

FMH606 Master's Thesis 2023  
Industrial IT and Automation

# **Colour Quality Monitoring System for Plastic Pipes Exterior Walls Colour with Machine Learning and Image Processing**

Hans-Christian Ringstad

Faculty of Technology, Natural Sciences and Maritime Sciences  
Campus Porsgrunn



**Course:** FMH606 Master's Thesis 2023  
**Title:** *Colour Quality Monitoring System for Plastic Pipes Exterior Walls Colour with Machine Learning and Image Processing*  
**Pages:** 89  
**Keywords:** *Plastic pipes, quality control, machine learning, CIELAB, CIE  $L^*a^*b^*$ ,  $L^*a^*b^*$ , Lab, Raspberry Pi, Picamera, python, scikit-learn, scikit-image*  
**Student:** *Hans-Christian Ringstad*  
**Supervisor:** *Ru Yan*  
**External partner:** *Daniel Zwick, Pipelife Norge AS*

### Summary:

Pipelife Norge AS is working on automating quality control for their pipes, specifically monitoring the exterior wall colour. Currently, operators manually inspect the pipes and rely on their experience to judge the colour.

When a colour monitoring system take a measurement of the colour they are usually measuring and comparing the colour against a standard, but at the factory a mathematical standard does not exist.

To modernize and streamline this process, a colour quality camera station has been constructed in this study. This station consists of a lightproof metal frame through which newly produced pipes can travel. The station is equipped with LED's and web cameras controlled by a Raspberry Pi 4 B, designed to capture the colour defects of each pipe in a more objective and consistent manner than human operators

Approximate 1100 images of brown and red PVC 110Ø pipes were collected to the dataset. The histogram used for the evaluation is based on the  $L^*$ ,  $C_{ab}^*$  and  $h_{ab}$  from the CIELAB colour space.

A set of machine learning models are developed using a voting ensemble method to evaluate the pipes. The model's performance was largely satisfactory, producing reliable results with most images. However, some challenges were observed with images that even a human eye would struggle to discern accurately. This indicates that while the system has made substantial strides in automating colour quality control, there are still areas for improvement and fine-tuning.

*The University of South-Eastern Norway accepts no responsibility for the results and conclusions presented in this report.*



# Preface

This thesis was written by Hans-Christian Ringstad, a student of the industry master version of the Master of Science Industrial IT and Automation study at the University of South-Eastern Norway, in collaborations with Pipelife Norge AS.

I would like to thank Ru Yan for supervising and her guidance in this project. Daniel Zwick and Marius Nordheim with their assistance with the project. And Nils Skei and the rest of the maintenance department for building the frame of the station and installing the equipment.

Porsgrunn, 15th May 2023

Hans-Christian Ringstad



# Contents

- Preface** **5**
  
- Contents** **8**
  - List of Figures . . . . . 10
  - List of Tables . . . . . 11
  
- 1 Introduction** **15**
  - 1.1 Background . . . . . 15
  - 1.2 Previous Proof-of-Concept . . . . . 16
  - 1.3 Problem Description . . . . . 18
  
- 2 Theory** **19**
  - 2.1 Light Reflection . . . . . 19
  - 2.2 Colour Analysis . . . . . 21
  - 2.3 Machine Learning Algorithms . . . . . 23
    - 2.3.1 SVM . . . . . 23
    - 2.3.2 Softmax Regression . . . . . 23
    - 2.3.3 Decision Tree . . . . . 24
    - 2.3.4 Random Forest . . . . . 24
    - 2.3.5 Voting Ensemble . . . . . 24
  
- 3 Methods** **25**
  - 3.1 Hardware . . . . . 25
    - 3.1.1 Material . . . . . 25
    - 3.1.2 Design of the Station . . . . . 26
    - 3.1.3 The Frame of the Station . . . . . 29
  - 3.2 Software . . . . . 32
    - 3.2.1 Design of the Software . . . . . 32
    - 3.2.2 Collect Samples . . . . . 38
    - 3.2.3 Image Processing . . . . . 40
    - 3.2.4 Development of Machine Learning Models . . . . . 42
  
- 4 Result** **51**
  - 4.1 Finished Design of the Station . . . . . 51
  - 4.2 Software . . . . . 52

4.3	Finalization of Machine Learning Models . . . . .	55
<b>5</b>	<b>Discussion</b>	<b>57</b>
5.1	Future Improvements and Use Cases . . . . .	59
<b>6</b>	<b>Conclusion</b>	<b>63</b>
	<b>Bibliography</b>	<b>65</b>
<b>A</b>	<b>Master Thesis Problem Description</b>	<b>67</b>
<b>B</b>	<b>Gantt Chart</b>	<b>71</b>
<b>C</b>	<b>Draft of the Station</b>	<b>75</b>
<b>D</b>	<b>Line Speed on EO1</b>	<b>79</b>
<b>E</b>	<b>Experiment Log of the Machine Learning Development</b>	<b>81</b>
<b>F</b>	<b>Image Set</b>	<b>83</b>
<b>G</b>	<b>Circle Reflection Simulator</b>	<b>85</b>
<b>H</b>	<b>Source Code</b>	<b>87</b>
<b>I</b>	<b>Work Folder</b>	<b>89</b>



# List of Figures

- 1.1 Simple figure of a extrusion line used to make plastic pipes. . . . . 15
- 1.2 The proof-of-concept used to test the feasibility of inspecting . . . . . 17
- 1.3 An example image captured by the previous proof-of-concept station. . . 17
- 1.4 The test conducted on the image of the pipe shown in figure 1.3. . . . . 18
  
- 2.1 Geometric relationships between the angles used when calculating the new directional angle when the light vector shall be reflected off the pipe  $\angle L^*$ . . 21
  
- 3.1 Drawing shows the coverage of the cameras on the pipe where the marked area is the area to be inspected and the green dotted lines are the . . . . . 27
- 3.2 The calculated reflections on the pipe where the sign of the direction for the light were set wrong. . . . . 28
- 3.3 The calculated reflections on the pipe, N is the total amount of lines, n is the amount of lines per light source and nl is amount of lamps. . . . . 28
- 3.4 Images of the metal frame. . . . . 30
- 3.5 Images of the cameras. . . . . 30
- 3.6 The LED strip was installed into the metal frame. . . . . 31
- 3.7 Images of the inside and the outside of the painted metal frame. . . . . 31
- 3.8 The base plate from the electrical cabinet with the wiring is done. . . . . 32
- 3.9 Use case model of the software used for the light box. . . . . 33
- 3.10 Domain model of the software used for the station. . . . . 34
- 3.11 The flowchart for the software used for the station. . . . . 35
- 3.12 The setup of the test of the system being tested showing the plate from figure 3.8 with the power connected and the cameras connect through a USB hub. . . . . 36
- 3.13 The ERD of the MES database. . . . . 37
- 3.14 The pipe sections used for generating datasets for training and evaluating machine learning models. . . . . 38
- 3.15 The GUI for the CQCS controller shows the LED controls and the image viewer. . . . . 39
- 3.16 The GUI for the CQCS controller showing the camera controls and the image viewer. . . . . 39
- 3.17 The original image from the good BROWN set compared with the blurred version of the same image. . . . . 40

3.18	The original image from the bad BROWN set compared with the blurred version of the same image. . . . .	40
3.19	The threshold at 0.875 and the blurred image from the good BROWN set with the threshold overlaid. . . . .	41
3.20	The threshold at 0.875 and the blurred image from the bad BROWN set with the threshold overlaid. . . . .	41
3.21	Here the image shown is the original image from the good BROWN set with the produced histogram of the $L^*$ , $a^*$ , $b^*$ , $C_{ab}^*$ , and $h_{ab}$ . . . . .	42
3.22	Here the image shown is the original image from the bad BROWN set with the produced histogram of the $L^*$ , $a^*$ , $b^*$ , $C_{ab}^*$ , and $h_{ab}$ . . . . .	42
3.23	Confusion matrix of experiment 1. . . . .	44
3.24	Confusion matrix of the errors of experiment 1. . . . .	44
3.25	Confusion matrix of experiment 7. . . . .	45
3.26	Confusion matrix of the errors experiment 7. . . . .	45
3.27	Confusion matrix of experiment 26. . . . .	46
3.28	Confusion matrix of the errors in experiment 26. . . . .	46
3.29	Confusion matrix of experiment 33. . . . .	47
3.30	Confusion matrix of the errors in experiment 33. . . . .	47
3.31	Confusion matrix of experiment 36. . . . .	49
3.32	Confusion matrix of the errors in experiment 36. . . . .	49
3.33	Confusion matrix of experiment 40. . . . .	50
3.34	Confusion matrix of the errors in experiment 40. . . . .	50
4.1	The calculated reflections on the pipe, N is the total amount of lines, n is the amount of lines per light source and nl is amount of lamps. . . . .	51
4.2	The front of the finished of the station. . . . .	52
4.3	The back of the finished station. . . . .	53
4.4	The display of the station showing brown pipes and a red pipe about to be placed in the station. . . . .	54
4.5	The display of the station showing red pipes. . . . .	54
4.6	Confusion matrix of the preliminary experiment 42. . . . .	56
4.7	Confusion matrix of the errors in the preliminary experiment 42. . . . .	56
5.1	Images captured by camera one showing the difference between normal colouring and the colouring after the blue tint has appeared. . . . .	57
5.2	Here the image shown is the original image from the good RED set with the produced histogram of the $L^*$ , $a^*$ , $b^*$ , $C_{ab}^*$ , and $h_{ab}$ . . . . .	58
5.3	Here the image shown is the original image from the good RED set with the produced histogram of the $L^*$ , $a^*$ , $b^*$ , $C_{ab}^*$ , and $h_{ab}$ . . . . .	58
5.4	The coverage of the of the pipe from the cameras if more stations are placed. . . . .	59

# List of Tables

- 3.1 Cross validation of the classifier from experiment 1. . . . . 44
- 3.2 Cross validation of the classifier from experiment 7. . . . . 45
- 3.3 Cross validation of the classifier from experiment 26. . . . . 46
- 3.4 Cross validation of the classifier from experiment 33. . . . . 47
- 3.5 Cross validation of the classifier from experiment 36. . . . . 49
- 3.6 Cross-validation of the classifier from experiment 40. . . . . 50
  
- 4.1 Cross validation of the classifier from experiment 42. . . . . 56
  
- C.1 Here the references and the description of the equipment in the found in  
the main electrical cabinet. . . . . 75



# Nomenclature

Symbol	Explanation
CQCS	Colour Quality Camera Station
ML	Machine Learning
GUI	Graphical User Interface
MES	Manufacturing execution systems
fps	Frames per second
ERD	Entity Relationship Diagram
SVM	Support Vector Machine
SVC	Support Vector Classification



# 1 Introduction

The world is becoming more and more automated. Automation removes humans from doing repetitive tasks. At Pipelife Norge AS the bundling of the pipes are being automated and part of this automation project is to automate the quality control. A part of the quality control will be to monitor the colour of the exterior wall of the pipes.

When a colour monitoring system take a measurement of the colour they are usually measuring and comparing the colour against a standard, but at the factory a mathematical colour standard does not exist. Today operators inspect the pipes manually and judges the colour based on their experience.

Here in this thesis I will make a colour quality camera station that will monitor and evaluate the colour of the exterior walls of plastic pipes.

## 1.1 Background

Pipelife Norge AS is one of the biggest producers of plastic pipes in Norway and is part of Pipelife International GmbH[1]. Today, plastic pipes are produced on production lines called extrusion lines, as illustrated in the figure 1.1.



Figure 1.1: Simple figure of a extrusion line used to make plastic pipes.

The production process starts with melting plastic pellets or granules made from recycled pipes and blending in a colourant to create a coloured liquid plastic. Then, the plastic liquid will be extruded and cooled to form a solid pipe. Next, the pipe will be cut into segments by a cutting unit, socketed at a pipe-socketing bench, and placed in a pipe cart.

In the end, operators will bundle the pipe cart onto a pallet. During the last step, the operators also will perform visual quality inspections of the pipe exterior colour before bundling the pipes.

However, with the implementation of a new automatic pipe bundling system in the factory, this visual quality inspection method has become unfeasible. Operators can no longer inspect the pipes' colours while the bundling machine is in operation, yet colour quality inspections remain necessary.

After consulting with the new bundle system's manufacturer, it was concluded that making a regular sensor-based quality check system for pipes exterior wall colour would be complicated and potentially unreliable. Consequently, applying computer vision (camera-based) and machine learning techniques to develop a colour quality inspection system was explored. A successful proof-of-concept for a camera-based solution was demonstrated using a single camera.

## 1.2 Previous Proof-of-Concept

A proof-of-concept was developed prior to this thesis. The proof-of-concept was based on a school assignment [2] in the course IIA1319 Software Engineering at USN where the assignment was decided by the student and here the assignment were to make a quality monitoring and inspection system for the pipe colour.

It was made to assess the feasibility of a system capable of inspecting the exterior walls of the produced pipes while they were being produced. The goal was to determine if it was possible to inspect these aspects by capturing an image of the pipe simultaneously.

Figure 1.2 shows the station used in the feasibility experiment. The cameras were positioned high enough to capture an image of the whole marking on the pipe. But at this distance, while also evaluating the pipe's color differentiation. The CIELAB color space was utilized to establish a hard upper and lower color limits, but due to the missing written colour standards at the factory, this method could not be directly applied to the production line..

Capturing an image of a pipe presents a significant challenge due to light source reflections on the pipe. This is the main reason for the requirement for the station to be as lightproof as possible. Even the ceiling light in the factory will cause unwanted reflections, leading to inaccuracies in the colouring measurements. In figure 1.3 an image captured under an experiment illustrates the noticeable reflection and resulting measurement noise.

In figure 1.4 a test to differentiate the colours using the CIELAB colour space is shown. The first image shows the original image to be differentiated for reference for the viewer. The marked areas on the other subfigures shows where the colour is outside of the limit





Figure 1.2: The proof-of-concept used to test the feasibility of inspecting



Figure 1.3: An example image captured by the previous proof-of-concept station.

shown in the title of the figure. The red markings means the value is above the upper limit, while the blue marking means they are below the lower limit. The  $L^*$ ,  $a^*$  and  $b^*$  subfigure has a grayscale image with the their markings overlaid. In the next subfigure the overlaid marking of the error,  $\Delta E^*$ , is seen where in the original image it was practically white. And

the last image shows the acceptable areas where the white colour is placed. Here as seen the reflection from the light source make the lines unusable for proper differentiation.

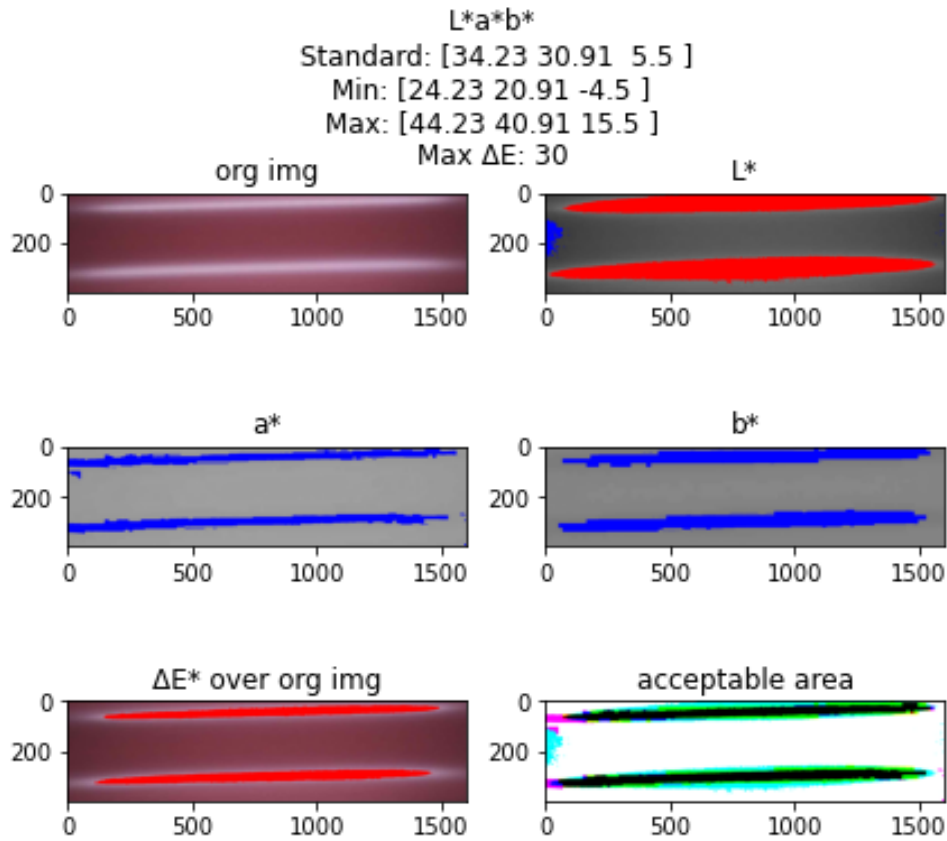


Figure 1.4: The test conducted on the image of the pipe shown in figure 1.3.

### 1.3 Problem Description

In this thesis a system will be made to inspect the colour of the exterior wall of a plastic pipes. The machine will be designed to collect a training and test set for the development of a machine learning algorithm, which will do the inspections of the colour. Part of the development will involve extracting any useful features from the dataset to facilitate the algorithm's functionality. Additionally, we will discuss potential further developments for the machine and explore other use cases within a factory setting, if applicable.

## 2 Theory

Prior to designing the station, it is necessary to calculate and simulate the reflections. Then some knowledge of the colour space which shall be used to find the difference in colour and lastly some words of the algorithm used.

When the the dataset shall be evaluated histograms will be used. Some statistical variables will be added to help with the evaluation. They will be the mean, the standard deviation and the variance. The equations 2.1 to 2.3 are from [3].

$$\text{Mean:} \quad \mu_X = E(X) = \sum_{\text{over all } x} xp(x) \quad (2.1)$$

$$\text{Standard deviation:} \quad \sigma_X^2 = E[(X - \mu_X)^2] = \sum_{\text{over all } x} (x - \mu_X)^2 p(x) \quad (2.2)$$

$$\text{Variance:} \quad \sigma_X = \sqrt{\sigma_X^2} \quad (2.3)$$

The histogram is treated as the stochastic variable  $X$  while the bins are the variable  $x$ .  $X$  is divided with  $N$ , the sum of the values in  $X$ , and is thereafter treated as a probability as  $p(x)$ .

### 2.1 Light Reflection

In this scenario, we have  $N$  light vectors being simulated, with  $n_l$  light sources, each generating  $n$  light vectors per light source. A light vector will always need a light source to originate from. Therefore, the total number of light vectors,  $N$ , is obtained by multiplying the amount of light sources,  $n_l$ , with the amount of light vectors per source,  $n$ .

The reference matrix,  $r_k$ , is used to represent the ...the position of the of the light vectors in the Cartesian plane and  $r_0$  is the known starting position vectors, Equation 2.4 shows the composition of  $r_k$ . The angle matrix  $\angle L$  represent the known directional angle of all the light vectors, and its composition is shown in equation 2.5. Equation 2.6 shows how one step of the light vectors is calculated.

The constant  $\delta_t$  is the step constant, determining the length of one vector step.

$$\text{Reference matrix: } r_k = \begin{bmatrix} r_{x,1,1,k} & r_{y,1,1,k} \\ r_{x,1,2,k} & r_{y,1,2,k} \\ \vdots & \vdots \\ r_{x,1,n,k} & r_{y,1,n,k} \\ r_{x,2,1,k} & r_{y,2,1,k} \\ r_{x,2,2,k} & r_{y,2,2,k} \\ \vdots & \vdots \\ r_{x,n_l,n,k} & r_{y,n_l,n,k} \end{bmatrix} \in \mathfrak{R}^{N \times 2} \quad (2.4)$$

$$\text{Angle matrix: } \angle L = \begin{bmatrix} 1_{n \times 1} \otimes \angle L_1 \\ 1_{n \times 1} \otimes \angle L_2 \\ \vdots \\ 1_{n \times 1} \otimes \angle L_{n_l} \end{bmatrix} \in \mathfrak{R}^{N \times 1} \quad (2.5)$$

$$\text{One step: } r_{k+1} = r_k + \delta_t [\cos(\angle L) \quad \sin(\angle L)] \quad (2.6)$$

To ensure the new angle for the light vector is correct the crossing point between the vector and the pipe wall is calculated. By combining the standard equation for a straight line,  $y = ax + b$ , representing the vector and a circle,  $r^2 = x^2 + y^2$ , representing the pipe wall the equation 2.7 is then derived. At the crossing point of the light vector and the pipe wall it is assumed the equation will be equal to 0. Then Newton's method is used to calculate the approximation of the point and the equations used are shown in equation 2.10 and 2.11. The  $a$  in equation 2.10 and 2.11 are used to control the convergence rate of the equations.

$$\text{Combined equation: } f = y^2 - y + x^2 + ax + b - r^2 \quad (2.7)$$

$$\text{Partial derivative of } f \text{ dependent of } x: \frac{\partial f}{\partial x} = 2x - a \quad (2.8)$$

$$\text{Partial derivative of } f \text{ dependent on } y: \frac{\partial f}{\partial y} = 2y - 1 \quad (2.9)$$

$$\text{Newtons Method for x coordinates: } x_{k+1} = x_k - a \cdot \frac{f}{\frac{\partial f}{\partial x}} \quad (2.10)$$

$$\text{Newtons Method for y coordinates: } y_{k+1} = y_k - a \cdot \frac{f}{\frac{\partial f}{\partial y}} \quad (2.11)$$

Afterwards, the new angle for the light vector angle is calculated. To get the angle  $\alpha$ , the input directional angle must be inputted into the equation 2.12 to adjust the incoming light vector angle. To get the angle  $\beta$  the crossing point coordinates are inputted into

the trigonometric function  $\arctan2$ . Then the difference between  $\alpha$  and  $\beta$  is calculated to yield the difference angle  $\gamma$ . Finally, the new light vector will be calculated by subtracting twice the difference times two.

The relationship between the angles is illustrated in figure 2.1.

$$\text{Incoming light vector angle: } \alpha = \angle L - \pi \quad (2.12)$$

$$\text{Crossing point angle: } \beta = \arctan2(y_{opt}, x_{opt}) \quad (2.13)$$

$$\text{Difference angle: } \gamma = \alpha - \beta \quad (2.14)$$

$$\text{New light vector angle: } \angle L^* = \alpha - 2\gamma \quad (2.15)$$

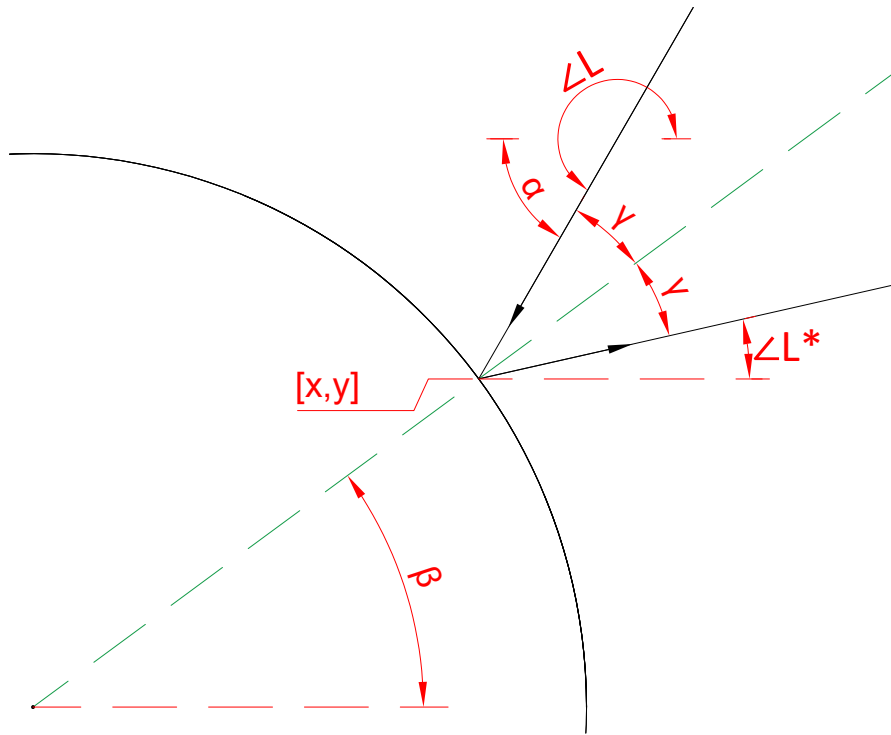


Figure 2.1: Geometric relationships between the angles used when calculating the new directional angle when the light vector shall be reflected off the pipe  $\angle L^*$ .

## 2.2 Colour Analysis

The most common computer images colour standard used is sRGB which has an 8-bit value range for each colour, red, green and blue, but these values are not ideal to be used to differentiate between colours. The CIELAB colour space was made to do differentiation

between colours. To calculate the  $L^*$ ,  $a^*$ ,  $b^*$  values the XYZ reference frame must be known. Equation 2.16 shows the transformation matrix from sRGB and to the CIE XYZ reference frame using illuminant D65 as recommended by CIE. Before using the equation the sRGB are linearized from the range  $[0,255]$  to  $[0,1]$ .

Equation 2.16 to 2.23 can be found in [4][5]. The scikit-images library[6] uses the equation below.

$$\text{sRGB to CIE XYZ: } \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{D65} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R_{Linear} \\ G_{Linear} \\ B_{Linear} \end{bmatrix} \quad (2.16)$$

The  $X_n$ ,  $Y_n$  and  $Z_n$  are the reference illuminant which are changed depending of what illuminants are used. The standard illuminant D65 is used here which means the reference illuminant will be  $X_n = 95.04$ ,  $Y_n = 100.0$ ,  $Z_n = 108.88$ .

The axes in the CIELAB colour space are defined where the  $L^*$  is the lightness, the  $a^*$  is the red-green opponency and  $b^*$  is the yellow-blue opponency. If  $a^*$  is a positive number the colour has a red part, but if it is negative it has a green part. If  $b^*$  is a positive number the colour has a yellow part, but if it is negative it has a blue part. The ranges of the values are  $L^* \in [0, 100]$ ,  $a^* \in [-128, 127]$ ,  $b^* \in [-128, 127]$ .

$$\text{CIELAB } L^*: \quad L^* = \begin{cases} 116 \left( \frac{Y}{Y_n} \right)^{1/3} - 16 & , \text{for } \left( \frac{Y}{Y_n} \right) > \left( \frac{6}{29} \right)^3 \\ \left( \frac{29}{3} \right)^3 \left( \frac{Y}{Y_n} \right) & , \text{for } \left( \frac{Y}{Y_n} \right) \leq \left( \frac{6}{29} \right)^3 \end{cases} \quad (2.17)$$

$$\text{CIELAB } a^*: \quad a^* = 500 \left[ \left( \frac{X}{X_n} \right)^{1/3} - \left( \frac{Y}{Y_n} \right)^{1/3} \right] \quad (2.18)$$

$$\text{CIELAB } b^*: \quad b^* = 200 \left[ \left( \frac{Y}{Y_n} \right)^{1/3} - \left( \frac{Z}{Z_n} \right)^{1/3} \right] \quad (2.19)$$

In the equation 2.20 and 2.21 the chroma and the hue angle is calculated. By comparing the CIELAB  $a^*$  and  $b^*$  to the CIELAB  $C_{ab}^*$  and  $h_{ab}$ . It can be interpreted that these values can be compared to transforming the Cartesian coordinates system to a cylindrical coordinates system.  $L^*$  is considered the height in both system.

$$\text{CIELAB chroma: } \quad C_{ab}^* = \sqrt{a^{*2} + b^{*2}} \quad (2.20)$$

$$\text{CIELAB hue angle: } \quad h_{ab} = \arctan \left( \frac{b^*}{a^*} \right) \quad (2.21)$$

The equation 2.22 and 2.23 is the difference equations for colour and hue difference. When a colour standard is known,  $(L_1^*, a_1^*, b_1^*)$ , then it would be possible calculate the difference with the measured colour,  $(L_2^*, a_2^*, b_2^*)$ .

$$\text{CIELAB colour difference: } \Delta E_{ab}^* = \sqrt{(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2} \quad (2.22)$$

$$\text{CIELAB hue difference: } \Delta H_{ab}^* = 2\sqrt{C_{ab,1}^* C_{ab,2}^*} \sin\left(\frac{h_{ab,1} - h_{ab,2}}{2}\right) \quad (2.23)$$

## 2.3 Machine Learning Algorithms

In this project, several machine learning algorithms were used, each with a brief description and relevant hyperparameters. All hyperparameters used are mentioned here. If a hyperparameter is not mentioned, it is either not in use or the default value is in use as stated in the documentation for the scikit-learn library[7].

### 2.3.1 SVM

Support Vector Machine, or SVM, is a machine learning algorithm that is an effective and versatile method to create a machine learning model. It is able to be used in both linear and nonlinear cases. It can use different kernel functions to predict a class, including both linear and polynomial functions. Here only a polynomial kernel function will be used. The hyperparameters for this algorithm will be the C, which is used by the L1 penalization to allow the control of how complex the model shall be, the d, the order of the polynomial, and r, the constant coefficient[8]. In the scikit-learn library the type of SVM used is called Support Vector Classification, or SVC[7].

### 2.3.2 Softmax Regression

Softmax regression, also known as multinomial logistic regression is a generalized version of logistic regression to support being used as a classifier. This classifier creates a score for each class. Then normalized exponential or softmax function is applied to the score and then returns a probability of every class. The hyperparameter C is used to regulate the allowed complexity of the model[7][8].

### **2.3.3 Decision Tree**

Decision Trees is a machine learning algorithm, which is a versatile method to create a machine learning model. It is cheap on computing power and it is possible to export the tree to see why it makes its decision. The tree can be seen as a collection of if-statements that are trained to find rules to fit a dataset. The hyperparameter for this method is the maximum allowed depth the tree is allowed to train to[7][8].

### **2.3.4 Random Forest**

Random Forest is an ensemble of decision trees and thereby why it is called a forest. This algorithm takes and train the data to subset of the dataset and returns an average probability of the classes. The hyperparameters for this method is the amount of trees and the maximum allowed depth the tree is allowed to train to[7][8].

### **2.3.5 Voting Ensemble**

Unlike the previously mentioned algorithms, the voting ensemble is not a single machine learning algorithm. Instead, it combines different algorithms into an ensemble and votes on what is the most probable correct prediction. This approach is useful when individual algorithms are not able to deliver good enough predictions[7][8].



## 3 Methods

In this chapter the methods to build the colour quality camera station is described. The first part will describe the hardware and the second part will describe the software.

### 3.1 Hardware

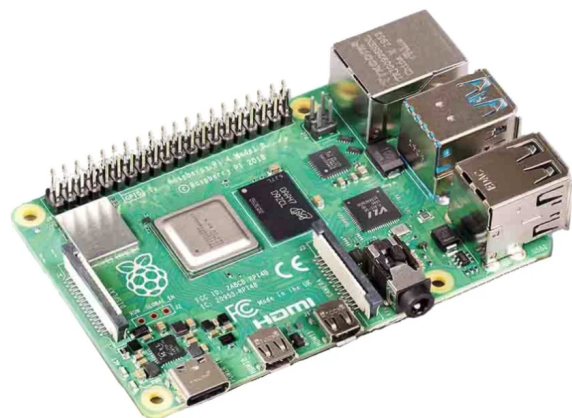
In this section the hardware for the station is described. The first part twill be to described materials who holds some significant importance to the result. Then the frame will be built and equipment installed.

#### 3.1.1 Material

The materials used for this project are of significant importance for achieving desired results. Here is a list of the materials:

##### Raspberry Pi

A Raspberry Pi is a microcomputer made by the Raspberry Pi Foundation and it is the computer to be used in this project. The board is able to use Python and its associated packages and other open-source packages, including scikit-learn which is used for the machine learning algorithm. It also has USB ports which can be used for web cameras and I/O pins that can be used for controlling relays. In this project a Raspberry Pi 4 B 8 ram is used[9].



## LED strips

A strip of LED lights will be used for the projects. It can be partitioned into smaller pieces every 10 cm or so. It has four types of LEDs with different colors, red, green, blue, and white. The each colour can be individually controlled by relays and dimmers[10].



## Camera

The cameras chosen for this project were the FIT0729 by DFRobot. The standard Pi Camera Module was considered but due to only one camera can be connected to each Raspberry Pi board and there being a shortage of boards a substitute was chosen. The FIT0729 is a USB web camera with similar specifications to the Pi Camera Module and could therefore be used[11].



## USB hub

A USB hub was used to make sure the cameras has enough power. This USB hub is able to handle the industrial environment of the factory. It has 4 USB ports and is powered by 24V DC input. With this hub, it is possible to power the cameras independently of the board[12].



### 3.1.2 Design of the Station

Before construction of the station can begin it must be designed. First the optimal position will be calculated, then a draft based on the optimal position will be drafted.

### Optimal Position:

When designing the station, the first step is to determine the optimal positions for the cameras and LED strips. Due to the reflections created by the LED strips on the pipes, as seen in the proof-of-concept, it will be assumed that it is not possible to get a  $360^\circ$  coverage from one station. Therefore, the goal is to obtain a minimum  $180^\circ$  coverage instead. Section 5.1 discusses methods for achieving greater coverage.

Figure 3.1 shows the coverage of one station with its four cameras. To get the desired  $180^\circ$  coverage of the pipe it is split into four sections of  $45^\circ$  which is the marked fields in the circle shown in the figure. The camera will be placed at a far enough distance from the pipe to get most of the pipe in its field of vision.

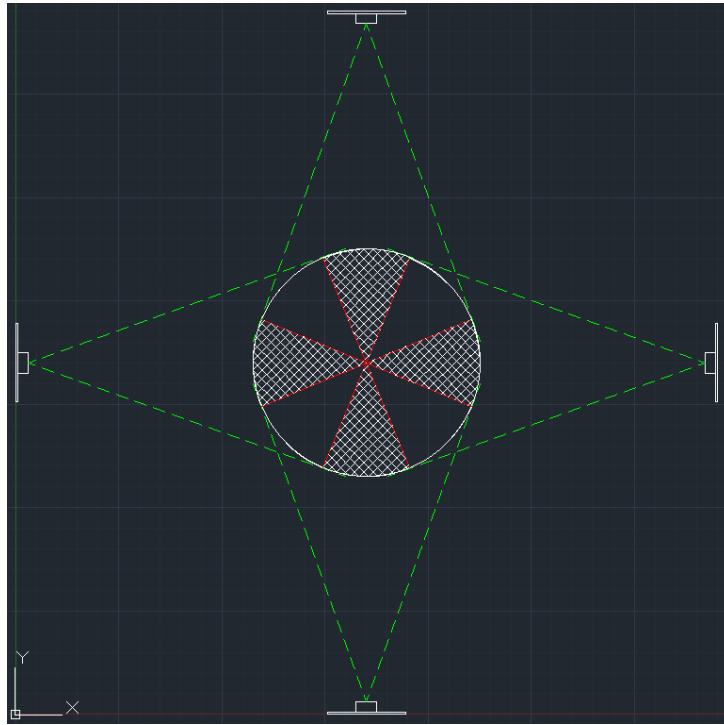


Figure 3.1: Drawing shows the coverage of the cameras on the pipe where the marked area is the area to be inspected and the green dotted lines are the .

Figure 3.3 shows the calculated light from the LED strip.

To find the optimal position the light from the lamps shall be simulated. The simulator, found in appendix G, is based on the equations given in section 2.1. First the reference values for  $r_0$  will be the coordinates for the light source where the light will be emitted from. The vectors will be evenly spaced between each other when emitted. The angle will be taken from the angle of the source as seen in figure 3.3. Then the vectors are simulated by iterating over equation 2.6. The constant  $\delta_t$  was set to 0.01.

For each iteration the distance of the vector is checked against the radius of the pipe. When it gets lesser then the radius newtons method is applied to get the crossing x- and y-coordinates as shown in equation 2.10 and 2.11.

Finally, a new directional angle is calculated according to equation 2.12 to 2.15. After it is calculated it will continue the iteration until it is done.

Figure 3.2 displays the initial versions of the software, where the reference points' angles were improperly aligned, and the light vectors' signs were incorrect.

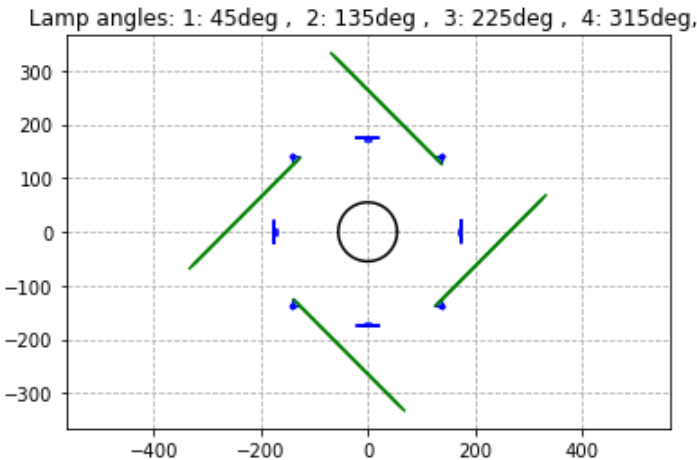


Figure 3.2: The calculated reflections on the pipe where the sign of the direction for the light were set wrong.

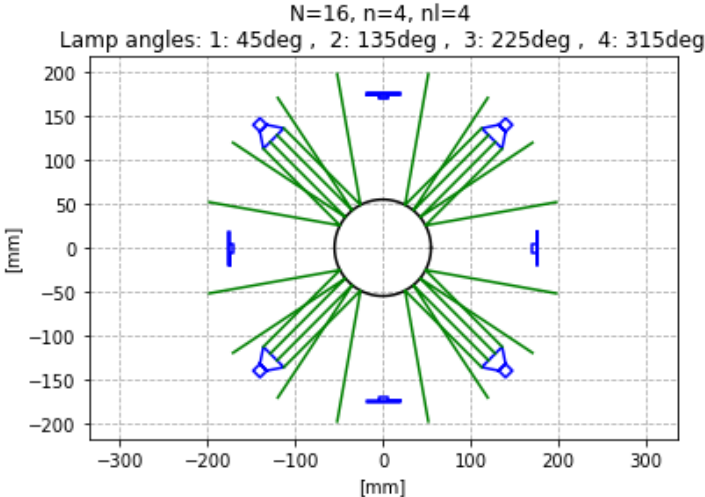


Figure 3.3: The calculated reflections on the pipe, N is the total amount of lines, n is the amount of lines per light source and nl is amount of lamps.

Figure 3.3 shows the light vectors simulated with the simulator found in appendix G. It shows it would be possible to put the cameras in the position shown in figure 3.1.

### **First Draft of the Station:**

After finding the theoretical light vectors the first draft of the design is drawn, as seen in appendix C. The station draft was made based on the calculated optimal position, with cameras set 115 mm from the outer pipe wall from the top, bottom and to the sides. The distance was measured to position the camera as close as possible while ensuring the entire pipe is within the cameras field of vision as seen in figure 3.1. The LED's were put in a 45° direction towards the pipe far enough from the pipe for a diffusion fabric to be fastened behind the cameras.

Then an arrangement diagram is made of the main electrical cabinet. The equipment used is also listed in appendix C. The schematic shows the placement of the components listed for the power supply for the Raspberry Pi, the USB hub, the LED's, and other components to control them.

Additionally, an electric schematic was also made illustrating the system's wiring and detailing how the relays are connected to the Raspberry Pi and how they are able to control the LED's.

### **3.1.3 The Frame of the Station**

Figure 3.4 to 3.8 depict the metal frame of the station. As shown, the design follows the drafts in appendix C. With the assistance from the maintenance department, the station was constructed. Some decisions about the construction were made by them regarding unspecified elements in the draft, such as the legs and the threaded rods. The legs are fastened with bolts pressing on the steal square bars legs as seen in figure 3.4a.

The thread rods are shown in figure 3.4b. They will be used to hold the cameras, LED strips, and the diffusion fabric. A rod was used so it would be possible to adjust the angle for cameras and LED's for when testing of the station begins.

Figure 3.5a shows the holder for the cameras with the installed cameras. The camera are fastened to a metal plate with a non-conductive material between the camera and the plate. The plate is welded to a cylinder, which is then inserted into the thread rods. An image of the cameras installed in the metal frame is shown in figure 3.5b.

The LED strips are fastened on the thread rods as shown in figure 3.6. They are secured to the threaded rods at the angle specified in appendix C. It uses four approximately 20 cm of the strip at each corner of the station.

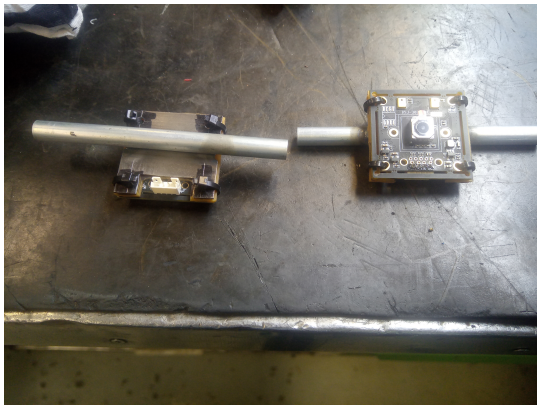


(a) The metal frame made from the draft in appendix C



(b) The inside of the metal frame showing the thread rods.

Figure 3.4: Images of the metal frame.



(a) The metal holder for the cameras with the fastening.



(b) The cameras are installed into the metal frame.

Figure 3.5: Images of the cameras.

And then, a  $20 \times 140$  cm diffusion fabric was inserted into the frame. It was put around the thread rods behind the cameras as seen in the draft. They are fastened with safety pins to keep it in place, as seen in figure 3.7a.

To prevent any foreign reflection a rubber ring is used around the pipe inlet hole. The ring will ensure that light from the outside of the station does not cause any noticeable

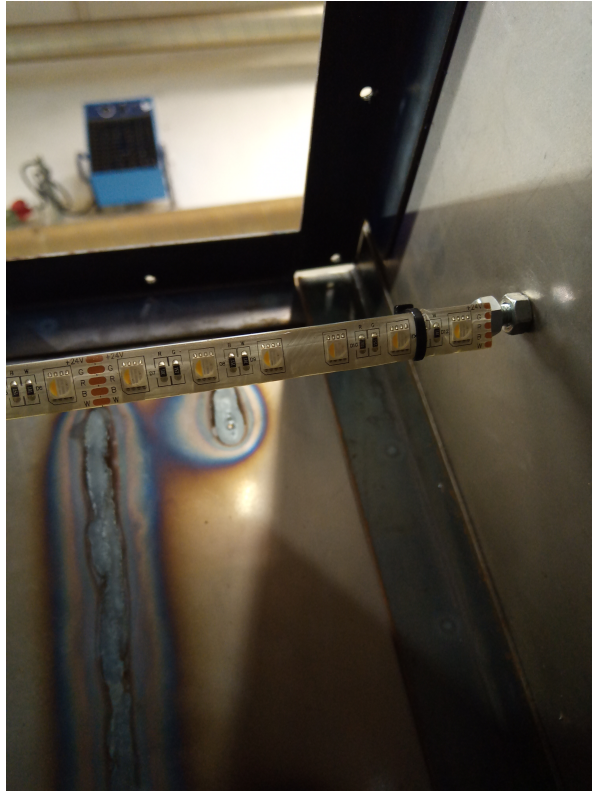
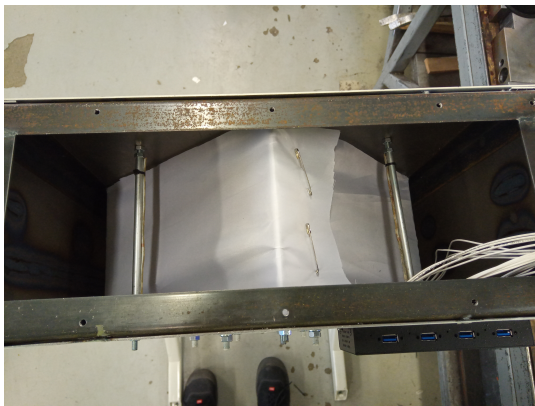
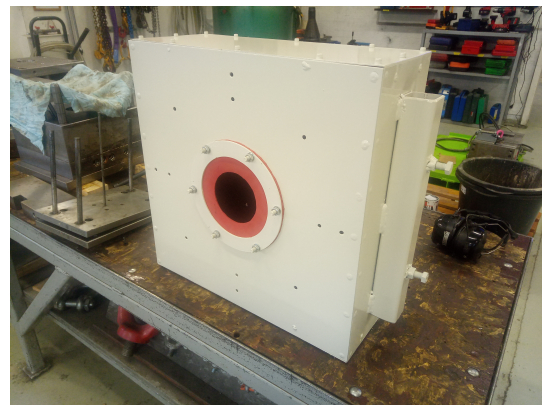


Figure 3.6: The LED strip was installed into the metal frame.



(a) The diffusion fabric installed in the metal frame.



(b) The painted metal frame showing the red rubber ring at the pipe inlet hole.

Figure 3.7: Images of the inside and the outside of the painted metal frame.

disturbances in the captured images. Figure 3.7b displays the completed, painted box and the rubber ring.

In figure 3.8 the equipment is mounted on the base plate for an electrical cabinet. It is following the arrangement diagram and the electrical schematic in appendix C.

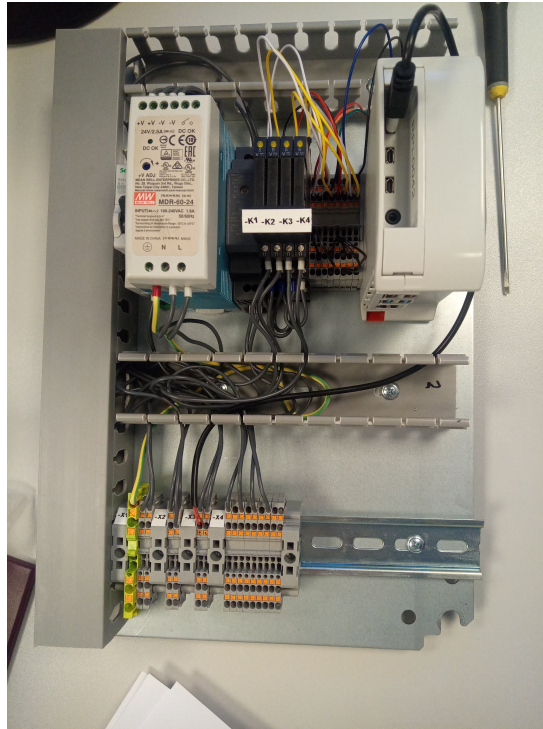


Figure 3.8: The base plate from the electrical cabinet with the wiring is done.

## 3.2 Software

Here the software for the station will be described. Two UML models, use case model and domain model, and a flowchart for the software is shown. Then describing how the light controls and camera control works. Then the setup of the database and its structure.

Afterwards a dataset must be collected to train the machine learning algorithms and how the dataset will be built.

Finally the setup of the machine learning algorithm and some notable experiments.

### 3.2.1 Design of the Software

Here the software will be designed. Two UML models, a use case model and a domain model, and a flowchart will be used as a base for the software. Then a description on how the light control and camera control will work and lastly the structure of the database.



### Use Case Model:

A use case model is developed to identify relevant use cases in this project. One use case involves controlling the LED strips within the station, allowing the operator to turn them on and off via a GUI or display.

Another use case involves displaying the captured images and their evaluation on the display/GUI.

Capturing images with the cameras at timed intervals is another use case, as is storing captured images in a local database.

The final use case is to evaluate the image using a machine learning (ML) model, and then storing the results in the local database, and sending the results to the MES database.

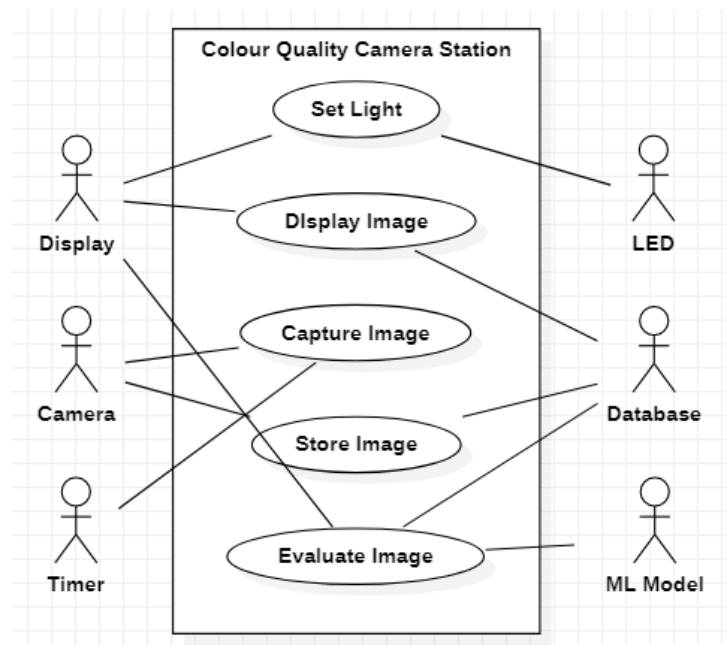


Figure 3.9: Use case model of the software used for the light box.

### Domain model:

A domain model for the software was also made, which is shown in 3.10. The main program will be the QualityCameraStation, it will control the cameras with the Timer telling when to capture an image.

The Display class is responsible for showing the last evaluation and the last image.

The Camera and LED class handles the communication to the relevant equipment. The Camera class communicates with the web cameras used to capture frames, while the LED class controls the colour emitted from the LED strips.

The MLModel class will be used to hold the classifier which will be used to evaluate and predict the colour in the images captured by the cameras.

The Database class connects to the local and MES database.

Lastly, is the Timer class and the ConfigDB class. The Timer class times the updating cycle for the QualityCameraStation class and Display class if needed. The ConfigDB class will store all configurable data and confidential data.

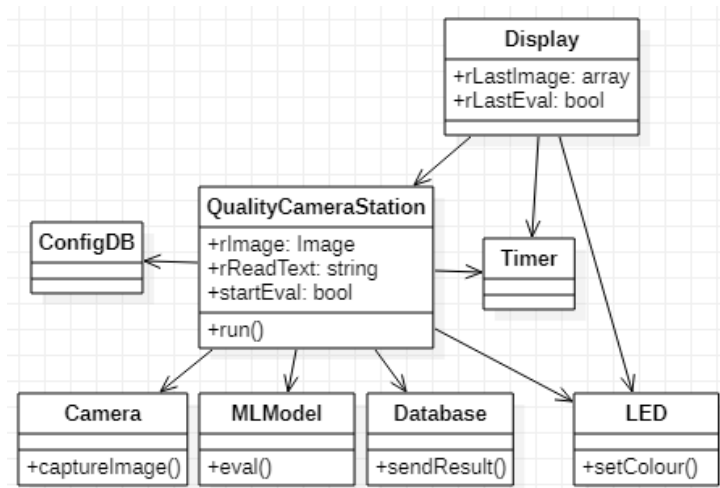


Figure 3.10: Domain model of the software used for the station.

### Flowchart:

A flowchart for the station software is presented in figure 3.11. It shows the flow that the software shall follow. The program initializes by setting up constants from the configuration, such as the video sources for the cameras and identifiers for the station.

Before the main loop starts, the LED's are set to ON or another setting specified in the configuration.

The main loop begins by capturing a frame from the cameras, processing the images, and inserting them into an evaluator algorithm. Lastly, after the processed image is evaluated the result will be sent to the local database and a MES database.

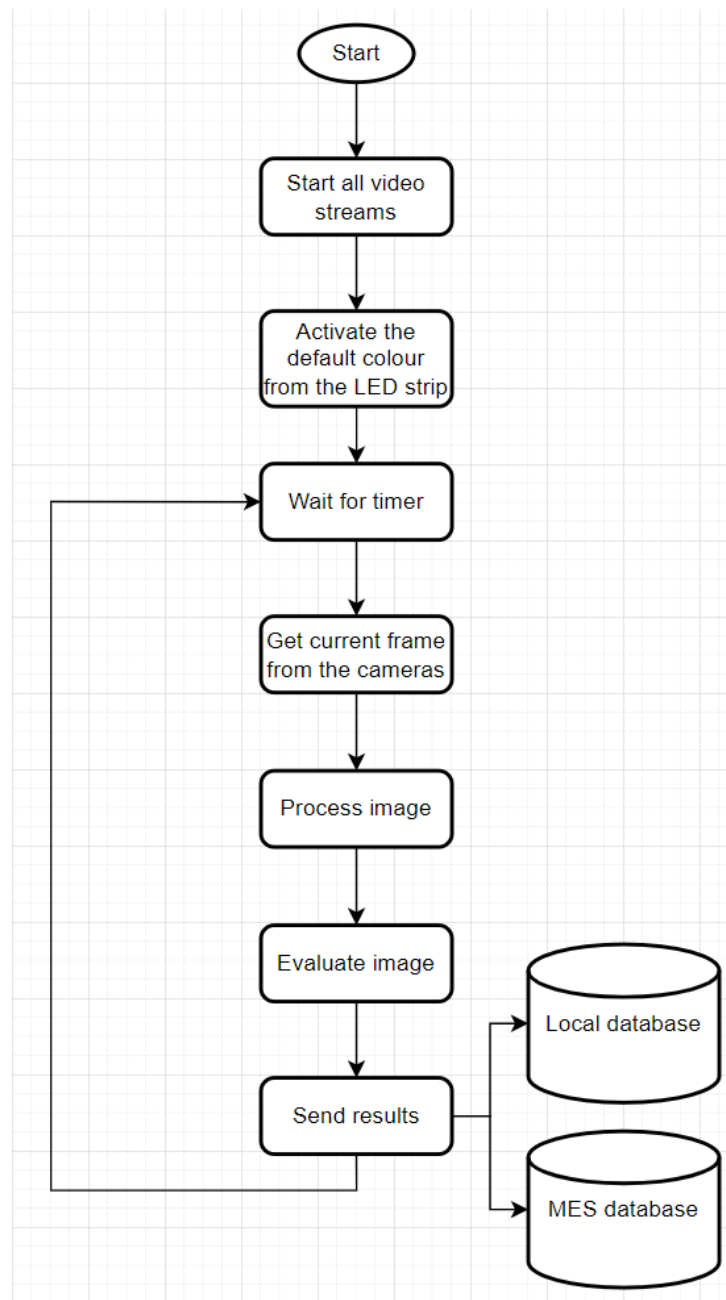


Figure 3.11: The flowchart for the software used for the station.

### Light and Camera Control:

Testing the cameras proved time-consuming, as improperly closed cameras often required a complete board reboot to free the video source.

The light control from the proof-of-concept was reused, simulating a switch-case statement

that does not exist in the Python 3.9 distribution. A class was made where the cases are functions in the class and a function called `switch()` will look for any attributes in a class by using the `getattr()` function. In each of the cases, the pins for the Raspberry Pi are set according to the case.

It was observed that using cameras in quick succession resulted in the next camera being unable to capture a single frame correctly. This issue is suspected to be caused by OpenCV creating background threads, which was resolved by implementing the `sleep()` function from the time library, allowing the main thread to wait for background threads to complete their tasks.

The logic for the camera was put into its own class, named `USBCamera`, which is used to communicate with the OpenCV-Python library. To use this class an object must be initialized first with the video source, resolution, and fps as parameters. Then use the `start()` function to start the camera in a new thread in the background to make sure the image is captured properly. Then a frame from the video stream is taken with the `getFrame()` function. Afterward, the camera can be stopped with the `stop()` function.



Figure 3.12: The setup of the test of the system being tested showing the plate from figure 3.8 with the power connected and the cameras connect through a USB hub.

In figure 3.12 the test setup is shown. The system is connected to power and the cameras are connected to the Raspberry Pi through a USB hub.

During testing , it was discovered that when the web cameras were used in rapid succession, they were unable to capture proper images. This issue is suspected to be caused by the Raspberry Pi is not able to draw enough power for all four cameras at once. Therefore a USB hub for extra power was added.

However, it was found that OpenCV could not adjust any camera parameters when using the USB hub, only passing on frames from the cameras. Although the frames from the cameras did return faster than before. Due to the power requirements, it was not possible to circumvent this problem. The frames returned with the resolution 1920x1080.

**Database:**

As this project will be used as a feedback sensor for a future colourant control system, the values obtained from the station must be stored in a database. The ERD for the database is shown in figure 3.13. Various tables were made to identify where the measurements were from. The identifiers with their own tables are the factory, the production line, the colour quality camera station, the cameras and the colours. The measurements table is the table to hold all measurements from the system. It will hold which camera the measurements came from, the expected colour and the timestamp to identify the individual measurements. Then the measurements used are the mean of  $L^*$ ,  $a^*$ ,  $b^*$  and the predicted evaluation from the machine learning algorithm.

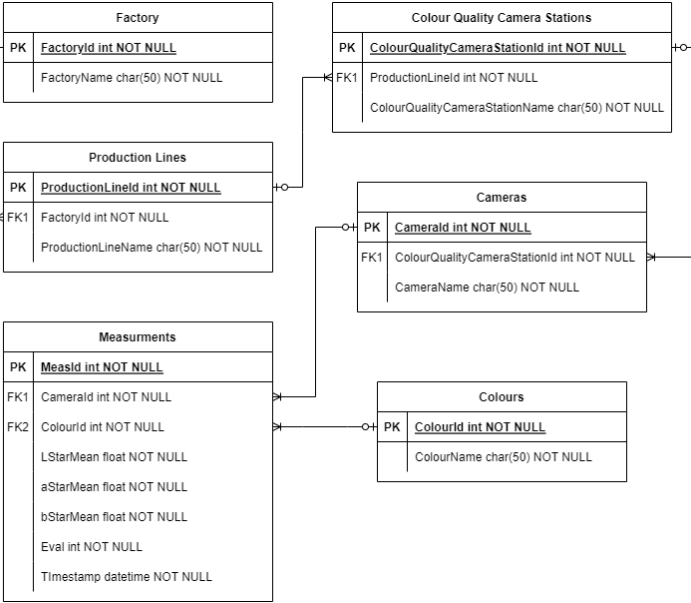


Figure 3.13: The ERD of the MES database.

The colours are stored in their own table to make sure any operator or programmer would not use an unknown colour in the table.

### 3.2.2 Collect Samples

Defective/bad pipes are not stored in storage at the Pipelife factory, as they are religiously granulated as fast as possible and reused for new pipes. To collect bad samples, new pipes were intentionally produced with insufficient colourant. The good pipes were taken from storage. The number of pipes used are 6 good and 7 bad brown pipes and 6 good and 5 bad red pipes. The pipes were 110mm in diameter and 1m in length, in figure 3.14 the pipes used are shown. The image set can be found in appendix F.



Figure 3.14: The pipe sections used for generating datasets for training and evaluating machine learning models.

A GUI was developed in PyQt5 to simplify the process of collecting sample images from the sample pipes. The interface developed in this study, is shown in figure 3.15 and figure 3.16. Figure 3.15 shows the default image if there are no images in the memory. The GUI is utilising the functions from the main program to capture the images. The GUI features a controller for adjusting the LED light settings in the box and offers different colour combinations, as shown in the LED controls in figure 3.15.

It will also have camera controls to capture images as shown in figure 3.16. It has a button to capture one image from all of the cameras and it also has some buttons to capture one individual image from one of the cameras. The images captured will be stored locally in a list with their generated filename. And lastly, it has a text field that will be used when image capturing is complete.

There is also an image viewer to display any of the images taken with the cameras. The oldest stored image is shown to the viewer for the user to decide on the desired action. The "Save image" button saves the image to the Raspberry Pi.

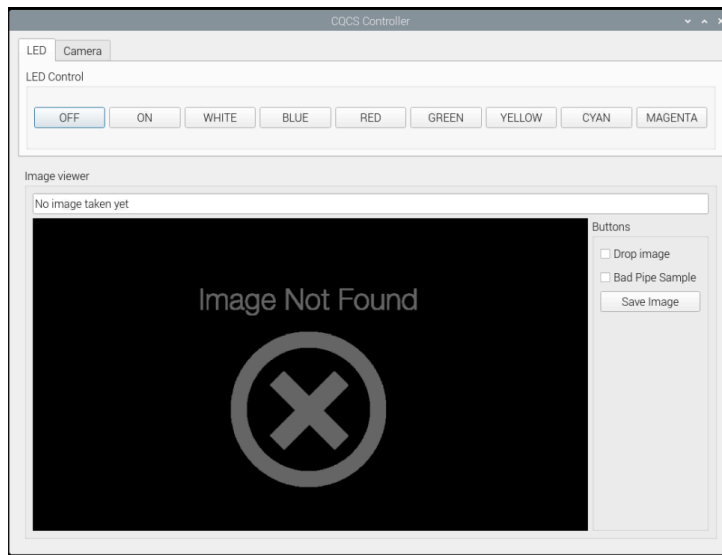


Figure 3.15: The GUI for the CQCS controller shows the LED controls and the image viewer.

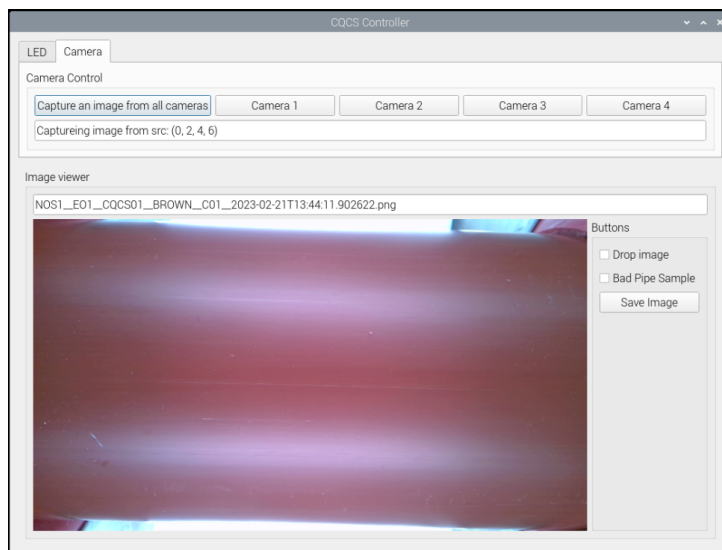


Figure 3.16: The GUI for the CQCS controller showing the camera controls and the image viewer.

Approximately 40 images per pipe were collected by moving and rotating the pipes through the station.

Labels chosen for the dataset were BROWN and RED for the good pipes and SCRAP for the bad pipes. The initial dataset is consistent with 170 BROWN samples, 316 RED samples and 467 SCRAP. After all the images were collected for the dataset, one of the good pipes for each of the colours had a marker mark the pipe to simulate a burn mark. 48 samples were captured and increased the SCRAP set to 515 samples. Later after experimenting with the machine learning algorithms, it was found the amount of samples

of BROWN was not enough for a stable algorithm so the amount was increased with 148 samples to a total of 318 samples.

### 3.2.3 Image Processing

The images must be processed to retain as much relevant information as possible and make it more accessible for the machine learning algorithm. The image is first blurred with the Gaussian filter from the scikit-image library[6] in its filter module. It is presumed the filter will remove any white noise in the image. Then the image is transformed into grayscale images so it is easier to get the threshold. A threshold is needed to minimize the noise in the images. The noise to be removed is the white diffusion fabric in the background of the images, as observed in figure 3.17 and 3.18.

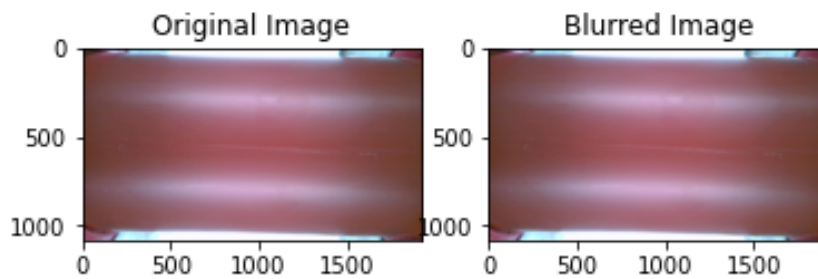


Figure 3.17: The original image from the good BROWN set compared with the blurred version of the same image.

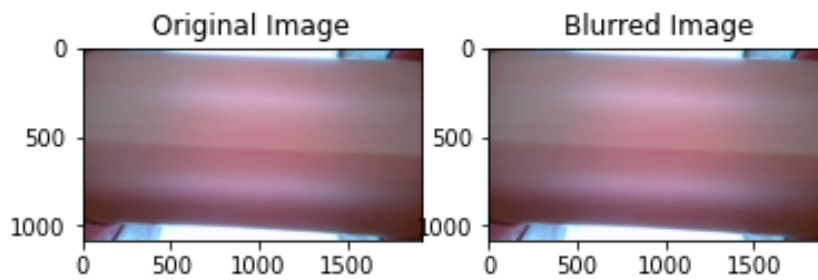


Figure 3.18: The original image from the bad BROWN set compared with the blurred version of the same image.

An automatic threshold was found to be around 0.57 when using the `threshold_otsu` function in the filter module. However, to exclude reflections from the light sources on the pipes, the threshold was adjusted to 0.875. Figure 3.19 and 3.20 shows the threshold overlaid on the blurred image.

To minimize the feature inputs into the machine learning algorithm, it was decided to use the CIELAB histogram instead of the images directly. If the image were to be used directly



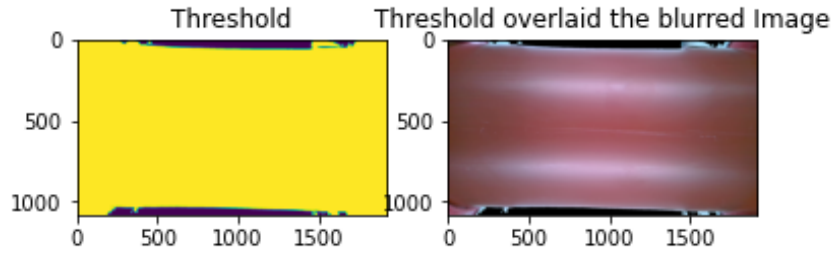


Figure 3.19: The threshold at 0.875 and the blurred image from the good BROWN set with the threshold overlaid.

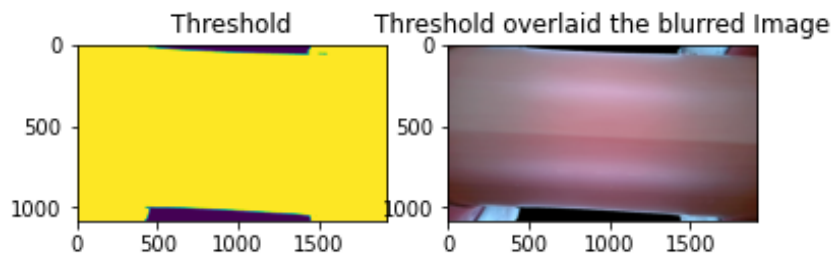


Figure 3.20: The threshold at 0.875 and the blurred image from the bad BROWN set with the threshold overlaid.

there would be 6220000 features ( $1080 \cdot 1920 \cdot 3 = 6220000$ ). But with the histogram, it should be less by several magnitudes. The input features shall be made up of some statistical variables based on the histograms from the values in the CIE LAB colour space and the histogram of  $L^*$ ,  $C_{ab}^*$ , and  $h_{ab}$ . After some tests the amount of bins in a histogram were decided to be  $2^{11} + 1$  or 2049. With this amount of bins it should be enough information for an algorithm and easier to remove noise from the image. When a histogram is made, the noise from the threshold has quite a large number compared to the rest of the histogram. The variables will be the mean, as calculated with equation 2.1, the standard deviation, as calculated with equation 2.2, and the variance, as calculated with equation 2.3. Then they are combined with the histogram of the lightness, chroma and hue angle. The  $a^*$  and  $b^*$  histograms are not used as they are used to calculate the chroma and the hue angle as seen in equation 2.20 and 2.21. The number of input features will then be 6162 ( $5 \cdot 3 + 2049 \cdot 3 = 6162$ ). The figures 3.21 and 3.22 show the histograms of the CIE LAB values of an image from the good BROWN set and the bad BROWN set. By comparing the two figures it is possible to see the difference between them which would mean it should be possible for a machine learning algorithm to distinguish them as well.

Before the histograms are combined, the position of the noise is located. An image with calculated noise is used for each processed image.

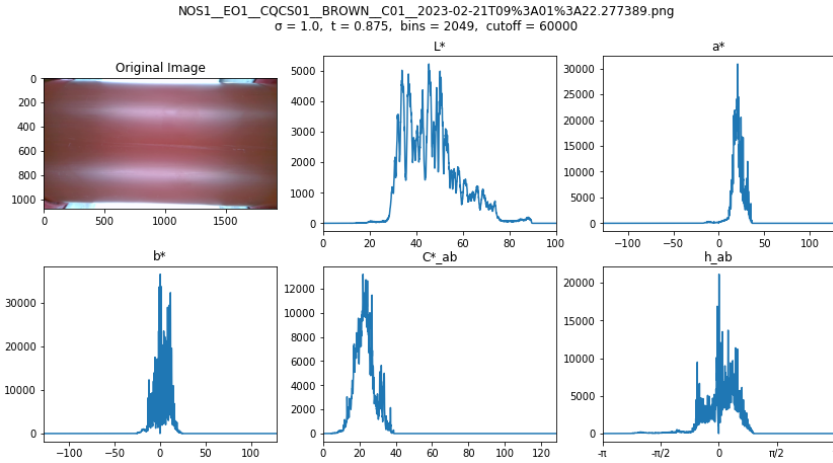


Figure 3.21: Here the image shown is the original image from the good BROWN set with the produced histogram of the  $L^*$ ,  $a^*$ ,  $b^*$ ,  $C_{ab}^*$ , and  $h_{ab}$ .

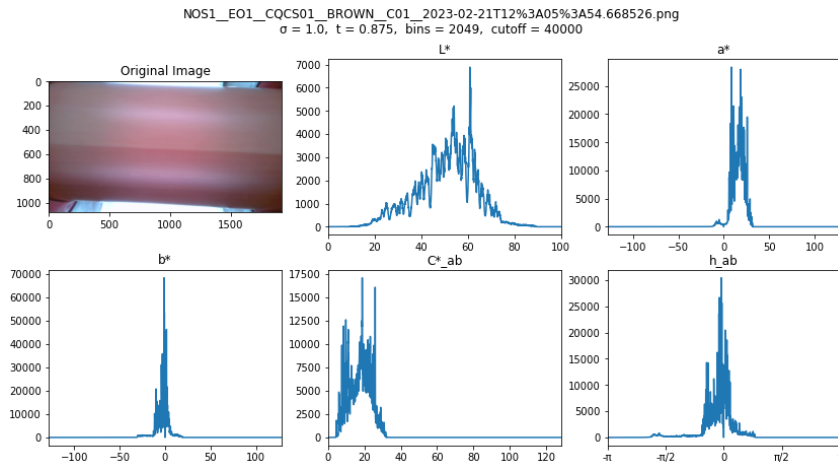


Figure 3.22: Here the image shown is the original image from the bad BROWN set with the produced histogram of the  $L^*$ ,  $a^*$ ,  $b^*$ ,  $C_{ab}^*$ , and  $h_{ab}$ .

### 3.2.4 Development of Machine Learning Models

After the image processing the images are incorporated into a dataset based on the method described in section 3.2.3. The first columns consist of statistical values found in the histograms for the CIELAB values  $L^*$ ,  $a^*$ ,  $b^*$ ,  $C_{ab}^*$ ,  $h_{ab}$ . The first 15 columns include the means, the standard deviation and the variance. The histogram for the lightness, chroma and the hue angle then will be added to the set. The red-green opponency and the yellow-blue opponency histograms will be disregarded to minimize the amount of features used for the algorithm, since the chroma and the hue is calculated from the opponency.

Various machine learning algorithms were tested. Appendix E shows the different exper-

iments done to make a satisfactory classifier. Each experiment documents the algorithm type, hyperparameters, dataset size, and the training and test set divisions. After each experiment, two confusion matrices are created: one showing the distribution of samples between the true and predicted labels, and the other showing the percentage of incorrect predictions.

Before deciding to make a machine learning algorithm the dataset are tested with the decision tree. In table 3.1 the cross validation of the classifier from experiment 1 is shown, figure 3.23 shows the confusion matrix of the preliminary experiment and figure 3.24 show the percent error of the experiment. This experiment were done to check if it would be possible to make a machine learning algorithm. And by comparing the cross validation and the confusion matrix it shows it would be possible to make an algorithm.

Here the the target features for the labels were set to 0 for SCRAP, 1 for BROWN and 2 for RED. For later experiment the target features were set to -1 for BROWN, 0 for SCRAP, 1 for RED. After the change the accuracy of the model were increased. And almost all experiment afterwards show 100% accuracy when differentiating RED and BROWN samples.

After the the first few experiments, it was found the algorithm were somewhat unstable and the BROWN sample had a high error rate compared to the RED sample. The set for the good pipes were then increased. Afterwards the tests did become more stable.

In experiment 7 the algorithm used is the decision tree. In table 3.2 the result of the cross validation is presented. Here the precision and the recall is about the same for each label with upto about 1.5% difference where the F1 score for SCRAP is 93.13%, BROWN is 93.47% and RED is 96.88%. Figure 3.25 and 3.26 shows the best found hyperparameter for the decision tree. Maximum depth of 5 were found to be giving the best results. Whenever a larger depth were used it would overfit and return a worse result.

In experiment 26 the algorithm used is softmax regression. In table 3.3, it presents the result of the cross validation from the softmax regression model. Here the colours have lower precision, but higher recall, so it is assumed it might have more false negatives than false positives. Figure 3.27 and 3.28 shows the best found hyperparameter for the regression. The optimized C were found to be 30. Other parameters used did not return as low error as other parameters that were tried. Other types of regressions algorithms were tested as well, but they were not able to handle classification.

In experiment 33 the algorithm used is random forest. In table 3.4 the result of the cross validation is presented. Like with experiment 26 here the colours have lower precision, but higher recall so it is assumed it might have more false negatives than false positives. Figure 3.29 and 3.30 shows the best found hyperparameter for the random forest. As with the decision tree the best depth were found to be also 5. The amount of trees used were the default value of 100.

Table 3.1: Cross validation of the classifier from experiment 1.

Labels	SCRAP	BROWN	RED
Precision	0.89519651	0.66507177	0.88922156
Recall	0.7961165	0.88922156	0.93987342
F1	0.84275437	0.73350923	0.91384615
Accuracy	0.83233533	0.89820359	0.85885886

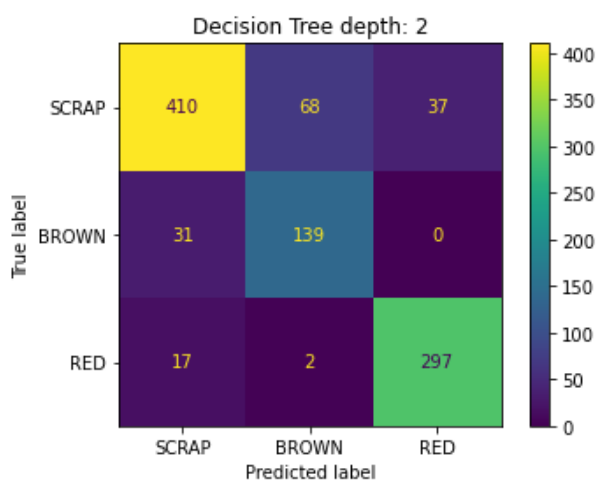


Figure 3.23: Confusion matrix of experiment 1.



Figure 3.24: Confusion matrix of the errors of experiment 1.

Table 3.2: Cross validation of the classifier from experiment 7.

Labels	BROWN	SCRAP	RED
Precision	0.92424242	0.93939394	0.96875
Recall	0.93846154	0.93	0.96875
F1	0.93129771	0.93467337	0.96875
Accuracy	0.90909091	0.92105263	0.97368421



Figure 3.25: Confusion matrix of experiment 7.



Figure 3.26: Confusion matrix of the errors experiment 7.

Table 3.3: Cross validation of the classifier from experiment 26.

Labels	BROWN	SCRAP	RED
Precision	0.88571429	0.95789474	0.90625
Recall	0.95384615	0.86666667	0.98305085
F1	0.91851852	0.91	0.94308943
Accuracy	0.92207792	0.88157895	0.96052632

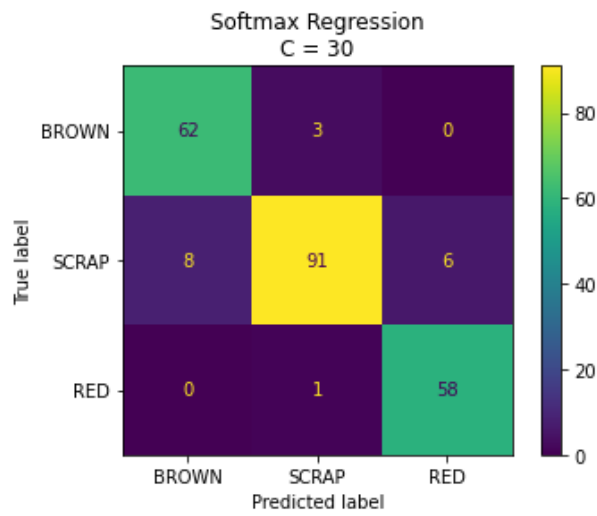


Figure 3.27: Confusion matrix of experiment 26.



Figure 3.28: Confusion matrix of the errors in experiment 26.

Table 3.4: Cross validation of the classifier from experiment 33.

Labels	BROWN	SCRAP	RED
Precision	0.88732394	0.96666667	0.91176471
Recall	0.95454545	0.86138614	1.
F1	0.91970803	0.91099476	0.95384615
Accuracy	0.88311688	0.94736842	0.94736842

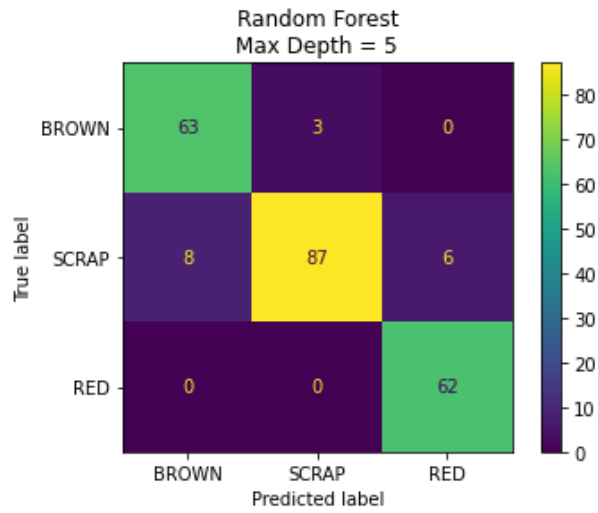


Figure 3.29: Confusion matrix of experiment 33.



Figure 3.30: Confusion matrix of the errors in experiment 33.

In experiment 36 the algorithm used is SVC. In table 3.5 the result of the cross validation is presented. Here only BROWN has lower precision and high recall where it is assumed it might have more false negatives than false positives. The RED has equal precision and recall. Figure 3.31 and 3.32 shows the best found hyperparameter for the SVC. The hyperparameters found were C equals to 1, the coefficient, r, is equal to 2 and the order of the polynomial, d, is equal to 3.

In experiment 40, the algorithm used is a voting ensemble. In table 3.6 the result of the cross-validation is presented. Here the BROWN pipes have lower precision but perfect recall. SCRAP here has high precision and lower recall so it is assumed it will have fewer false negatives than false positives. And RED has about equal precision and recall. Figure 3.33 and 3.34 shows the best-found hyperparameter for the Voting Ensemble. This experiment is using the previous algorithms with their found hyperparameters. It uses the weights 2.5 for the SVC, 1.2 for the decision tree, 2.5 for the random forest, and 1.8 for the Softmax regression and it uses the voting type hard voting.

Unfortunately, the classifier made in experiment 40 could not be used due to compatibility issues between the 64-bit and 32-bit versions of NumPy on the Raspberry Pi 4. Subsequent experiments are detailed in section 4.3.



Table 3.5: Cross validation of the classifier from experiment 36.

Labels	BROWN	SCRAP	RED
Precision	0.9375	0.96039604	0.953125
Recall	0.98360656	0.93269231	0.953125
F1	0.96	0.94634146	0.953125
Accuracy	0.97402597	0.97368421	0.90789474

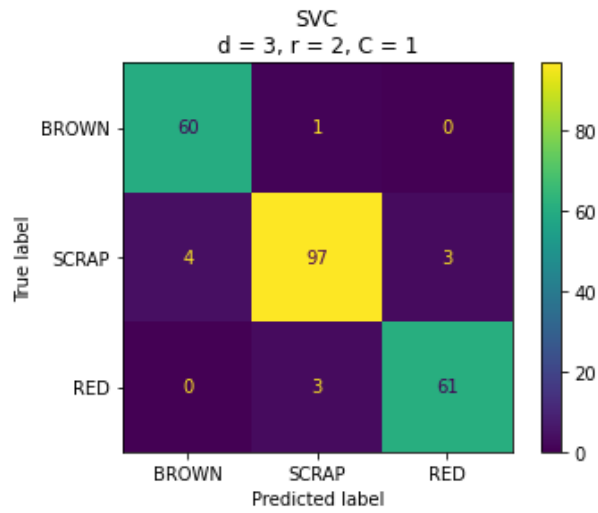


Figure 3.31: Confusion matrix of experiment 36.

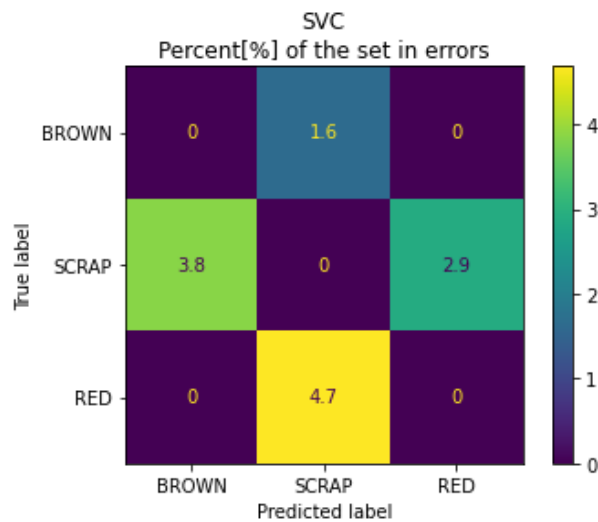


Figure 3.32: Confusion matrix of the errors in experiment 36.

Table 3.6: Cross-validation of the classifier from experiment 40.

Labels	BROWN	SCRAP	RED
Precision	0.91549296	0.9673913	0.96969697
Recall	1.	0.91752577	0.95522388
F1	0.95588235	0.94179894	0.96240602
Accuracy	0.94805195	0.97368421	0.92105263

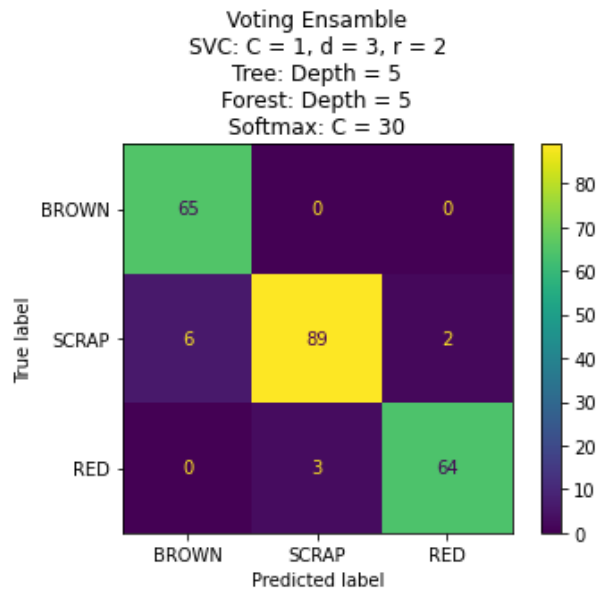


Figure 3.33: Confusion matrix of experiment 40.

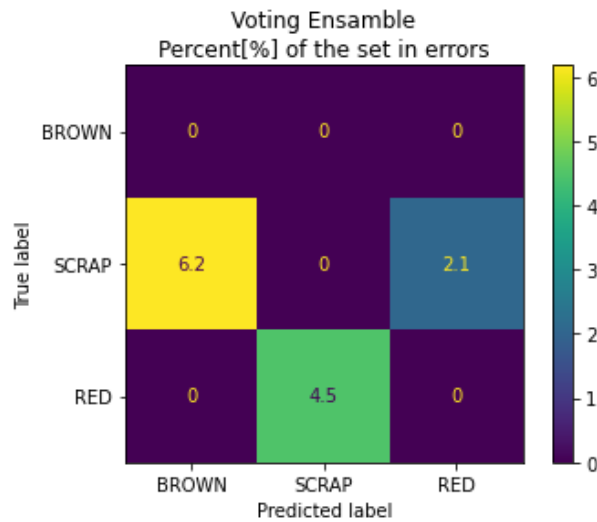


Figure 3.34: Confusion matrix of the errors in experiment 40.

## 4 Result

Here the result of the project will be described. The finished colour quality camera station is shown and the optimal position for the cameras and the simulated reflection. Then a little about the cycle time for the software created for the project and the display. And lastly the finalization of the machine learning model development.

### 4.1 Finished Design of the Station

Figure 4.1 shows the simulation of the light reflection without a diffusion fabric, illustrating the field where the light reflection would be significant enough disturbance to interfere with the camera when capturing an image. The cameras are positioned at an angle where they can avoid the disturbance from the light.

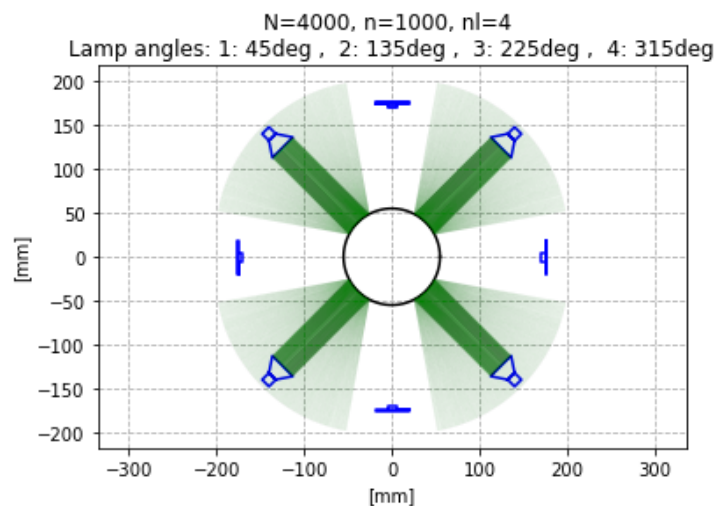


Figure 4.1: The calculated reflections on the pipe, N is the total amount of lines, n is the amount of lines per light source and nl is amount of lamps.

The simulator used for this project, found in appendix G, is owned by Pipelife Norge AS and is confidential. It is not available to the public and can only be viewed upon request.

Due to computing limitations with the Raspberry Pi, cameras could not be started before the main loop and instead start and stop when a frame is needed.

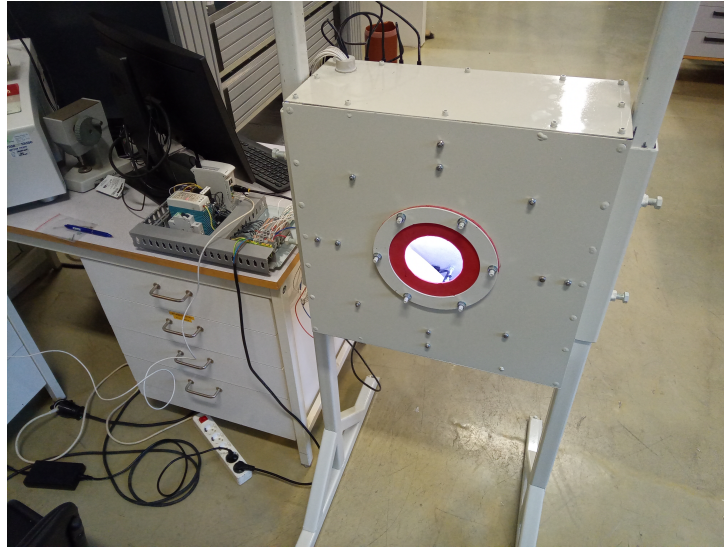


Figure 4.2: The front of the finished of the station.

Figures 4.2 and 4.3 display the completed station. The first figure shows the station awaiting a pipe, with no light escaping unless a pipe is not in the inlet hole. Cameras are numbered 1 to 4, with the top camera being Camera 1 and proceeding clockwise when seen from the front. The second figure demonstrates a pipe inserted into the inlet of the station, with no light escaping the station.

The figure 4.3 shows the a pipe has been inserted and as mention no light escapes the station. The USB hub were fastened on the side to ensure all cables would reach it. The base plate accommodated the equipment and is placed on the table on the side. Unfortunately the cabinet were not delivered in time for the end of the project.

## 4.2 Software

The finished program consistently ran with a cycle time between 120 to 125 seconds. By looking at the speed on the production line from appendix D where the speed ranges from 3.2 to 4.1 m/min at normal operation. By transforming the cycle time to minutes it will be 2 to 2.08 minutes. If the station captured an image without a timer stopping the program between cycles it would be able to get an image every 6.4 to 8.54 m ( $cycletime \cdot linespeed = distance$ ) of a newly extruded pipe.

Figure 4.4 and 4.5 shows the display for the system, and it shows the captured images from one loop. It also shows the evaluation from the algorithm above each image. In the

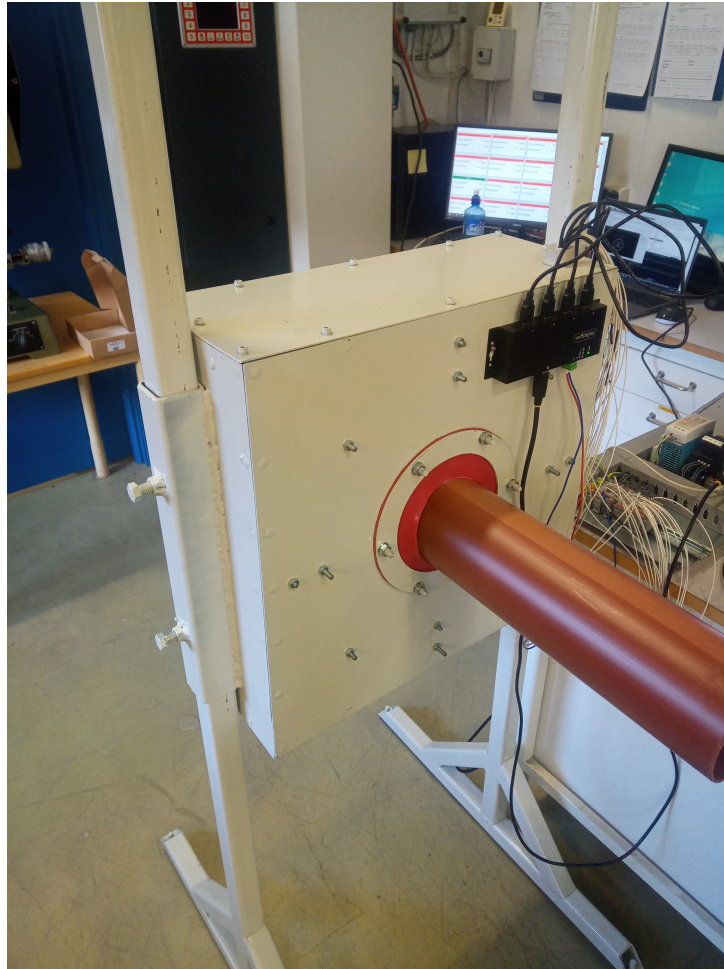


Figure 4.3: The back of the finished station.

first figure, all images have the correct evaluation except Camera 4 where the brown pipe were changed out with a red pipe. The evaluation of this pipe is seen in the next figure. Here there is one false negative at Camera 1 and the rest are true positive.

The display is written in HTML. To be sure it is kept updated with the latest images and evaluations it utilizes this command: `<meta http-equiv="refresh" content="60">`. This command is placed in the header and it will make the file refresh itself every 60 seconds. In the main program before the loop starts a web browser is opened and opens the HTML file. To update the file it is overwritten as an unformatted string stored in the configuration file. In every loop it is formatted with the evaluation from the algorithm.

The source code for this project in appendix H is owned by Pipelife Norge AS. It is not available to the public and it is confidential and can only be viewed upon request.

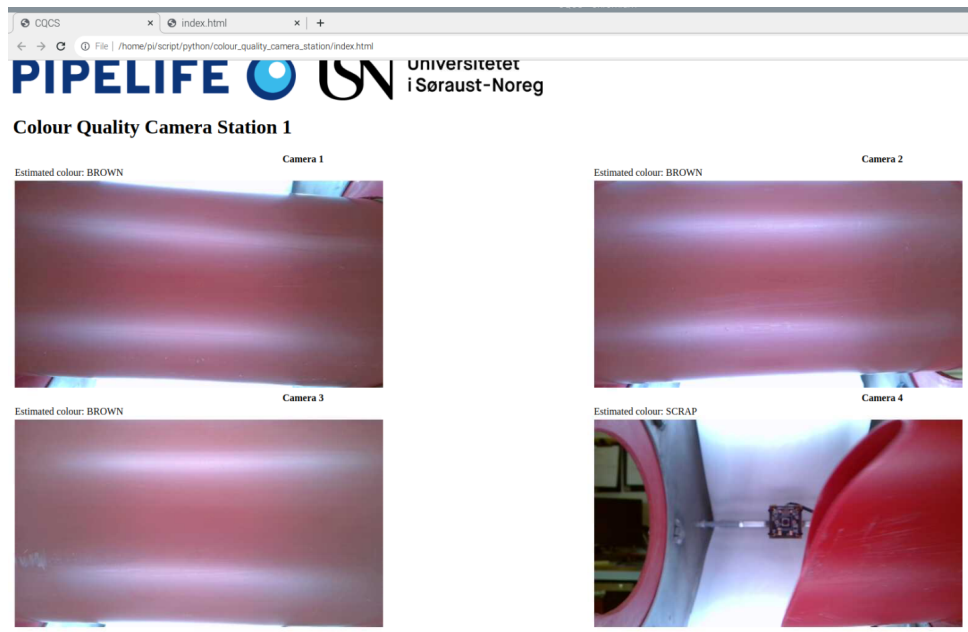


Figure 4.4: The display of the station showing brown pipes and a red pipe about to be placed in the station.

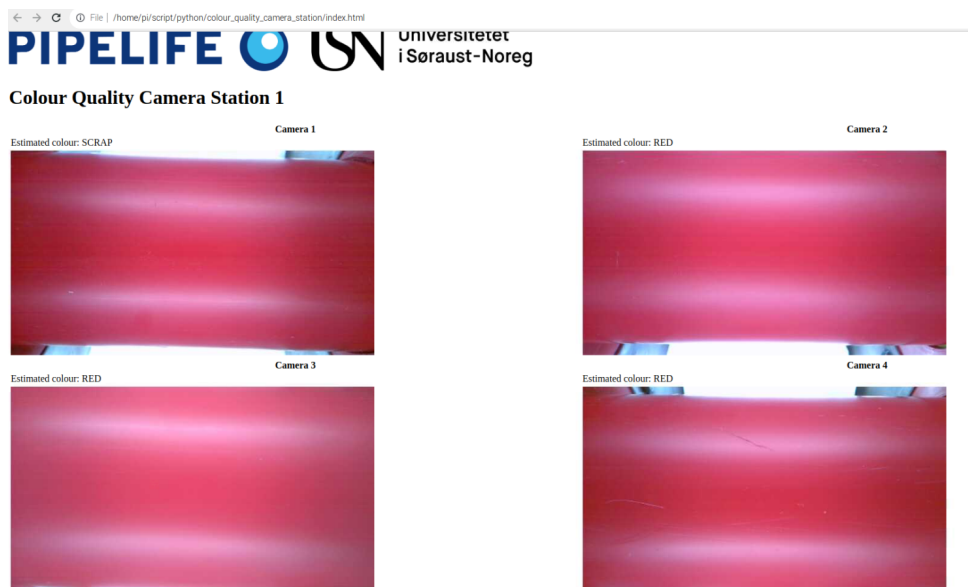


Figure 4.5: The display of the station showing red pipes.

### 4.3 Finalization of Machine Learning Models

Appendix E shows all the experiments conducted to develop a satisfactory classifier. Experiment 40 yielded the most satisfactory classifier, but due to the Raspberry Pi using a 32-bit operative system and 32-bit version of NumPy instead of a 64-bit version, the classifier had to be remade on the Raspberry Pi. It uses voting ensemble machine learning algorithm. The voting ensemble algorithm uses SVC, Decision Tree, Random Forest, and Softmax regression with weights of 2.5, 1.2, 2.5, and 1.8 respectively. The hyperparameters used are the same as in Experiment 40, as shown in figure 4.6.

Table 4.1 displays the classifier's cross-validation results, with high F1 scores for all labels and precision and recall values above 90%. However, which is higher for the labels is different. For SCRAP the the precision is 98.95% and the recall is 90.38%. And for the colours BROWN the precision is 90.91% and the recall is 100.% and precision for RED is 94.11% and the recall is 98.46% respectively.

When testing experiment 42 against the whole dataset to see which images it wrongly predicted and it was noticed that an image in the good BROWN set were misplaced from the bad BROWN set.

Table 4.1: Cross validation of the classifier from experiment 42.

Target Features	BROWN	SCRAP	RED
Precision	0.90909091	0.98947368	0.94117647
Recall	1.	0.90384615	0.98461538
F1	0.95238095	0.94472362	0.96240602
Accuracy	0.90909091	1.	0.94736842

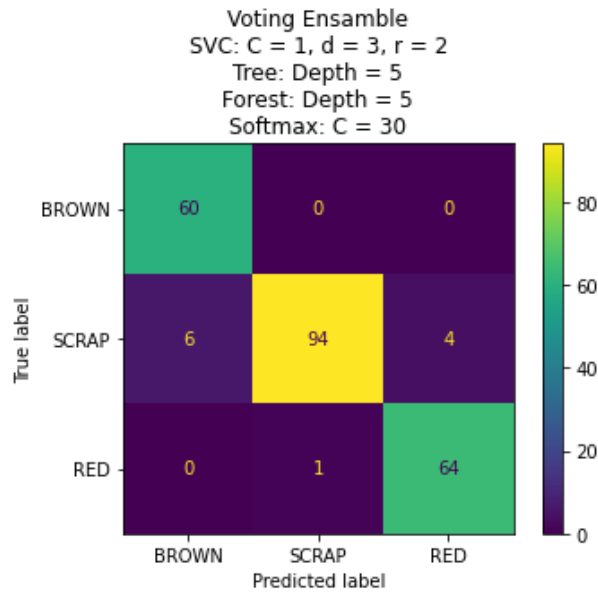


Figure 4.6: Confusion matrix of the preliminary experiment 42.

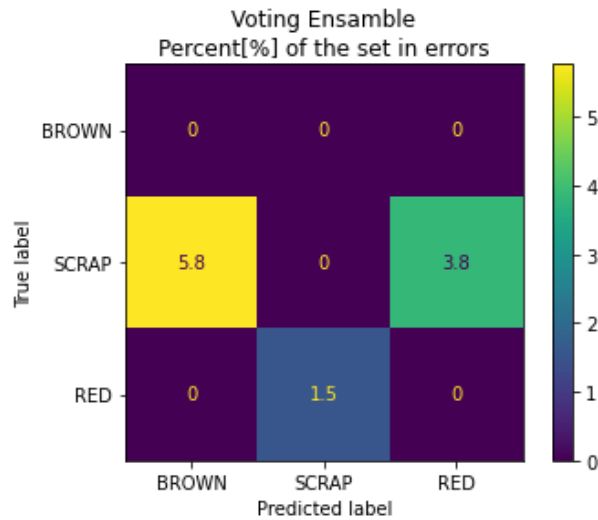


Figure 4.7: Confusion matrix of the errors in the preliminary experiment 42.



## 5 Discussion

During the collection of samples from red pipes, an unusual phenomenon was observed: the hue in the  $b^*$  axis shifted as seen in figure 5.2 and 5.3, giving the images a blue tint as seen in figure 5.1. This phenomenon was not observed in brown pipes or most bad red pipes. It is suspected that this may be due to a heating disturbance affecting the cameras or the cameras auto-adjusting their settings due to the red reflections from the pipe. However, the Raspberry Pi was unable to adjust any camera settings. Despite the blue tint, the algorithm could still make accurate predictions.



Figure 5.1: Images captured by camera one showing the difference between normal colouring and the colouring after the blue tint has appeared.

When experimenting with machine learning it was found that it was not possible to get a consistent error when experimenting. If some of the dataset which has the burn marks on the pipes were not in the training set the classifier would not be able to differentiate them from the good pipe images.

With these results from experiment 42, it is assumed the classifier will identify a lot of real images of pipes that would be considered SCRAP. But it might miss identifying some that would not be considered SCRAP instead of the true colour, but as mentioned when tested against the images from the dataset it was able to find a misplaced image from the bad BROWN set which was found in the good BROWN set.

Normally when accuracy is found to be 100% it usually is a sign of a bad model, but when testing with experiment 42 it was able to even find an image misplaced in another images set, but it also wrongly predicting some images, which were on the border between acceptable and unacceptable, even to the human eye.

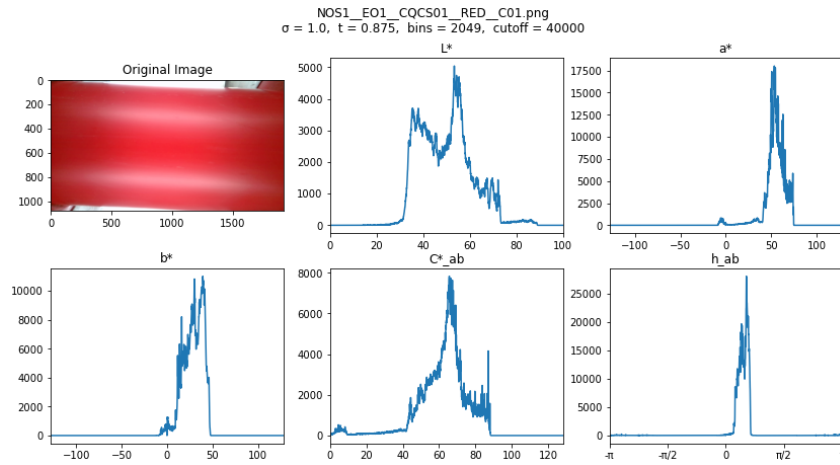


Figure 5.2: Here the image shown is the original image from the good RED set with the produced histogram of the  $L^*$ ,  $a^*$ ,  $b^*$ ,  $C_{ab}^*$ , and  $h_{ab}$ .

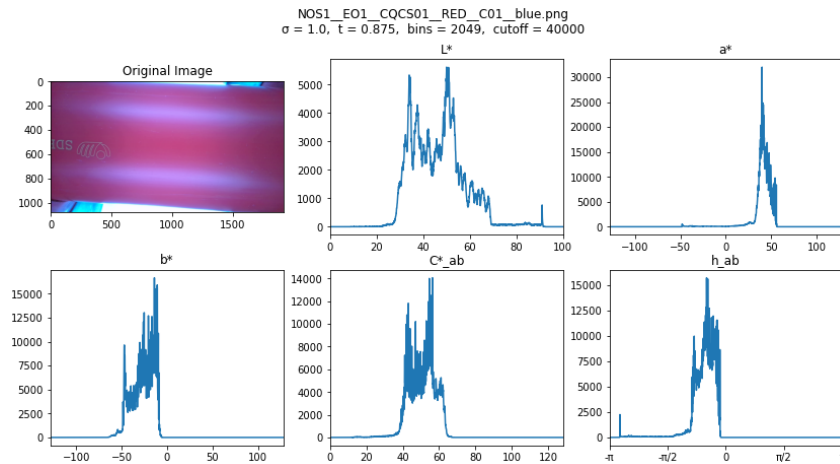


Figure 5.3: Here the image shown is the original image from the good RED set with the produced histogram of the  $L^*$ ,  $a^*$ ,  $b^*$ ,  $C_{ab}^*$ , and  $h_{ab}$ .

It should be noted that almost all experiments of the classifier algorithms were able to differentiate between the colours with about 100% accuracy. With this, it can be inferred that if a new camera system that classifies colours is to be constructed it might be better to use the colour difference equations like equation 2.22 and 2.23 instead of a machine learning algorithm.

After experimenting with the dataset, it became difficult to further increase the accuracy. The current dataset may be insufficient in size and could benefit from an expansion of 5 to 10 times its current size.

After a discussion with between the external partner and the company about the database it was decide a MES database would not be created. As the station were not put in the

production line they deemed it not necessary to make one. Therefore the only database created is a local csv file which has the same setup as the Measurements table in figure 3.13.

The work folder for this project in appendix I is owned by Pipelife Norge AS. It is not available to the public and it is confidential and can only be viewed upon request.

## 5.1 Future Improvements and Use Cases

To improve the accuracy of the of the colour monitoring it would be possible to add more stations and set them in a series. By rotating the cameras and the light sources around the center of the pipe in the new station it will be possible to cover other parts of the pipe wall aswell.

With one station there is a minimum  $180^\circ$  coverage of the outer pipe wall. Then if the the cameras and light in the second station are rotated  $45^\circ$  around the pipe all  $360^\circ$  of the pipe wall should be covered. For redundancy it is also possible to have four stations where the cameras and the light in the third and fourth station are rotated  $22.5^\circ$  around the pipe. The proposed position of the cameras is seen in figure 5.4.

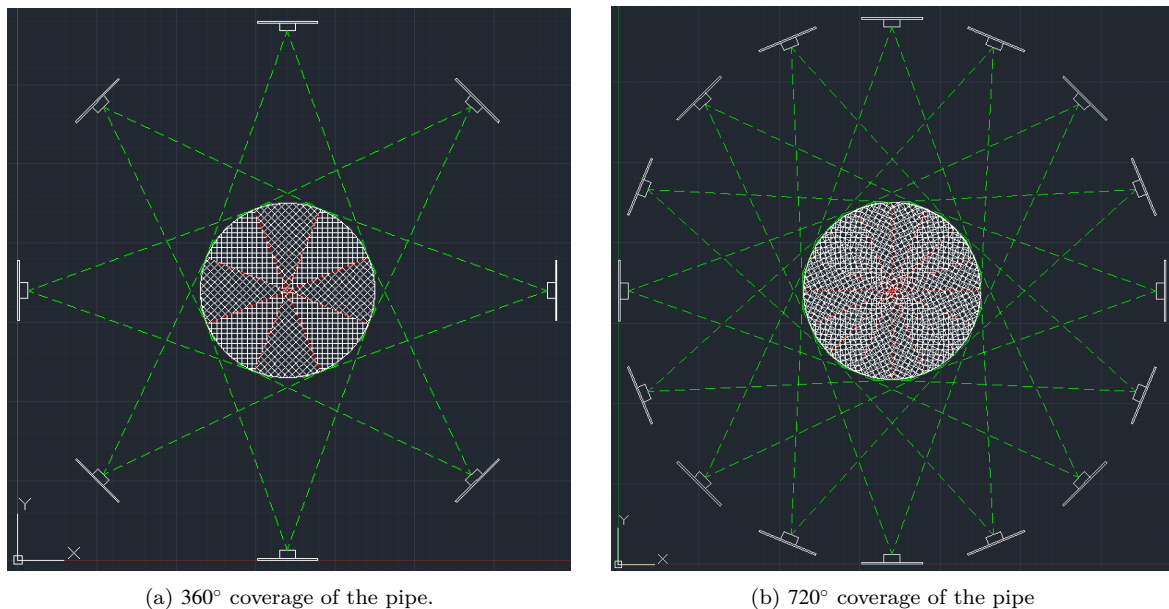


Figure 5.4: The coverage of the of the pipe from the cameras if more stations are placed.

It is clear from testing that the Raspberry Pi used in this project is not the best for image processing. The cameras needed to use an external USB hub for additional power

to operate and decrease the cycle time, which were found to be over two minutes. A stronger computer should be used for the finished developed product.

Before implementing the classifier a way to handle any inaccurate results from the algorithm must be made. A way to limit the inaccuracy is to make the operators client programs reading the evaluations decide if they think the evaluation is correct. A suggestion will be to not give a warning to operators until there are two consecutive readings from the same camera or from two different cameras. If there are still any inaccuracies then an experiment to see how much tolerance for wrong evaluation is needed to minimize any false alarms of the colour quality.

A way to improve the quality of the image from the camera would might be to add dimmers to the lights. The LED strips used is dimmable and the reflections on the pipe would be reduced in the images. Also it might be due to the strong light the peculiar phenomena with the blue tint appeared.

Another way to diffuse the light might be to angle the light towards the walls instead of angling them directly towards the pipe. If the light hits the walls then the pipe than the pipe directly the light might disperse enough to light up the inside of the station, but would not have a visible reflection when an image is captured by the cameras.

An improvement that would be desired is that the station would be able to handle different sizes of pipe. The sizes of the PVC pipes made are from 50 to 250 mm diameter on the production lines similar to the one this one was based on being used for. If one station could be used for different sizes it would cut down on the amount of station needed for the production.

The machine learning algorithm shown in section 4.3 experiment 42 can be seen as a good classifier. But it still classifies some labels wrong. It would be possible for an operator to check each image captured during production if the evaluations were correct. Then after confirmation the classifiers is trained on corrected images by doing online learning or batch learning.

Another improvement to the machine learning algorithm would be to expand the algorithm to handle more colours, but it might be better to train a new model. The target features might need to be changed to using one-hot encoding to be an handle more colours.

But there are also some more improvements to the existing system that are possible. There should be an increase in more images of red and brown pipes in a both good and bad conditions. It would be recommended to expand the sets 5 to 10 times its current size..

When an image is processed the noise is removed. A image where the noise position is easy to find is used to get those position. In the current system the position is calculated

every time an image is processed. To improve the cycle time the position should only be calculated once.

Another problem with removing the noise will come when a classifier will try to monitor the colour of the black pipes. As when the noise is removed from the image it removes the black colour after the threshold is overlaid on the image. This might remove the pipe from the image.

An addition for the operators would be to develop a GUI. This GUI would present the current evaluation from the stations, similarly to the display from figure 4.4 and 4.5. It could also be used to control the station by changing which colour it shall identify as the correct colour. Another ability could be to use the same GUI to verify that the images are correct.

There are some future use cases that can be developed from this project. This type of machine learning and camera system could be added to any of the production lines in the factory where there is a requirement for the quality of the colour. An example would be with the injection molding production lines. After the molding is done a camera would capture an image then the algorithm would evaluate if the colour is correct or not, but the cycle time must be low enough that a prediction can happen before the part is put into a box for delivery.

Another example would be to use the algorithm in the production for double socket and running socket for PVC pipe. When producing these a part of the process is to heat up one end of a PVC pipe then socketing the heated up part. If the pipe is left too long in the heater it will begin to burn. A camera could be inserted to monitor if any burn marks are visible before socketing but as mentioned the cycle time has to be reduced.

Another use case would be for a machine learning model to check if markers have marked the pipes properly. Marker mark is put around the other end of the socketed end of the pipes. By using a camera an image can be captured then the threshold can be computed against pipe and marker line. By using the threshold image directly an algorithm could be made.

Another use case would be to use a machine learning algorithm to check the contours of the sockets. By using a camera an image can be captured then the threshold can be computed against socketed end of the pipe and a background with different colour than the pipe. By using the threshold image directly with an algorithm it should be possible to make a model.



## 6 Conclusion

To conclude a functioning colour quality camera station was made and functions as expected, but with some improvements needed. There is a peculiar phenomena where the yellow-blue opponency,  $b^*$ , is shifted towards blue and gives the images a blue tint after some time has passed, but this did not affect accuracy of the predictions from the machine learning algorithm. It is recommended this phenomena is looked into after the Raspberry Pi has been changed to a stronger computer that does not need a USB hub to power the web cameras. When the cameras used the USB hub, OpenCV was unable to adjust any settings. The reason for this is unknown, but as it was possible when the hub was not in use it is suspected that if Raspberry Pi is replaced with stronger hardware this problem would be removed.

The reflection on the pipe images was still visible but not as disturbing as in the previous proof-of-concept. One layer of diffusion fabric was used to diffuse the light, but in future iteration it is recommended to add more layers of fabric or angle the light sources and let the light bounce of the walls before lighting up the pipe.

A satisfactory classifier was made for the station in experiment 40. Even if it had to be remade due to the experiment being trained on a 64-bit computer, while the Raspberry Pi uses a 32-bit operating system. The remade model in experiment 42 used the same hyperparameters and got about the same performance and was better then expected. It was able to find a misplaced image from the bad BROWN set in the good BROWN set.

It is expected the model will be able to identify SCRAP, but it might misidentify the other labels as SCRAP. A possible workaround to this problem would be for a GUI on the operator side waiting for a repeated identification, but an analysis must be done to decide how many repetitions would be needed.

It was also clear that using voting ensemble made a better model. When combining multiple types of algorithms it was able to get a more stable and accurate model, but a larger dataset would make an even better model. An example would be that there should be more pipes with burn marks in the set as it was prone to be not properly added into the training set.





# Bibliography

- [1] ‘Pipelife norge as.’ (), [Online]. Available: <https://www.pipelife.no/om-oss.html>.
- [2] H.-C. Ringstad, *IIA1319: Software engineering: Assignment 4: Software development based on user specification: Quality control*, Course Assignment, 2021.
- [3] F. Frisvold and J. G. Moe, *Statistikk for ingeniører*. Bergen (Norway): Fagboksforlaget Vigmostad & Bjørke AS, 2004.
- [4] J. Schanda, Ed., *Colorimetry: Understanding the CIE System*, 1st ed. Hoboken (NJ): John Wiley & Son. inc, 2007.
- [5] C. Oleari and G. Simone, *Standard Colorimetry: Definitions, Algorithms and Software*, 1st ed. Chichester (UK): John Wiley & Son. Ltd, 2016.
- [6] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias *et al.*, ‘Scikit-image: Image processing in Python,’ *PeerJ*, vol. 2, e453, Jun. 2014, ISSN: 2167-8359. DOI: 10.7717/peerj.453. [Online]. Available: <https://doi.org/10.7717/peerj.453>.
- [7] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, ‘Scikit-learn: Machine learning in Python,’ *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [8] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. Sebastopol (California): O’Reilly Media, Inc., 2019.
- [9] ‘Raspberry pi 4 8g model b | raspberry pi 4 b 8gb | RS.’ (), [Online]. Available: <https://no.rs-online.com/web/p/raspberry-pi/1822098> (visited on 17/04/2023).
- [10] ‘F12-RGBNW-24-60-65-FP | PowerLED 24v dc RGBW LED strip light, 4000k colour temp, 5m length | RS.’ (), [Online]. Available: <https://no.rs-online.com/web/p/led-strip-lights/1845180> (visited on 17/04/2023).
- [11] ‘8 megapixels USB camera with microphone (compatible with raspberry pi/ LattePanda/ jetson nano).’ (), [Online]. Available: <https://www.dfrobot.com/product-2188.html> (visited on 17/04/2023).
- [12] ‘5g4aaindp-USB-a-HUB | StarTech.com 4 port USB 3.0 USB a USB 3.0 hub, USB bus powered, 7.0 x 2.3 x 0.9in | RS.’ (), [Online]. Available: <https://no.rs-online.com/web/p/usb-hubs/2566913> (visited on 17/04/2023).



## **Appendix A**

# **Colour Quality Monitoring System for Plastic Pipes Exterior Wall Colour with Machine Learning and Image Processing Thesis Problem Description**

Here the master thesis description is found and signed by the student and USN supervisor.

## FMH606 Master's Thesis

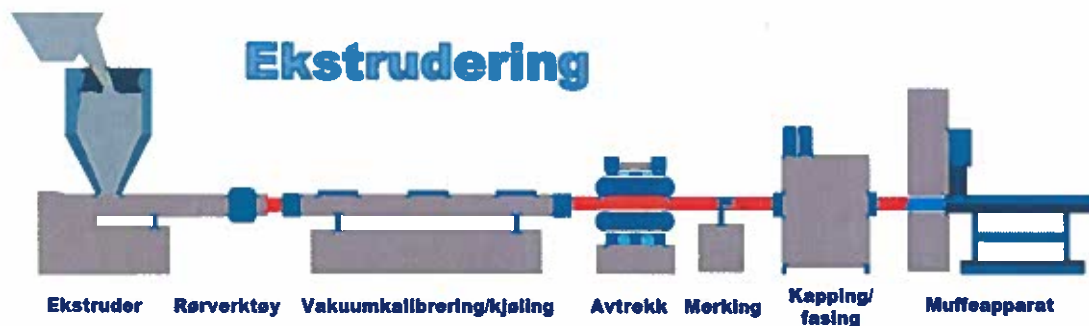
**Title:** Colour quality monitoring system for plastic pipes exterior wall colour with machine learning and image processing

**USN supervisor:** Ru Yan

**External partner:** Daniel Zwick, Pipelife Norge AS

### **Task background:**

Pipelife Norge AS (<https://www.pipelife.no/>) is one of the biggest producers of plastic pipes in Norway and is part of Pipelife International GmbH. Today, plastic pipes are produced on production lines called extrusion lines, as illustrated in the figure below.



The production process starts with melting plastic pellets or granules made from recycled pipes and mixing a colouring mixture into a plastic liquid. Then, the plastic liquid will be extruded and cooled into a solid pipe. Further, the pipe will be cut into segments by a cutting unit, socketed at a pipe-socketing bench, and placed in a pipe cart. In the end, operators will bundle the pipe cart onto a pallet. During the last step, the operators also will perform quality checks / visual inspections of the pipe exterior colour before bundling the pipes.

However, this method of to check the quality is no longer available after installing a new automatic pipe bundle system in the factory. The operator cannot visually inspect the produced pipes while the bundling machine is running. Nevertheless, colour quality checks are still required.

After discussing with the new bundle system's manufacturer, it was concluded that making a regular sensor-based quality check system for pipe exterior colour would be complicated and unreliable. Therefore, applying computer vision (camera-based) and machine learning techniques to develop a colour quality inspection system has been considered. Moreover, to realize the feasibility, a proof of concept for a camera-based solution was made using one camera, which was considered a success.

### **Task description:**

- I. Extend the proof of concept of a colour quality check for a PVC pipe exterior colour:
  - 1) Design a lightproof box where a produced pipe can travel through it.
  - 2) Find an optimal position(s) for light and cameras to monitor up to 360° of the pipes outside wall.
- II. Data collection and Model development:

- 3) Collecting training and test data samples from both "good"/ proper coloured and "bad"/ improper coloured pipes
  - 4) Applying suitable image processing and machine learning algorithms and training the model(s) to identify the acceptable colour range.
  - 5) A minimum of one colour should be verified.
  - 6) A minimum of one size should be verified.
  - 7) The points (3) – (6) may be repeated several times based on the results obtained in (5) and (6).
- III. Developing a prototype of the colour quality check system:
- 8) Prepare the System to be used as a feedback sensor for an automated control system for colour mixing to minimize production costs.
  - 9) If time permits, investigate any unforeseen challenges or improvements.

**Student category:** IIA

**Is the task suitable for online students (not present at the campus)?** Reserved for Hans-Christian Ringstad


**Practical arrangements:**

Hardware and software will be available from Pipelife Norge AS with necessary support.

**Supervision:**

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

**Signatures:**

Supervisor (date and signature):  27.01.2023

Student (write clearly in all capitalized letters): HANS-CHRISTIAN RINGSTAD

Student (date and signature): 27.01.23 H-C Ringstad



# Appendix B

## Gantt Chart

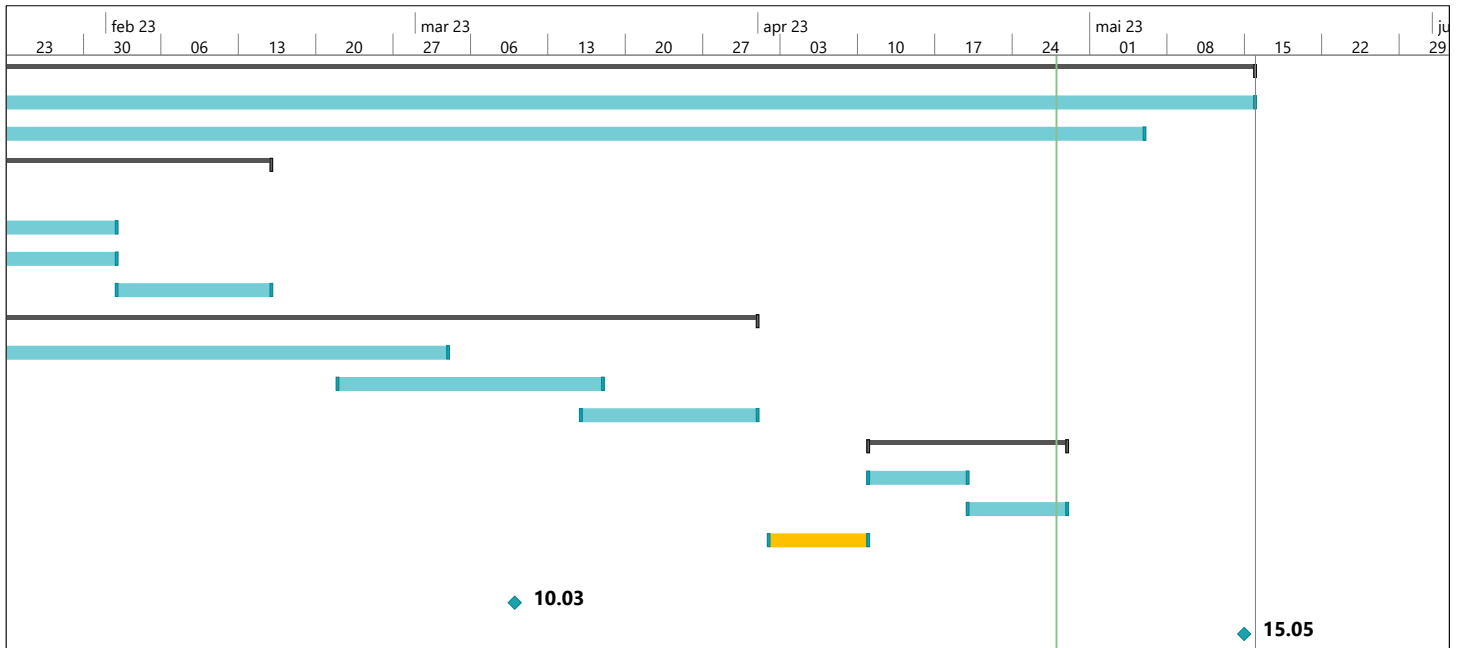
Here the timetable of the project can be seen. A Gantt chart was used for readability purposes.

ID	Aktivitetsmodus	Aktivitetsnavn	Varighet	Start	jan 23					
					19	26	02	09	16	23
1		<b>Write Master Thesis</b>	<b>96 dager</b>	<b>ma 02.01.23</b>						
2		Write thesis	96 dager	ma 02.01.23						
3		Literature search	90 dager	ma 02.01.23						
4		<b>Design lightbox</b>	<b>28 dager</b>	<b>ma 09.01.23</b>						
5		Make some models for the software	10 dager	ma 09.01.23						
6		Find optimal position	13 dager	ma 16.01.23						
7		Draw the first design	13 dager	ma 16.01.23						
8		Program the imaging software	10 dager	to 02.02.23						
9		<b>Data collection and model development</b>	<b>59 dager</b>	<b>ti 10.01.23</b>						
10		Collect training set and test set	39 dager	ti 10.01.23						
11		Image processing and data analysis	18 dager	on 22.02.23						
12		Create ML model	12 dager	to 16.03.23						
13		<b>Further development</b>	<b>14 dager</b>	<b>ti 11.04.23</b>						
14		Find and comment on further improvements	7 dager	ti 11.04.23						
15		Find and comment on further use cases	7 dager	to 20.04.23						
16		Easter	7 dager	sø 02.04.23						
17		Formal meeting 1	0 dager	fr 06.01.23						
18		Formal meeting 2	0 dager	fr 10.03.23						
19		Date of delivery	0 dager	ma 15.05.23						



Prosjekt: Gantt chart Dato: fr 28.04.23	Aktivitet		Manuell sammendragsfremheving	
	Deling		Manuelt sammendrag	
	Milepæl		Bare start	
	Sammendrag		Bare slutt	
	Prosjektsammendrag		Eksterne aktiviteter	
	Inaktiv aktivitet		Ekstern milepæl	
	Inaktiv milepæl		Tidsfrist	
	Inaktivt sammendrag		Fremdrift	
	Manuell aktivitet		Manuell fremdrift	
	Bare varighet			





Prosjekt: Gantt chart Dato: fr 28.04.23	Aktivitet		Manuell sammendragfremheving	
	Deling		Manuelt sammendrag	
	Milepæl		Bare start	
	Sammendrag		Bare slutt	
	Prosjektsammendrag		Eksterne aktiviteter	
	Inaktiv aktivitet		Ekstern milepæl	
	Inaktiv milepæl		Tidsfrist	
	Inaktivt sammendrag		Fremdrift	
	Manuell aktivitet		Manuell fremdrift	
	Bare varighet			



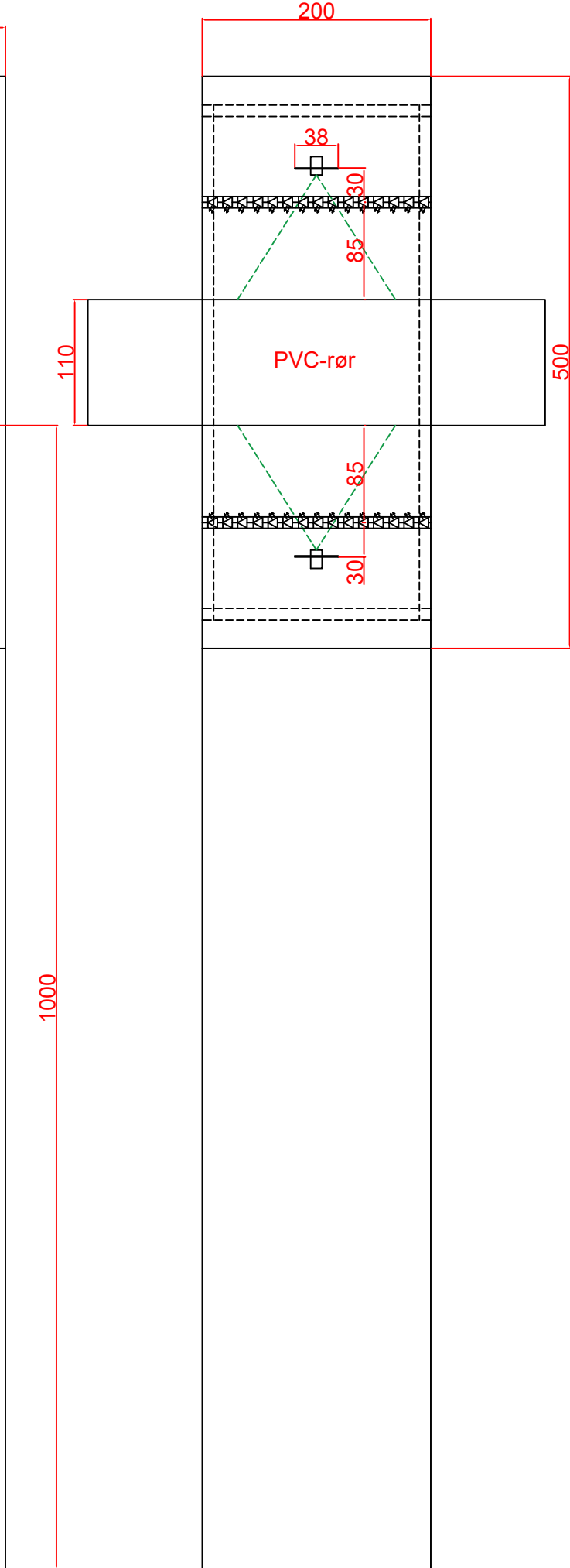
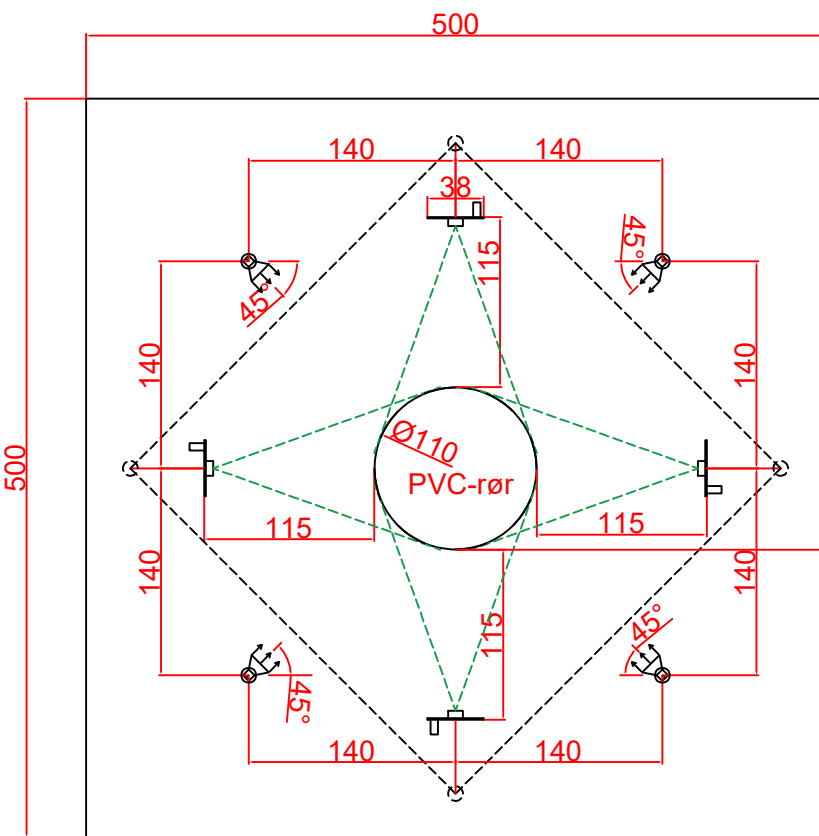
# Appendix C

## First Draft of the Station

The first draft of the colour quality camera station. The first page is the draft of the station itself and shows where the cameras, the LED's is placed and diffusion fabric. The black dotted line is the diffusion fabric and the green dotted line is the approximate field of vision of the camera. The second the page is the arrangement diagram for the main electrical cabinet for the station. The last page is the electrical schematic of the main electrical cabinet of the station.

Table C.1: Here the references and the description of the equipment in the found in the main electrical cabinet.

Reference	Description
-H1, -H2, -H3, -H4	PowerLED RGBW LED Strip 5m 24V dc
-A1	Raspberry Pi 4 B 8GB
-J1, -J2, -J3, -J4	DFRobot, USB 2.0 with 3280 x 2464, FIT0729
-G1	Power supply, 230V AC → 24V DC
-G2	Power supply, 24V DC → 5V DC
-F1	Automatic Fuse, 1P 6A B 230V
-K1, -K2, -K3, -K4	3.5 A Solid State Relay, Zero Cross
-X1, -X2, -X3, -X4, -X5	Grey PT 1.5/S-QUATTRO Feed Through Terminal Block
-N1	4 Port USB 3.0 USB A USB 3.0 Hub, USB Bus Powered

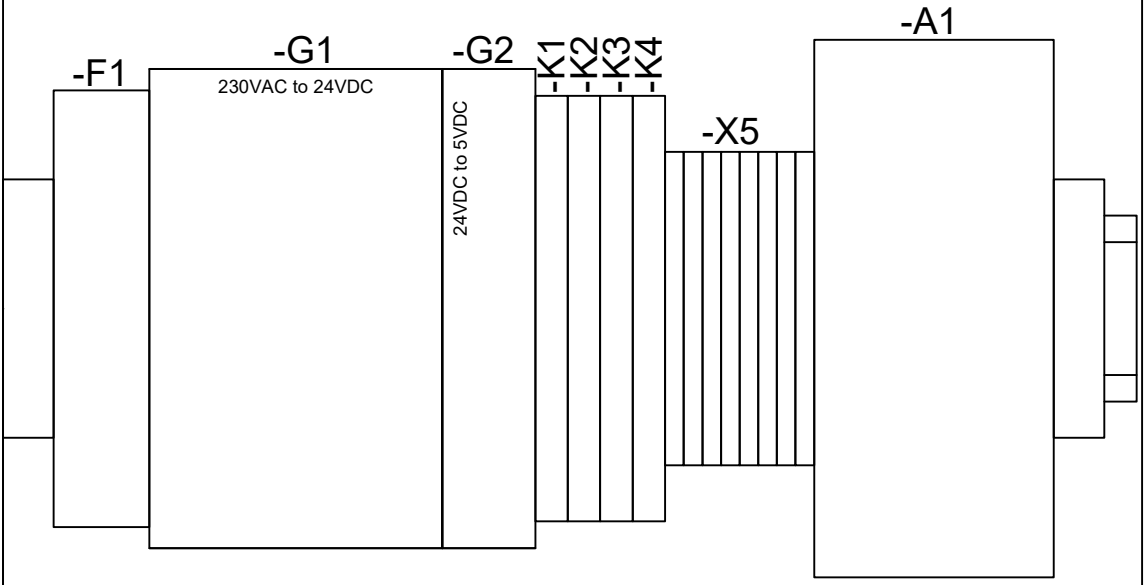


365

250

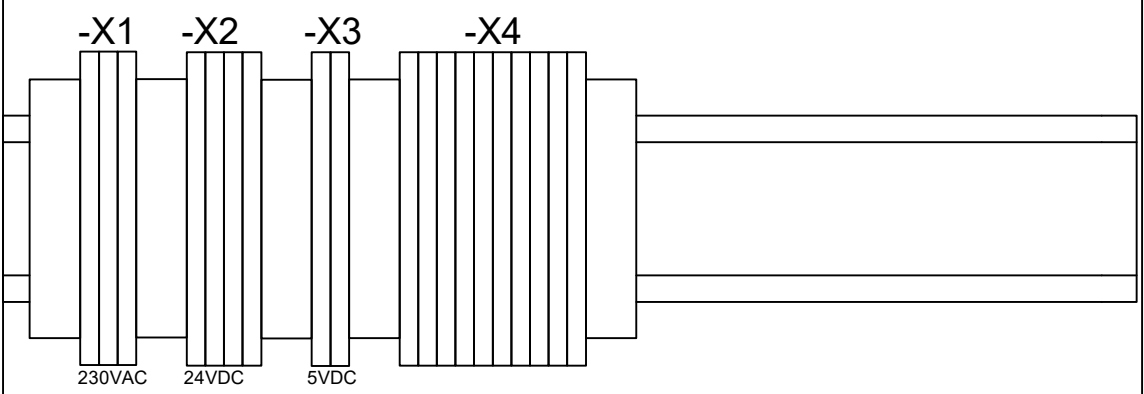
214

22,5



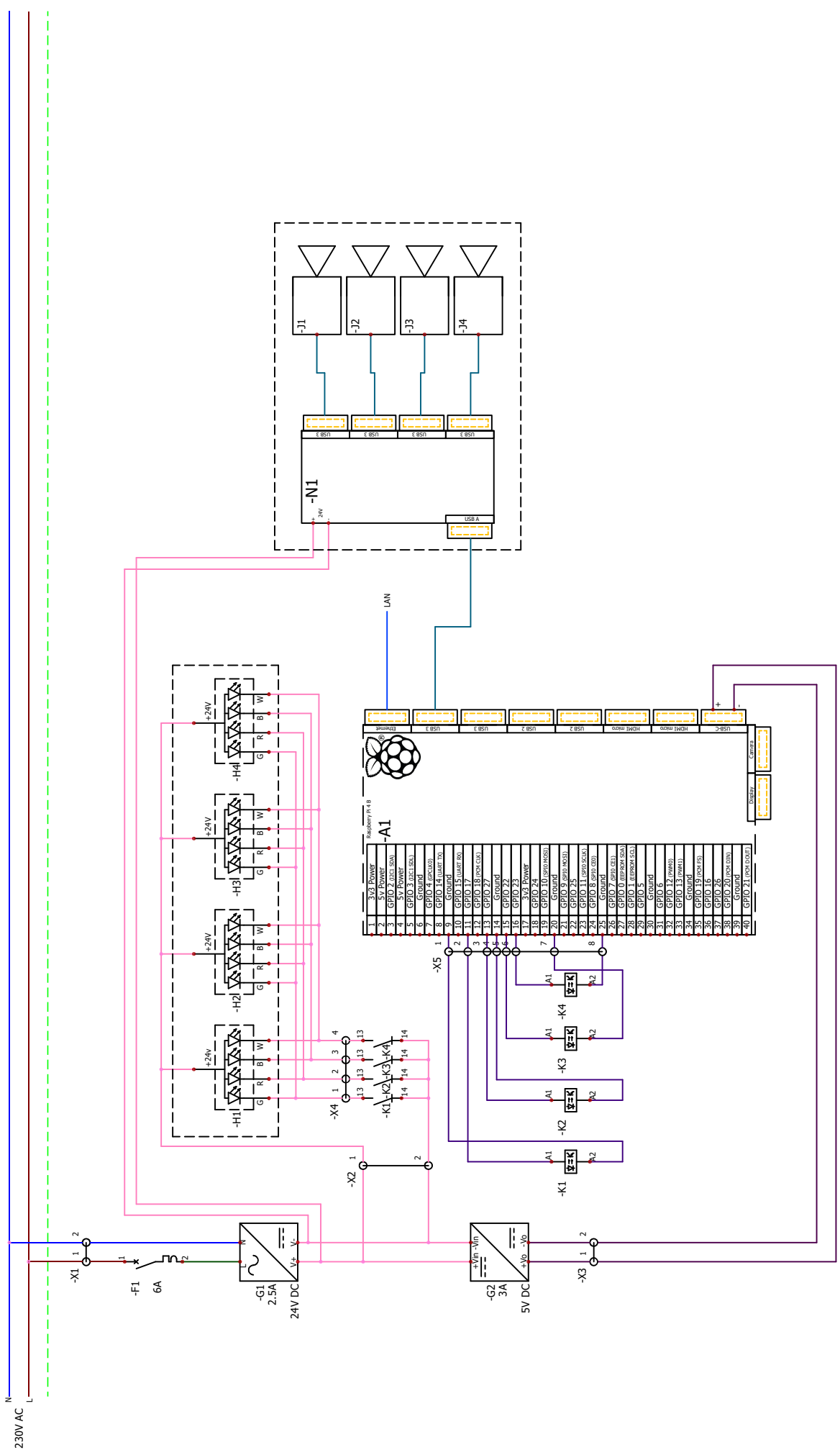
95

95



95

57,5

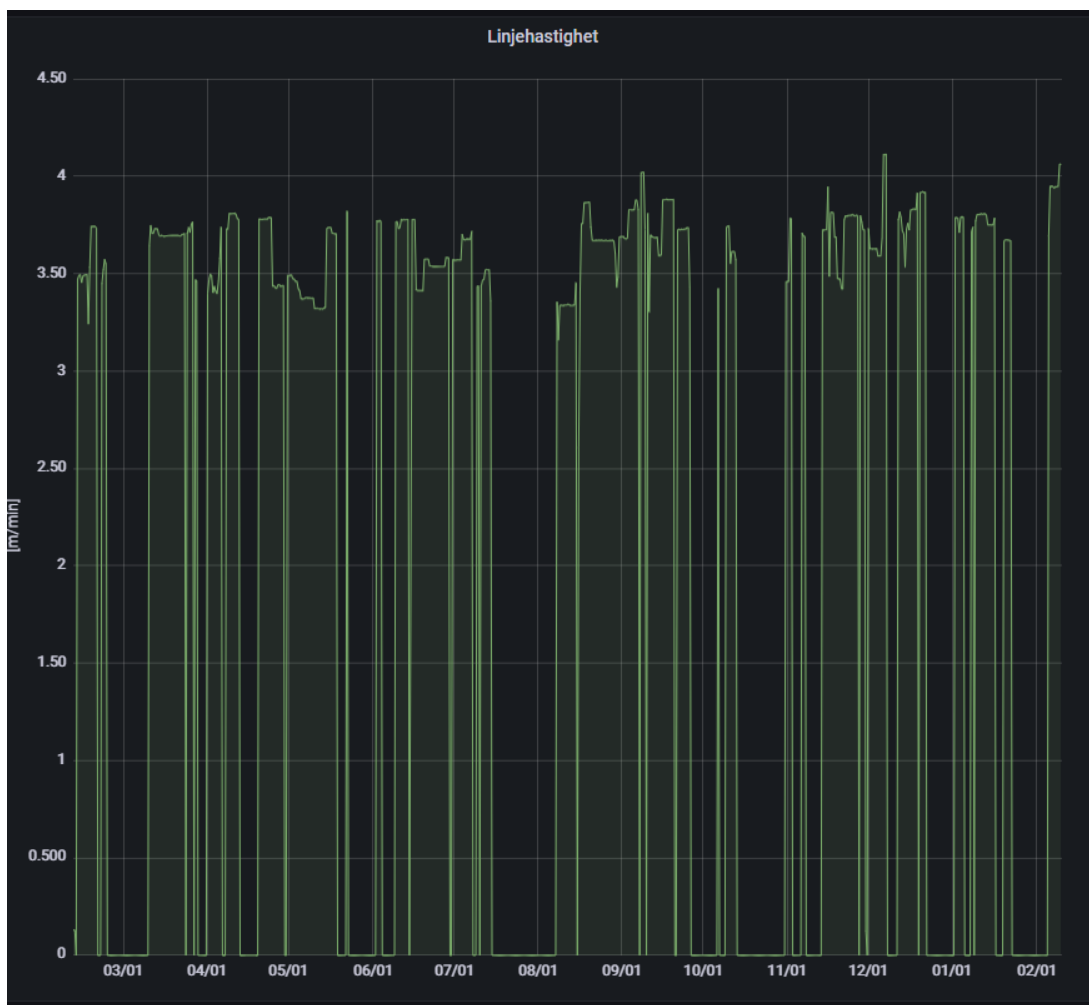


PiPelfe Norge AS FOU		Document book		REVISION	
				0	
				SCHEME	
				03	
				User data 1	
				User data 2	
				CHANGES	
				User data 1	
				User data 2	
				NAME	
				horinhan	
				DATE	
				19.01.2023	
				REV.	
				0	
				User data 1	
				User data 2	
				LOCATION:	
				+L1	
				Main electrical closet	
CONTRACT: 236815					

# Appendix D

## Line Speed on E01 for the Year 2022

Here the graph for the line speed of the production line E01 where mostly PVC 110 diameter pipes are produced. As seen in the graph the speed when the line is active ranges from 3.2 to 4.1 m/min.







# Appendix E

## ml\_experiment\_log.pdf

A log of the experiments done to decide which machine learning algorithm to be used for the project and its corresponding hyperparameter.

The experiment log is the property of Pipelife Norge AS and is not available to the public and is only available on request at:

Pipelife Norge AS  
Hannesvegen 97  
6650 Surnadal  
Telefon: +47 71 65 88 00  
E-mail: [firmapost@pipelife.com](mailto:firmapost@pipelife.com)



# Appendix F

## **pipe\_image\_set.zip**

The complete dataset used to train the machine learning models.

The dataset is the property of Pipelife Norge AS and is not available to the public and is only available on request at:

Pipelife Norge AS  
Hammesvegen 97  
6650 Surnadal  
Telefon: +47 71 65 88 00  
E-mail: [firmapost@pipelife.com](mailto:firmapost@pipelife.com)



# Appendix G

## circleReflectionSim.py

The script is the property of Pipelife Norge AS and is not available to the public and is only available on request at:

Pipelife Norge AS  
Hannesvegen 97  
6650 Surnadal  
Telefon: +47 71 65 88 00  
E-mail: [firmapost@pipelife.com](mailto:firmapost@pipelife.com)



## Appendix H

### **colour\_quality\_camera\_station\_source\_code.zip**

The source code for the software used in the colour quality camera station.

The code is the property of Pipelife Norge AS and is not available to the public and is only available on request at:

Pipelife Norge AS  
Hammesvegen 97  
6650 Surnadal  
Telefon: +47 71 65 88 00  
E-mail: [firmapost@pipelife.com](mailto:firmapost@pipelife.com)





# Appendix I

## FMH606-1 23V Master's Thesis.zip

The work folder of this project is the property of Pipelife Norge AS and is not available to the public and is only available on request at:

Pipelife Norge AS  
Hannesvegen 97  
6650 Surnadal  
Telefon: +47 71 65 88 00  
E-mail: [firmapost@pipelife.com](mailto:firmapost@pipelife.com)