debug == 1) {
90                Serial.println("Moving RIGHT with safety OK");
91              }
92
93              moveRightMotor(RIGHT_DIR_DOWN);
94              barrierCounterRight = 0;
95            }
96          }
97          //Both endstops are activated, so bottom is found
98          if (!digitalRead(LEFT_BOTTOM_SWITCH) && !digitalRead(RIGHT_BOTTOM_SWITCH))
99          {
100           currentPos.zLeft = 0;
101           currentPos.zRight = 0;
102           state = FIND_TOP;
103           if (debug == 1) {
104             Serial.println("reached bottom");
105           }
106         }
107         break;
108
109       //------------------------------------------------------------------------
110       case FIND_TOP:
111         // run left motor and count steps until away position is reached
112         if (digitalRead(LEFT_TOP_SWITCH))
113         {
114           //IF safetybarrier is activated, check the barrier counter
115           if (safetyBarrierTop() || safetyBarrierBottom())
116           {
117             if ((barrierCounterLeft < (flagDetectionSteps + safetyFactorOffset)) &&
                !counterExceeded())
118             {
119               moveLeftMotor(LEFT_DIR_UP);
120               ++barrierCounterLeft;
121             }
122             else
123             {
124               state = FAILURE;
125             }
126           }
127           //No problem moving, move normally
128           else
129           {
130             moveLeftMotor(LEFT_DIR_UP);
131             barrierCounterLeft = 0;
132           }
133
134         }
135
136         // run y-axis motor and count stepil away position is reached
137         if (digitalRead(RIGHT_TOP_SWITCH))
138         {
139           //IF safetybarrier is activated, check the barrier counter
140           if (safetyBarrierTop() || safetyBarrierBottom())
141           {
142             if ((barrierCounterRight < (flagDetectionSteps + safetyFactorOffset)) &&
                !counterExceeded())
```

```
            {
              moveRightMotor(RIGHT_DIR_UP);
              ++barrierCounterRight;
            }
            else
            {
              state = FAILURE;
            }
          }
          //No problem moving
          else
          {
            moveRightMotor(RIGHT_DIR_UP);
            barrierCounterRight = 0;
          }
        }


        // if left and right motor is at their top positions set variable to true
        if (!digitalRead(LEFT_TOP_SWITCH) && !digitalRead(RIGHT_TOP_SWITCH))
        {

          //Save the found variables
          //Save the one with the least steps
          if (currentPos.zLeft < currentPos.zRight)
          {
            maxPos.zLeft = currentPos.zLeft;
            maxPos.zRight = -currentPos.zLeft;
            currentPos.zRight = -currentPos.zLeft;
          }
          else
          {
            maxPos.zRight = currentPos.zRight;
            maxPos.zLeft = -currentPos.zRight;
            currentPos.zLeft = -currentPos.zRight;
          }


          state = FIND_FLAGS;

          if (debug == 1)
          {
            Serial.println("reached top");
          }

        }
        break;

      //Go down and find the trays
      case FIND_FLAGS:
        if (debug == 1)
        {
          Serial.println("Finding flag");
        }
        //IF safetybarrier is activated, check the barrier counter
        if (digitalRead(LEFT_BOTTOM_SWITCH) && digitalRead(RIGHT_BOTTOM_SWITCH))
        {
          if (debug == 1)
          {
            printSteps();
          }

          //IF safetybarrier is activated, check the barrier counter
          if (safetyBarrierTop() || safetyBarrierBottom())
          {
            if (debug == 1)
              Serial.println("FindFlag - SafetyBarr");
            //Check for the bottom safety barrier
            if (safetyBarrierBottom())
            {
              if (debug == 1) {
                Serial.println("Safety Bottom");
              }
```

```
216
217                 //Check if counter is not exceeded, then move the motors and update
                    counters
218                 if (!counterExceeded())
219                 {
220                   moveLeftMotor(LEFT_DIR_DOWN);
221                   moveRightMotor(RIGHT_DIR_DOWN);
222                   //Check safety barriers
223                   if (topSafetyBarrier)
224                   {
225                     ++barrierCounterTop;
226                   }
227                   else
228                   {
229                     barrierCounterTop = 0;
230                   }
231                   if (bottomSafetyBarrier)
232                   {
233                     ++barrierCounterBottom;
234                   }
235                   else
236                   {
237                     barrierCounterBottom = 0;
238                   }
239
240
241                   //Check if safety barrier is still active
242                   //IF NOT - it means a flag has been detected
243                   if (!safetyBarrierBottom())
244                   {
245                     if (debug == 1) {
246                       Serial.println("Creating tray");
247                     }
248                     createTrayOnPos();
249                     if (debug) {
250                       Serial.print("Nr of trays in reg: ");
251                       Serial.print(trayReg.nrOfTrays);
252                     }
253                   }
254                 }
255                 //Set state ti failure as counter safety counter has exceed its limits
256                 else
257                 {
258                   state = FAILURE;
259                 }
260
261
262               }
263               //Top safety barrier was not activated
264               else
265               {
266                 //Check if counter is not exceeded, then move the motors and update
                    counters
267                 if (!counterExceeded())
268                 {
269                   moveLeftMotor(LEFT_DIR_DOWN);
270                   moveRightMotor(RIGHT_DIR_DOWN);
271                   updateCntBottomBarrier();
272                   updateCntTopBarrier();
273                 }
274                 else
275                 {
276                   state = FAILURE;
277                 }
278               }
279
280
281             }
282             //No problem moving
283             else
284             {
285               if (debug == 1) {
286                 Serial.println("Moving normal");
```

```
            }

            moveRightMotor(RIGHT_DIR_DOWN);
            barrierCounterRight = 0;
            moveLeftMotor(LEFT_DIR_DOWN);
            barrierCounterLeft = 0;

            //Update the counter barrier
            updateCntBottomBarrier();
            updateCntTopBarrier();
          }
        }
        else
        {
          state = DONE;
        }
        break;


      //Exit the moving loop
      case DONE:
        moving = false;
        break;


      //Failure happened during the calibration
      case FAILURE:
        calibrationFinished = false;
        moving = false;
        break;
    }
  }
  //Calibration exits, Turn On brakes and return finish bool
  turnOnBrakeDisableMotors();
  return calibrationFinished;
}
```

```
/**
   FUNCTIONS TO CONTROL THE MOTORS
*/

//Set the enable signal for the motors HIGH
void enableMotors()
{
  digitalWrite(LEFT_MOTOR_ENABLE, HIGH);
  digitalWrite(RIGHT_MOTOR_ENABLE, HIGH);
}

//Set the enable signal for the motors LOW
void disableMotors()
{
  digitalWrite(LEFT_MOTOR_ENABLE, LOW);
  digitalWrite(RIGHT_MOTOR_ENABLE, LOW);
}

//Turn the brakes on
void turnOnBrake()
{
  digitalWrite(MOTOR_BREAK, LOW);
  brake = true;
}

//Turn the brakes off
void turnOffBrake()
{
  digitalWrite(MOTOR_BREAK, HIGH);
  brake = false;
}

//Turns on brake, waits given time before disable motors.
void turnOnBrakeDisableMotors() {
  turnOnBrake();
  delay(10);
  disableMotors();
}

// Enables motor waits given time before turning of motors.
void turnOnMotorsDisableBrakes() {
  enableMotors();
  turnOffBrake();
}
```

```
/**
   Slope of a line can be expressed as deltaX/deltaY. A number of steps on Y equals
   a number of steps on X.
   This will help us step along Y and adds deltaX to a counter, when the counter >=Y
   steps a number on X.'
   The motors will toggle simultaniously to the newX and newY.
*/
//Move the stepper motors in the direction it new pos indicates.
//Returns false if it should move more, true if it is in new pos
boolean moveStep(Pos newZ) {

  // Turns on motors and disabled the brakes
  turnOnMotorsDisableBrakes();

  int deltaZ = newZ.zLeft - currentPos.zLeft; //distance to move

  //Return bool to indicate if the steppers still need to move to reach position
  boolean stillMoving = false;


  //Check if X should move at all
  if (deltaZ != 0)
  {
    //Increment currentpos
    moveLeftMotor(dirZLeft(newZ.zLeft));
    moveRightMotor(dirZRight(newZ.zRight));

    //Set the return bool to indicate further actions are required
    stillMoving = true;
  }
  //No moving required
  else
  {
    if (debug == 1)
    {
      Serial.println("Done moving");
    }
  }


  // Turns on motors and disabled the brakes
  turnOnMotorsDisableBrakes();
  //Return the bool which indicated if position is reached or not
  return stillMoving;

}


//Move the left motor
void  moveLeftMotor(int dir)
{
  motor_left.step(dir);
  motor_left.setSpeed(stepSpeed);
  increaseLeftStep(dir);
}

//Move the right motor
void  moveRightMotor(int dir)
{
  motor_right.step(dir);
  motor_right.setSpeed(stepSpeed);
  increaseRightStep(dir);
}

/**
   Increase the step in left direction with one
*/
void increaseLeftStep(int dir)
{
  currentPos.zLeft = (currentPos.zLeft + dir);
}
void increaseRightStep(int dir)
{
```

```
72      currentPos.zRight = (currentPos.zRight + dir);
73    }
74
75    /*****************************************************/
76
77
```

```
2   /***Different calculations for the Arduino***/
```

```
/*** All the safety functions for the arduino ***/



void stopMotors() {
  mainCommand = stopRobot;
}

/**
    Check all the end stop sensors.
    Will set all the appropriate bools.
    Returns true if one or more sensors is trigged
*/
bool limitTrigged()
{
  bool limitIsTrigged = false;

  //Left bottom switch, set Return to true
  if (!digitalRead(LEFT_BOTTOM_SWITCH))
  {
    leftBottomSwitch = true;
    limitIsTrigged = true;
  }
  //Set return to false
  else
  {
    leftBottomSwitch = false;
  }

  //Left top switch, set Return to true
  if (!digitalRead(LEFT_TOP_SWITCH))
  { leftTopSwitch = true;
    limitIsTrigged = true;
  }
  //Set return to false
  else
  {
    leftTopSwitch = false;
  }

  //Right bottom switch, set Return to true
  if (!digitalRead(RIGHT_BOTTOM_SWITCH))
  { rightBottomSwitch = true;
    limitIsTrigged = true;
  }
  //Set return to false
  else
  {
    rightBottomSwitch = false;
  }

  //Right top switch, set Return to true
  if (!digitalRead(RIGHT_TOP_SWITCH))
  {
    rightTopSwitch = true;
    limitIsTrigged = true;
  }
  //Set return to false
  else
  {
    rightTopSwitch = false;
  }


  return limitIsTrigged;
}


/*
   Return the state of the safety barrier(trigged TRUE else false) and set the
   respective booleans
*/
boolean safetyBarrier()
```

```
{
  boolean barrierTrigged = false;

  //Check Top IR sensor
  if (!digitalRead(TOP_IR_RELAY))
  {
    topSafetyBarrier = true;
    barrierTrigged = true;
  }
  else
  {
      topSafetyBarrier = false;
  }

  //Check bottom IR sensor
  if (!digitalRead(BOTTOM_IR_RELAY))
  {
    bottomSafetyBarrier = true;
    barrierTrigged = true;
  }
  else
  {
      bottomSafetyBarrier = false;
  }

  //Return the
  return barrierTrigged;
}

/**
   Returns state of top barrier
*/
boolean safetyBarrierTop()
{
  boolean barrierTrigged = false;

  if (!digitalRead(TOP_IR_RELAY))
  {
    topSafetyBarrier = true;
    barrierTrigged = true;
  }
  else
  {
      topSafetyBarrier = false;
  }

  return barrierTrigged;
}
/**
   Increase the active barriers
*/
void increaseActiveBarrier()
{
  updateCntTopBarrier();
  updateCntBottomBarrier();
}

/**
   Update the top barrier counter
*/
void updateCntTopBarrier()
{
  if (topSafetyBarrier)
  {
    barrierCounterTop += dirZLeft(newPos.zLeft);
  }
  else
  {
    barrierCounterTop = 0;
  }
}
/**
   Update the bottom barrier counter
```

```
146   */
147   void updateCntBottomBarrier()
148   {
149     if (bottomSafetyBarrier)
150     {
151       //Check if the robot is moving upwards, and if the bottom sensor is trigged,
      dont count
152       if(!(LEFT_DIR_UP == dirZLeft(newPos.zLeft)))
153       {
154         barrierCounterBottom += dirZRight(newPos.zRight);
155       }
156       //else, dont count
157     }
158     else
159     {
160       barrierCounterBottom = 0;
161     }
162   }
163
164
165   /**
166      Returns true if the safety counters is exceeded
167   */
168   boolean counterExceeded()
169   {
170     boolean failure = false;
171
172     //Check the counters
173     if ((barrierCounterTop > (flagDetectionSteps + safetyFactorOffset)) ||
      (barrierCounterBottom > (flagDetectionSteps + safetyFactorOffset)))
174     {
175       failure = true;
176     }
177
178     return failure;
179   }
180
181
182   /**
183      Returns state of bottom barrier
184   */
185   boolean safetyBarrierBottom()
186   {
187     boolean barrierTrigged = false;
188
189     if (!digitalRead(BOTTOM_IR_RELAY))
190     {
191       bottomSafetyBarrier = true;
192       barrierTrigged = true;
193     }
194     else
195     {
196       if (bottomSafetyBarrier)
197       {
198         bottomSafetyBarrier = false;
199       }
200     }
201     return barrierTrigged;
202   }
203
204
205
206   /**
207      Function to check if the new direction is safe
208   */
209   boolean isDirectionSafe(Pos newPos)
210   {
211
212     //The return bool if direction is safe
213     boolean safe = false;
214     if (debug == 2) {
215       Serial.println("YOU FOUND WALDO");
216     }
```

```
217     //Check which switch is activated
218     if (leftBottomSwitch)
219     {
220       //Check new direction of given new pos
221       if (dirZLeft(newPos.zLeft) == LEFT_DIR_UP)
222       {
223         safe = true;
224       }
225     }
226
227     //Check which switch is activated
228     if (rightBottomSwitch)
229     {
230       //Check new direction of given new pos
231       if (dirZRight(newPos.zRight) == RIGHT_DIR_UP)
232       {
233         safe = true;
234       }
235     }
236
237     //Check which switch is activated
238     if (leftTopSwitch)
239     {
240       //Check new direction of given new pos
241       if (dirZLeft(newPos.zLeft) == LEFT_DIR_DOWN)
242       {
243         safe = true;
244       }
245     }
246
247     //Check which switch is activated
248     if (rightTopSwitch)
249     {
250       //Check new direction of given new pos
251       if (dirZRight(newPos.zRight) == RIGHT_DIR_DOWN)
252       {
253         safe = true;
254       }
255     }
256
257
258     return safe;
259
260   }
261
262
263
264
265
266
267
268   /**
269      Toggle the bool given to this function
270   */
271   boolean toggleBool(boolean toggleThis)
272   {
273
274     //Check if its high
275     if (toggleThis)
276     {
277       toggleThis = false;
278     }
279     else if (!toggleThis)
280     {
281       toggleThis = true;
282     }
283
284     return toggleThis;
285   }
286
287
288
289
```

```
290
291
292
293
294
295
296
297
298
299   /*if (Y_HOME_SWITCH == triggedSwitch)
300     {
301
302     }
303     else if (Y_AWAY_SWITCH == triggedSwitch)
304     {
305
306     }
307     else if (X_HOME_SWITCH == triggedSwitch)
308     {
309
310     }
311     else if (Y_AWAY_SWITCH == triggedSwitch)
312     {
313
314     }
315   */
316
317
```

```
/* Controlls the vacuum pump */
// Turns on vacumm pump
void turnVacuumOn() {
  digitalWrite(PUMP, HIGH);
}
// Turns off vacumm pump
void turnVacuumOff() {
  digitalWrite(PUMP, LOW);
}

/**
 * Turn the vacuum on off, with the given intervall by parameter
 */
void turnVacuumOnOff(int delayTime) {

  digitalWrite(PUMP, HIGH);
  delay(delayTime);
  digitalWrite(PUMP, LOW);
}

```

```
/**
  This function checks the Serial input buffer.

  This function does the parsing and saves the parsed data. Sets the appropriate
  MAIN COMMAND to do the task which it recieved.

  If message:
  Parses the data, checks first for device address.
  If device addres is found, check which command it has recieved, and parse the
  attached payload if there is any.
*/

void receiveSerialEvent() {
  if (debug) {
    Serial.println("RecieveEvent(W)");
  }
  //Check if wire is ready to be read from/incomming bits
  if (Serial.available() > 0) {
    if (Serial.find(DEVICE_NAME)) {
      if (debug) {
        Serial.println("This massage is for me!!");
      }

      // Response array made for returning calib param
      // The calib param contains in state,  maxPos.zLeft and (GONEmaxPos.zRigth) +
      the pos of each tray found. (Left)
      // Therefor we need an array of 1(instate) +  1(maxPos) + (numberOftrays)
      int arrayLength = (2 + (trayReg.nrOfTrays ));
      int response[arrayLength];
      int cnt = 0;


      //-------------------------------------------------------------------------------
      --------------------
      // Reads incomming command
      int inCommand = Serial.parseInt(); // Secound Byte represents an command.
      // Updates the commands
      recieveCommand = inCommand;

      //Switch command to perform the tasks given by address
      switch (recieveCommand)
      {
        //-----------------------------------------------------------------
        //Command for moving the robot
        //TODO: READ INCOMMING BYTES AND STORE THEM IN DESIRED VARIABLES
        case moveRobot:
          if (debug) {
            Serial.println("*****DO MOVEROBOT RECOGNIZED*****");
          }
         // sendInt(ACK); // Send an Acknowledge
          /***Incomming values parsing****/
          //Set the new incomming positions
          newPos.zLeft = MMToSteps(Serial.parseFloat());
          newPos.zRight = -newPos.zLeft;

          if (debug) {
            //Print the values
            Serial.print("Z left value ");
            Serial.println(newPos.zLeft);
            Serial.print("Z rigth value ");
            Serial.println(newPos.zRight);
          }
          movingState = normalMove;
          // Main case to move robot
          mainCommand = recieveCommand;
          break;




        //-----------------------------------------------------------------------
        //Set the command for calibrating the robot.
        case doCalibrate:
```

```
69              if (debug) {
70                Serial.println("*****DO CALIBRATE RECOGNIZED*****");
71              }
72              //sendInt(ACK); // Send an Acknowledge
73              mainCommand = recieveCommand;
74              break;
75            //-----------------
76
77
78            //-------------------------------------------------------------------------
79            //Set the command for calibrating the robot.
80            case stopRobot:
81              if (debug) {
82                Serial.println("*****Stop ROBOT recognized****");
83              }
84              //sendInt(ACK); // Send an Acknowledge
85              mainCommand = recieveCommand;
86              break;
87
88          //-------------------------------------------------------------------------
89              case doSuction:
90              if (debug) {
91                Serial.println("*****DO SUCTION NO*****");
92              }
93              mainCommand = recieveCommand;
94              break;
95
96
97
98            //-------------------------------------------------------------------------
99            //Changing velocity command receved
100           case changeVelocity:
101             if (debug) {
102               Serial.println("***** CANGE VELOCETY  *****");
103             }
104             //sendInt(ACK); // Send an Acknowledge
105
106             // Uppdate the step speed
107             stepSpeed = Serial.parseInt();
108
109             if (debug) {
110               Serial.print("Z velocety: ");
111               Serial.println(stepSpeed);
112             }
113             break;
114
115
116           //-------------------------------------------------------------------------
117           //Set the command for calibrating the robot.
118           case changeAcceleration:
119             if (debug) {
120               Serial.println("*****CHANGE ACCELERATION RECOGNIZED*****");
121             }
122           //  sendInt(ACK); // Send an Acknowledge
123             stepSpeed = Serial.parseInt();
124             break;
125           //-----------------
126
127
128
129
130
131           /**** REQUEST EVENTS ****/
132           //-------------------------------------------------------------------------
133           // Retrive state command recived.
134           case retrieveState:
135             if (debug) {
136               Serial.println("******RETRIVE STATE*******: ");
137             }
138            // sendInt(ACK); // Send an Acknowledge
139             sendInt(inState);
140             break;
141
```

```
142   //------------------------------------------------------------------------
143         case calibParam:
144           if (debug) {
145             Serial.println("CalibParam recognized        --");
146           }
147          // sendInt(ACK); // Send an Acknowledge
148
149           //Add the values
150           response[cnt++] = parameter;
151           response[cnt++] = stepsToMM(maxPos.zLeft);
152           // response[cnt++] = stepsToMM(maxPos.zRight);
153
154           for (int i = 0; i < trayReg.nrOfTrays; i++) {
155             Tray tray = trayReg.trayList[i];
156             Pos pos = tray.zPos;
157             // Stor the tray positions in the response
158             response[cnt++] = stepsToMM(pos.zLeft);
159             // response[cnt++] = stepsToMM(pos.zRight);
160           }
161           sendIntegers(response, arrayLength);
162           // If debug print responce
163           if (debug) {
164             for (int i = 0; i < arrayLength; i++) {
165               Serial.print("Response: " );
166               Serial.println(response[i]);
167             }
168           }
169           break;
170
171   //------------------------------------------------------------------------
172         default:
173           if (debug) {
174             Serial.println("Recieve command not recognized");
175           }
176          // sendInt(NACK); // Send an Acknowledge
177
178       }
179
180
      //------------------------------------------------------------------------
      --------------------
181
182     } else {
183       //Remove the serial buffer
184       Serial.flush();
185     }
186    }
187  }
188
189
```

```
/***** Functions responseble for sending data over an serial communication. ****/

// Sends one byte over and serial communication using string.
// Uses devise name in front and devide the byte witn an , as delimiter
// It writes to the serial bus.


//Send one int to the serial port
void sendInt(int intToSend) {
  //Make the string, with device ID, int and ending
  String stringToSend = DEVICE_NAME;
  stringToSend = stringToSend + ",";
  stringToSend = stringToSend + intToSend;
  stringToSend = stringToSend + "\n";

  // Make char pointer and dynamically allocate the memory for all the chars the
pointer have access to
  char* cString = (char*) malloc(sizeof(char) * (stringToSend.length() + 1));
  stringToSend.toCharArray(cString, stringToSend.length() + 1);

  //Send the chars to the serial port
 Serial.write(cString);
  // Free the allocated memory.
  free(cString);

  if (debug) {
    Serial.println();
  }
}



// Sends multiple integers to the serial port.
void sendIntegers(int integers[], int arrayLength) {

  String stringToSend = DEVICE_NAME;
  //Iterate the integer array and add them to a String.
  for (int i = 0; i <  arrayLength; i++) {
    stringToSend = stringToSend + ",";
    stringToSend = stringToSend + integers[i];
  }
  //Set END to string
  stringToSend = stringToSend + "\n";



  //Make char string from arduino string by memory allocating the array.
  //Give the char pointer the memory for all the chars it should have control over
  char* cString = (char*) malloc(sizeof(char) * (stringToSend.length() + 1));
  //Use toChar function to split all the char's in the string. They will be
```

```
allocated to the memory the pointer has control of.
  stringToSend.toCharArray(cString, stringToSend.length() + 1);

  //Send all the chars to the serial port
  Serial.write(cString);

  // Free the allocated memory.
  free(cString);

  if (debug) {
    Serial.println();
  }
}
```

Appendix J

Matlab code

This is a image processing developed for the RoeBot project.

The processing takes a image from a tray in a hatchery, and searches it for dead roe.

```
function [ roeCoordinates ] = locateRoe(trayImage)

    % display original image
    figure(1)
    imshow(trayImage);
    title("original image");
    impixelinfo();

    % convert rgb image to gray-scale image
    gray = rgb2gray(trayImage);


    % display gray-scale image
    figure(2)
    imshow(gray);
    impixelinfo();
    title("rgb to gray-scale image");


    % binarize the gray-scale image with adaptive thresholding
    bw =
imbinarize(gray,'adaptive', 'ForegroundPolarity','bright', 'Sensitivity',
0.1);


    % display the binarized image
    figure(3)
    imshow(bw);
    title("adaptive binarized gray-scale image");


    % create a structure element for using in eroding and dilating.
    % creates a circular structure element, as the roe is circular.
    % makes the objects after erosion and dilating stay circular.
    SE = strel('disk',6);

    % erode binary image for for removing artifacts such as light
glare
    eroded = imerode(bw, SE, 'same');


    % display eroded image
    figure(4)
    imshow(eroded);
    title("eroded image");
```

```
    % grow the dilated objects back to the same size as before
 eroding.
    % uses the same structure element as for erosion.
    dilated = imdilate(eroded, SE, 'same');

    % display dialted image
    figure(5)
    imshow(dilated);
    title("dilated image");
    impixelinfo();


    % lable all objects in the dilated image for being able to extract
    % properties from them
    labeledbw = bwlabel(dilated);

    % extract the position of the labeled objects centroids
    stats = regionprops('table',labeledbw,'Centroid');

    % save the coordinates of the located objects centroids to the
 varible
    % returned from the function
    roeCoordinates = stats.Centroid;
end
```

Published with MATLAB® R2017b