NTNU

Norwegian University of Science and Technology

Pick and Sort Robot

IE303812 Real Time Programming

Vegard Solheim, Petter Drønnen, Magnus Kvendseth Øye

A student project presented for the Real Time Programming Course

Faculty of Information Technology and Electronics

Norwegian University of Science and Technology

Norway

November 27, 2019

Preface

This report is written on NTNU campus Ålesund for NTNU in the subject IE303812 Real Time Programming. The project was started in September and finished in late November. The group consist of three students from Automatiseringsteknikk at NTNU Ålesund. All members have similar backround from automation, and all three have certificate of apprenticeship in automation.

Ålesund, November 27, 2019

Vegard Solheim	Petter Drønnen	Magnus Kvendseth Øye

Acknowledgement

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

- Our supervisor Ivar Blindheim at NTNU for guidance throughout the project.
- Anders Sætersmoen at NTNU for the help with ordering parts, and lending equipment.

Executive summary

The aim of this project is to develop a robot system for sorting multiple and different shaped objects. In order to perform this task, the robot will require a high level of precision and accuracy. Furthermore, the system should be able to handle robust working conditions, where the external light source may vary.

The result left us confident that we have proved the concept, as well as creating a foundation for a full-scale operation in an industrialized context. This project shows that picking and sorting different type of objects is possible by the use of robots and automated systems, and the solutions and technology presented in this project can be used to automate pick and sorting task with high precision and accuracy in real time.

Contents

Terminology

BOM BIll Of Material

CV Computer Vision

DOF Degrees Of Freedom

I/O Input and Output

UART Universal Asynchronous Receiver/Transmitter

OS Operating System

TTL Transistor to Transistor Logic

USB Universal Serial Bus

PID Proportional integral derivative controller

GUI Graphical User Interface, makes it possible to interact with a computer

API Application Programming Interface, activates functions from a remote software

REST REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web

TCP Transmission Control Protocol, connection oriented transmission protocol of information.

IP Internet Protocol is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries

JSON JavaScript Object Notation is a lightweight data-interchange format

AI Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.

CNN Convolutional Neural Network is a Deep Learning algorithm which can take in an input image

MCU Microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

This project is meant to set the foundation of the grade in the subject *IE303812 Real Time Programming*. The requirements states that the project should consist of multiple threads, and have thread-safe operations. Meaning code only manipulates shared data structures in a manner that ensures that all threads behave properly and fulfill their design specifications without unintended interaction.

1.2 Project introduction

We seek to prove that an autonomous industrial robot can optimize the pick and sort operations in a production lineup. With doing so, we wish to eliminate human interaction, as this can be a tedious task over time. In this paper, the emphasis is on real time programming design, machine learning, and finally the sorting sequence. During consideration of different choices, the priority will be on robustness, speed and accuracy.

1.3 Aim and objectives

The aim for this project is to build a prototype of an autonomous industrial robot as proof of concept for future automated solutions in sorting systems. The robot will operate on a frame

system, where the objects are localized under, and in between the frame. The main task will be to detect, classify and sort the objects as fast and accurate as possible. Where as the core objectives for creating this robot is mentioned below:

- Design and develop a prototype robot suited to operate in various types of system, like for example a conveyor belt system. It needs to have high enough accuracy in both classification and movement. During development, concurrency and thread safe operations will be taken into account.
- Program the robot so it can position it self within a three dimensional coordinate system.
- Develop software for locating and sorting of objects by the use of *computer vision* and *Convolutional Neural Network*.
- Develop a system that is fast and accurate, and will work in robust environments.
- Assemble the prototype and test it according to specified requirements.
- Design and implement a graphical user interface that can easily be operated by instructed personnel.

1.4 Report content

This technical report does not follow any standard layout, but is based on the recommended standard layout for a bachelor thesis. It contains some additionally chapters that elaborates on system implementation and design.

Chapter 2 - Theoretical Basis contains the theoretical content needed to make decisions and solve the problems later on in the report.
Chapter 3 - Approach is a description of what is deemed as necessary specifications as well as how to approach tasks and problems to develop the robot. At last, some tests to perform on individual parts of the robot are described.
Chapter 4 - Materials contains the materials, components and information used for creating the prototype in this project.

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

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

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

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

Chapter 9 - Discussion discusses the results, problems encountered along the way, and improvements for the future.

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

Chapter 2

Theoretical basis

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

2.1 Industrial robots

An industrial robot is a mechanical and programmable unit that can perform repetitive tasks. These robots possess at least three degrees of freedom, left-right, up-down and forward-backward. In addition, to move a robot freely in a three-dimensional space, three degrees-of-freedom has to be added. These are called yaw, pitch and roll, and is providing rotation around the x, y, and z-axis. Robots achieve motion using joints controlled by actuators. Two types of joints can be applied. The first is a prismatic joint (P) which provides linear motion, and the other is a revolute joint (R) that provides rotational motion. Every robot has a base and a free end. The base works as a fixed link to which all the other links are relative to. A free end is where the end-effector or a tool can be assembled [47].

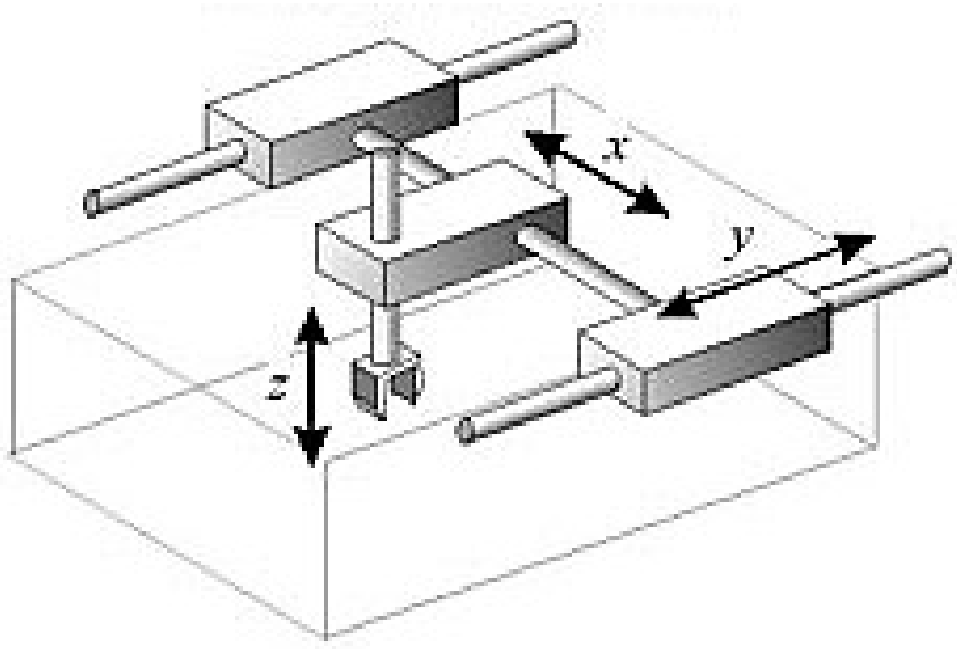


Figure 2.1: Cartesian robot

2.1.1 Cartesian robot

Unique for Cartesian robots is that they have three independent prismatic joints (PPP), which leads to a linear kinematics. They are constructed to move in linear motion with high repeatability through the full workspace. Although this robot often involves a considerable floor space, it reaches a large area within this space. Therefore it considered to be workspace efficient. A drawback however is that Cartesian robots are one of the slower manipulators on the marked. A basic illustration of the robot is shown in figure 2.1 [42].

2.2 Robot kinematics

Kinematics is the study of motion of a robot regarding position, velocity, and acceleration of the manipulator joints. Kinematics does not consider the force or torque required for generating motion. It is used for transforming a coordinate system to another, and back. Figure 2.2 and 2.3 illustrates the transformation from a joint coordinate system to a coordinate system representing real-world coordinates. A joint coordinate system defines each joint with its position or angle. The real world coordinate system uses Cartesian coordinates and has its origin at the base

of the robots frame. In robot kinematics there are two main problems that needs to be solved, which is forward kinematics and inverse kinematics [54].

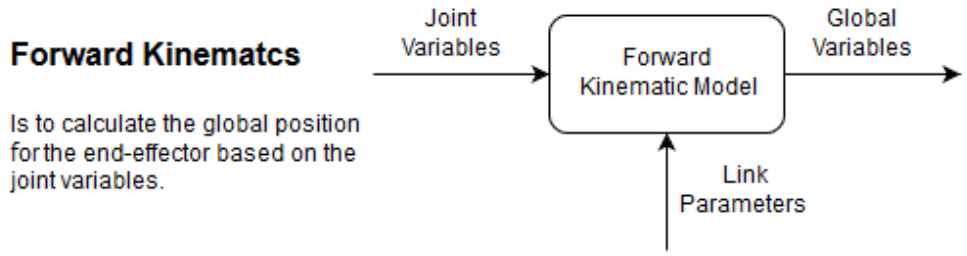


Figure 2.2: Forward kinematics

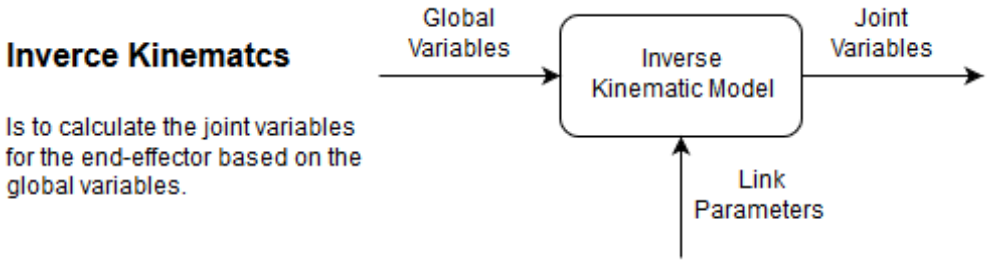


Figure 2.3: Inverse kinematics

2.3 Linear motion

Some electrical machines require linear movement to work efficient. Because most electrical actuators provide rotational motion, it is necessary to convert rotational motion into accurate linear motion. There are several ways of achieving this conversion. Some of the commonly used techniques are either by belt-driven actuators, ball screw driven actuators, or by rack and pinion driven actuators. The three actuator implementations, all apply linear motion by rotary motors, but they differ in strengths and limitations. Accuracy provided by the different methods depends on the actuator providing the revolution motion[17].

2.3.1 Belt-driven motion

Belt-driven actuators as shown in figure 2.4 are often used in applications with movement of low weight components over longer distances. They use a timing belt between two or more circular pulleys and a clamp for moving a carry along the motion of the timing belt. Motion is applied

by turning one of the pulleys, and thereby moving the belt. The distance traveled by the trolley is relative to the size of the circular pulley where motion is applied. The distance a belt moves is given by the pulley pitch circumference using the pitch diameter.

Figure 2.4: Belt driven actuator

2.4 Electric motors

Electric motors are machines that convert electrical energy to mechanical energy.

2.4.1 Brushless motor

A brushless motor is asynchronous motor powered by direct current (DC) electricity via an inverter, or switching power supply which produces an alternating current (AC) electric current to drive each phase of the motor via a closed loop controller. The controller provides pulses of current to the motor windings that control the speed and torque of the motor [39].

2.4.2 Motor driver

Motor drivers controls the torque, speed, and direction of rotary and linear electric motors. They function by taking a low-current control signal and turning it into a higher-current signal that drives the motor [31].

2.5 Encoder

An encoder is a sensing device that provides feedback on motion. Encoders convert motion to an electrical signal that can be read by some type of control device in a motion control system, such as a counter or PLC. The encoder sends a feedback signal that can be used to determine position, count, speed, or direction. A control device can use this information to send a command for a particular function [7].

2.5.1 Incremental encoder

An incremental encoder generates a pulse for each incremental step in it's rotation. Although the incremental encoder does not output absolute position, it can provide high resolution. For example, an incremental encoder with a single code track, referred to as a tachometer encoder, generates a pulse signal whose frequency indicates the velocity of displacement [25].

2.6 Festo vacuum generator

Festo vacuum generators operates by the Festo principle. It generates vacuum by sending motive fluid into the Festo chamber through a narrow nozzle. Both compressed gas and liquid, can be used as motive fluid. Because the nozzle has smaller volume than the input port, the motive fluid will have higher speed and lower pressure inside the chamber [9].

2.7 PID - controller

A proportional integrated derivative controller (PID controller) is a feedback mechanism used in industrial control systems and a variety of other applications that require continuous modulated control. A PID controller continuously calculates an error value *e(t)* that is the difference between the desired set point *(SP)* and a measured process variable *(PV)* and uses a correction based on proportional, integrated and derivative conditions to make the error value move to zero.[50].

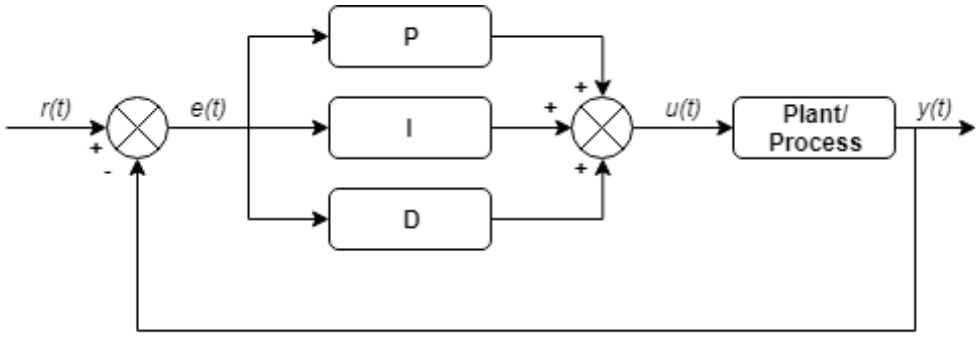


Figure 2.5: PID Controller

2.7.1 Operation

The feature of the PID controller is the ability to use the three control conditions proportional, integrated and derivative influence on the regulator output to apply accurate and optimal control. Figure 2.5 illustrates a block diagram showing the principles for how these concepts are generated and used. It shows a PID controller that continuously calculates an error value *e(t)* which is the difference between a desired set point *r(t)* and a measured process variable *y(t)*, and uses a correction based on proportional, integrated and derivatives. As a result, the controller will try to minimize the error over time by adjust a control variable *u(t)*.

2.8 Communication protocols

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

2.8.1 Serial

What characterizes serial communication is that it transmits one bit at a time, compared to parallel communication which sends several bits at a time. The two main types of serial protocols are based on Synchronous and Asynchronous communication [11].

Synchronous Serial has at least two wires, where one of them is a clock signal wire paired with a

data signal wire. This means all the devices follows the external clock and all the transfers will be based on this clock. It makes synchronizing the speeds easier and the protocol more straightforward. Some of the other synchronous protocols are I2C and Serial Peripheral Interface Bus (SPI).

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

2.8.2 TCP/IP

TCP stands for Transmission Control Protocol and is a connection oriented protocol that belongs to the transport layer in the OSI model. TCP is a standard that defines how to establish and maintain communication between devices, and transfers data by creating one fixed connection between two host units. After the connection is established, the devices can communicate continuously while the connection is active. TCP provides reliable communication, where TCP ensures that all data is delivered to the recipient and in properly order [29].

IP stands for Internet Protocol and belongs to the network layer in the OSI model. IP defines what, how, and in what way machines should communicate. Each host unit has its own unique IP address that identifies it from other devices in the Internet. This along with TCP defines how computers send packets of data to each other.

2.8.3 REST-API

A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST leverages less bandwidth, making it more suitable for internet usage. An API for a website is code that allows two software programs to communicate with each another. The API spells out the proper way for a developer to write a program requesting services

from an operating system or other application [52].

2.9 Cloud Computing

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. If the connection to the user is relatively close, it may be designated as an edge server [43].

2.10 Computer vision

Computer vision is a machines way of solving complex problems without human interaction. This is done by gaining visual information from digital images. All applications that use information gained from image processing in order to make decisions, uses computer vision [5].

2.10.1 Camera

Cameras are the sensors in computer vision, and are used for producing images for further processing. Various factors in the application affects which camera is best suited for image capturing. These factors is among other things, light, surrounding medium, available space and weight. Regarding camera types, they vary in features as resolution, image sensor, frame rate, shutting technique and the ability to capture color [41].

2.10.2 Data extraction

Last step of computer vision is extracting data from the images captured. This process is called digital image processing.

2.11 Image processing

Image processing consists of processing and extraction information from images. The use of image processing can be separated into two application areas: human vision applications and computer vision. What these application areas have in common is that they are both used for

image analysis figure 2.6. Human vision applications have humans as the end user, thus limiting the amount of information that can be extracted from the image to what is possible to see with the eye. With computer vision applications, the end user is a computer. Large amount of information is possible to extract, as a computer is capable of revealing almost the entire electromagnetic spectrum, while the human eye is limited to detecting only visible wavelengths. Another advantage computer vision has over human vision is the ability to neglect the features that are irrelevant for the task at hand [46].

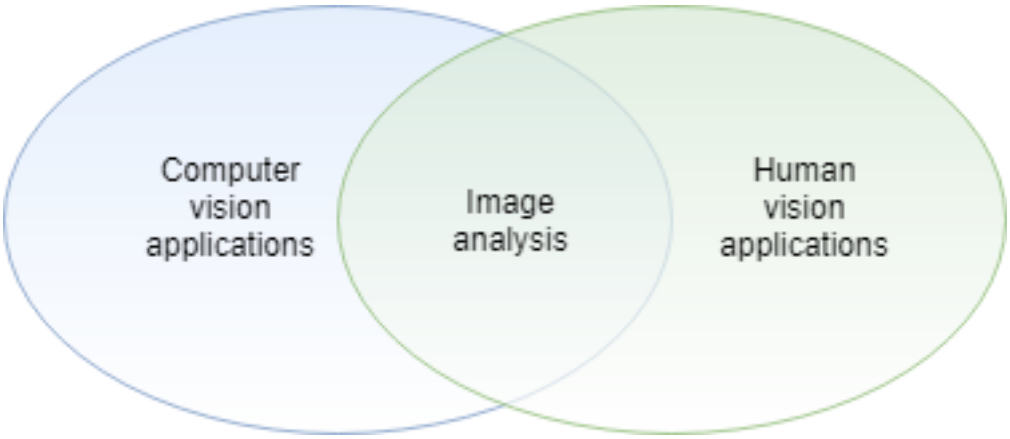


Figure 2.6: Image processing

2.11.1 Region of interest

A region of interest (ROI) is a portion of an image that you want to filter or perform some other operation on. The process usually consists of the removal of some of the peripheral areas of an image to remove extraneous trash from the picture, to improve its framing, to change the aspect ratio, or to accentuate or isolate the subject matter from its background [53].

2.11.2 Digital containers

A digital containers is a digital representation of physical containers. This is done by limiting the containers in software, and visualize them by drawing them in real time on frame witch is presented on the graphical user interface (GUI).

2.12 Machine learning

Machine learning is all about understanding and extracting knowledge from data. It is a researching field within Artificial Intelligence and has a lot of different fields branching out of it. The use of application helped by machine learning algorithm is to find in all our lives. All from picking out recommended music, videos, movies and other entertainment to unlocking our phones and make our work easier. The use of machine learning has a big influence on how data-driven research is done today. It helps science understand problems not necessarily are recognised by humans, such as finding particles, analysing DNA and recognising cancer [33].

2.12.1 Neural networks

Neural networks is inspired by our own human brain. The human brain contains roughly 86 billion neurons and the connections in between this neurons is what make our mind so powerful. This makes all, from controlling our body, thoughts, memories and more possible. The idea to use this model of neural network in computing is not a new research field. It was presented for over 60 years ago, but the technology at that time had no possibilities to apply such a model. In this section we are going to look close into how neural networks function and specially how the convolution neural networks function (CNN) works. The CNN is the computers equal to human eyes and is copying how shapes, color and shades are processed in our brain [19]. There are many kinds of neural networks and some of the easiest to imagine is a feed-forward neural network which contains the following parts:

- The input layer
- The hidden layers
- Neurons
- The output layer
- Activation function

2.12.2 Input layer

The input-layer works as a entrance to the hidden layers. The entrance is tailored for the data to fit to the neural network. The data often needs to be processed to fit into a input layer, and when the data is in the same size and dimensions as the input layer, the data is ready to be processed in the neural network. The hidden layers is what makes up the neural network, and is often called a "black box", however this is where the "magic" happens. In the hidden layer we are talking about numbers of hidden layers. This is the amount of layers with a decided amount of neurons in the depth of the network. In between the layers are connections between the neurons from the previous and to the next layer [19].

2.12.3 Neurons

Neurons are connected to the previous layer and receives values from the previous neurons with a weight and then pushed through a activation function. The value of the current neuron can be mathematically described like: $ActivationFunction(N(1-1l)*w1+N(2-1l))=N$ [19].

2.12.4 Activation function

As mentioned in the section above, the activation function is a function used to calculate the new value of a neuron. This is used to normalize data and keep the neurons under control. A neuron can often make conclusions by looking at noise (data that are misleading or confusing) and keep evolving around this misunderstanding. There are therefore a lot of different types of activation functions, and the most known of them are listed in the figure 2.7 [19]

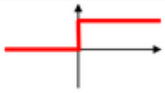
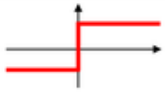
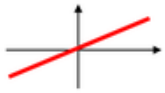
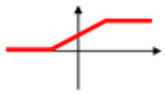
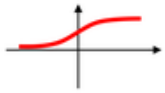
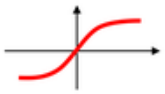
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Figure 2.7: Different sort of activation functions

2.12.5 Output layer

The output layer is similar to the input layer. It is the exit of the neural network and this is where you get your results. Therefor the output layer has to be in the same dimensions and size as the results you expect. This means that the labelled known training results should have the same size and dimensions as the output layer. So if you classifying a image to be either a "Circle" or a "Square" the output should be an approximation of what the network think it is e.g. [0.95, 0.02]. This is showing that this neural network think it is a "Circle".

2.12.6 Deep neural network

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutions neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, machine translation, medical image analysis, material inspection and many more, where they have produced results comparable to, and in some cases superior to human experts [19].

2.12.7 Dataset

Picking better data

To solve the problem, it is useful to pick the data that suits the problem best. This includes removing data that are not useful or generating/adding more data that we want and what supports our conclusion.

Pre-processing and scaling

In neural networks and convolutional neural networks, the algorithms are sensitive to the scale of the data. Therefore, it is a common practice to scale and pre-process the dataset before giving it to the network. The data should be adjusted in such a way that it suits and optimises the performance of the network. In a lot of cases the data is too detailed, the content is "irrelevant" for the problem that you are trying to solve, or just not fit for the network to function optimally. In this case, scale the data to fit the network by e.g. reducing dimensions (PCA), changing the data type, or fitting it into another shape that suits better. A very important thing to remember when you are changing the training data is that the test data should be pre-processed in the same way [14].

2.12.8 Multi-class classification

In a lot of cases it is enough to calculate single values like is it a circle in this image or not. However, in cases where there are more than two classes to classify, you get a problem that is an instance of multi-class classification. When this is the case and each single point should be classified into only one of the categories, meaning that one of the categories is 1 and rest is 0, you have a instance of single-label, multi-class classification [14].

2.12.9 One hot encoding

When classifying multiple classes or solving categorical variables (single-label, multi-class classification) such as the one mentioned in the chapter above. The most common way to represent the data is using the one-hot encoding, also known as categorical encoding. The reason behind

this is to simplify the variables into true-false, or 0-1 vales to describe what class it is representing [14].

2.12.10 Convolutional neural network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The figure 2.8 shows an example of a Convolutional neural network [30].

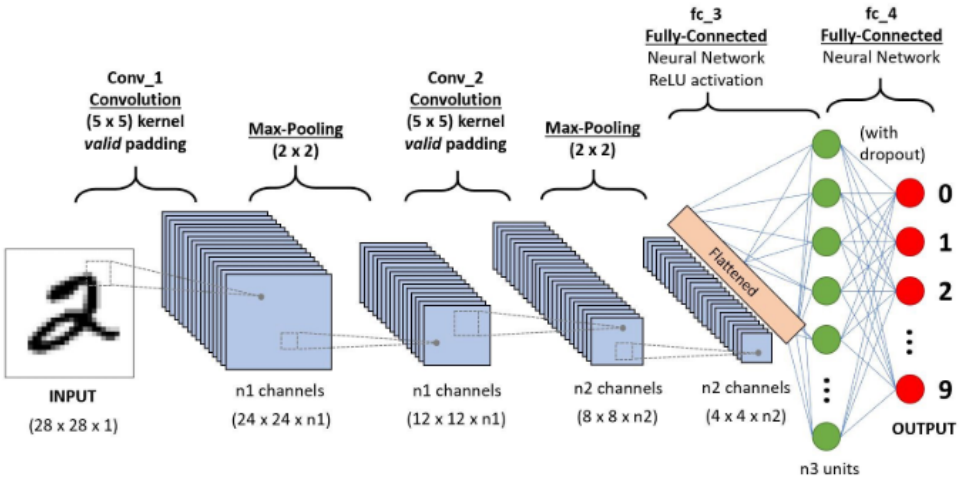


Figure 2.8: Convolutional neural network

2.12.11 Convolution layer

The convolution layer performs a 2D convolution that compares a squared filter over the entire images. The filter could represent an edge, line, dark spot or light spot, that scores the parts of the images to how similar that part of the image is to the filter. After applying the filters the convolution layer gives the result to the next layer [30].

The first convolution layer looks at simple filters, but the filters get more complex in the following layers and could represent entire objects instead of lines or spots in the end. In Figure 2.8 one can see how the different layers have different complexity in the filters and can recognize objects like faces and cars in the last layer [30].

2.12.12 Pooling layer

The convolution networks often includes local or global pooling layers. This down-samples the output and it uses for example max pooling or average pooling to define the down-sampled value. The pooling layer also makes the filter less sensitive to position which is good because you don't want to focus on an exact position, but what actually is appearing in the image [30].

2.12.13 Fully connected layer

The final layer in a CNN is named *Fully Connected Layer*. This is where every value that has gone through all the filters earlier gets a vote to find what the answer is going to be. We take all the values and list them up in a single fully connected table where it weights against all objects. The one object with the strongest average is what the CNN will return as its result [30].

2.12.14 Transfer learning

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. This area of research bears some relation to the long history of psychological literature on transfer of learning, although formal ties between the two fields are limited [32].

2.12.15 Training

Training a neural network typically consists of two phases:

- A forward phase, where the input is passed completely through the network. During the forward phase, each layer will cache any data (like inputs, intermediate values, etc) it is

later used in the backward phase. This means that any backward phase must be preceded by a corresponding forward phase.

- Backward phase, is where gradients are back-propagated (backprop) and weights are updated. During the backward phase, each layer will receive a gradient and also return a gradient. It will receive the gradient of loss with respect to its outputs $\frac{\partial L}{\partial \text{out}}$ and return the gradient of loss with respect to its inputs $\frac{\partial L}{\partial \text{in}}$.

2.13 Real time programming

Real-time programs must guarantee response within specified time constraints, often referred to as "deadlines". The correctness of these types of systems depends on their temporal aspects as well as their functional aspects. Real-time responses are often understood to be in the order of milliseconds, and sometimes microseconds. A system not specified as operating in real time cannot usually guarantee a response within any timeframe, although typical or expected response times may be given [4].

2.13.1 Threads

Threads are a way for a program to divide (termed "split") itself into two or more simultaneously (or pseudo-simultaneously) running tasks. Threads and processes differ from one operating system to another but, in general, a thread is contained inside a process and different threads in the same process share same resources while different processes in the same multitasking operating system do not. Threads are lightweight, in terms of the system resources they consume, as compared with processes [4].

2.13.2 Thread safety

When using threads, it is important to take care that no more than one thread operates on a variable at the same time. A program that addresses all issues related to multiple threads, is called thread safe. There are mainly four scenarios that can cause problems. These are "Race condition", "deadlock", "livelock" and "resource hunger".

A "Racecondition" is returned to, or multiple objects to be written to the same variable simultaneous. "Deadlock" is when multiple tasks await a common resource they want to get access to, but another task is currently using the resource, which makes it unusable for the given moment. "Livelock" is when tasks change state because of the other. "Resource hunger" is when no one get the resources they need [4].

2.13.3 Executor & Scheduler

Executor is used to manage threads in a program. When using an executor, one can decide when to do a task. This is done by calling the objects run() - method. When you perform a task, the executor will get a thread from a ThreadPool and assign it to Runnable task. If there are no threads available, it will wait in a queue until a thread is free. You can create a Runnable object and send it to the executor who will manage the thread handling. Threads managed by the executor are thread-safe as long as the methods for the tasks are implemented correctly [4].

Assignments can be scheduled to run once or periodically. To do this, use one scheduler that can be part of the executor. When using a scheduler you can say that a particular task should run every 100ms, or run them after 10 minutes, and then never again [4].

2.13.4 Concurrency

The principle behind "Concurrency" is to run multiple tasks of a program seemingly simultaneously. Each task will be assigned a thread, where the scheduler or executor selects which thread that will run. If you have a single cored processor executor it will switch between threads either by time or by events. When using concurrency in a program one can optimize CPU usage by taking full advantage of the speed and cores of the processor [4].

2.13.5 Synchronized

Using threads can cause problems when sharing data. If two or more threads read and write to the same variable, one must coordinate and synchronize access. The method of synchronization can be used either on some methods in an object, or on the whole object itself. When using

synchronized, only one thread has access method at a time, if there are several who try to write or read to it, they must wait until the thread having the lock is finished. "wait()" and "notifyAll()" are methods used to communicate between threads when sharing data [4].

2.13.6 Event

An event is used to signal from one class to another when a desired state is reached. For example, when new data is available to be read. The event class usually uses predefined variables and states, such as "UP" and "DOWN". The class has some methods used, such as; "Set()" which sets the state to UP, "reset() "to set the state to DOWN," toggle() "will change the state, and "wait()". The "wait()" method is used by tasks that are waiting for a specific state [4].

2.13.7 Event Listener

Event Listeners are used to trigger an event when an input is set. This may be when you type on a keyboard. Each keystroke triggers an event from the event listener. The listener then triggers an event that will start a task. When you have an event, the listener is an important feature so it needs to respond quickly and whether it needs to start a long task, it should be done in another separate thread [4].

2.14 Programming Language

A programming language is a formal language, which comprises a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms and structures.

2.14.1 Java

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to

bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture [48].

2.14.2 Python

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library [51].

2.14.3 C++

C++ is a general-purpose programming language, and modern C++ has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation.

C++ was designed with a bias toward system programming and embedded, resource-constrained software and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, and performance-critical applications [44].

2.14.4 JavaScript

JavaScript often abbreviated as JS, is a high-level, interpreted scripting language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it,

and major web browsers have a dedicated JavaScript engine to execute it .

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. [49].

2.14.5 HTML and CSS

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document [45].

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content [40].

Chapter 3

Approach

Chapter three is a subjective project approach. It gives an understanding of what has been important to the project regarding desired quality, requirements and specifications. This includes group organization, selection of robot, motion solution, neural network solution, and the data flow.

3.1 Project Approach

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

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

- Make a detailed concept drawing of the total system. It contains kinematic, measure-

ments and a basic design. This will give all the team members an overview of task needed to be done and a basic idea of what the result should be in the end.

- Early start on designing the prototype. The robot will be designed using 3D CAD. One of our members did already have most of the most critical parts at hand. Other minor parts will be ordered early in the project. Consequently, we can create parts needed to be manufactured while waiting for the parts ordered.

- Programming will start at once. We will start making the program structure, and then proceed to programming the communication part and at last the robot motion. Since a lot of the parts will be 3D printed which takes time, the production of these parts will be done along the programming.

- Assembling the robot will be started simultaneous with the programming since crucial parts are already at hand.

- Testing will be performed when the robot is assembled and the necessary programs are completed. All objectives will be tested separately before tested combined.

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

3.2 Robot

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

3.2.1 Robot selection basis

The robot needs to be designed in a way suited for the facilities. This means that the robot needs to be as space efficient as possible. Also since there can be people working with other task in the same facilities, the robot needs to be designed with low risk of harming them.

For making a robot, one or more methods providing motion is needed. An industrial robot has

to possess at least three degrees of freedom as mentioned in chapter 2.1. Thus the robot needs to perform at least three motions. When choosing the method for creating motion there are two criteria that needs to be met. First criteria being accuracy for operation. Second criteria being the working space. For sorting small objects, the accuracy of the robot needs to be high enough to position itself at the center of the object.

For generating motion, some kind of actuator is required. To find fitting actuators, it is important to find one that deliver within our system requirements. When an actuator has been selected, a fitting driver is necessary for operating it.

3.2.2 Object sorting

The robots main task is to pick and sort different objects within the workspace. Since the operation can be on a conveyor belt that is moving, it is important to have accurate and fast actions. Therefor the robot needs to be able to operate at high speeds while maintaining accuracy.

In order to sort different objects, the robot must be able to locate it, and for this computer vision with convolution neural network can be used. The convolution neural network needs to be capable of detect and classify each object with the challenges provided by the varying lightning conditions, see chapter 2.12.10 for more information.

After the convolution neural network has found one or more objects, it needs to be able to pick it as fast and accurate as possible. The most critical part of picking is to accurately locate the center to avoid any displacements while sorting. For that reason, the method used for detection and classification is done remote on a high end computer.

The robot do also have a way too be limited to only one object type, and therefore excluding other types. Then by cooperating more robots, a larger quantity can be picked and sorted.

3.3 Review

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

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

3.4 Testing

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

3.4.1 Image processing testing

The image processing and manipulation techniques developed, needs to be tested in order to verify that the *digital containers* and boxing, labeling objects are placed in the correct place. By measuring multiple known positions the correct position can be compared to the digital bounding boxes and containers.

3.4.2 Convolution neural network testing

Multiple convolution neural networks were tested. The CNN has to deliver an extremely high accuracy while running fast. Therefor multiple different CNN were trained and evaluated to

find the correct CNN which had the performance metrics needed for our system.

3.4.3 Communication testing

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

3.4.4 Suction testing

The suction method has to be tested to verify that it suits the implementation. A criteria is considered important when testing and choosing the suction method and technique. This being that the suction provides enough vacuum to pick up and hold objects with rapid acceleration.

3.4.5 Electrical testing

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

3.4.6 Robot testing

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

- Move to a specific position
- Stop if it moves outside of its workspace

- Detect objects
- Classify objects
- Pick and sort sequence

Chapter 4

Materials

This chapter gives insight and explanation of choices that have been made. We are taking a stand in which type of robot to be built, and why. The components that were chosen will also be elaborated.

4.1 Robot selection

Since this project task is mainly focused on real-time software implementation, the selection of robot-type was limited to what the school had stored. However the Cartesian robot is an excellent choice for the problem since its able to operate in three degrees of freedom, as all joints are independent of each other. It is also able to reach a large volume relative to the area possessed.

This robot gives us an effective workspace compared to the usage of space. The Cartesian robot chosen, has one pneumatic joint moving vertically along the z-axis, and two prismatic joint moving horizontal, where one is for the x-axis and the other for the y-axis. This setup will provide three degrees of freedom in a Cartesian coordinate system. An illustration of the robots kinematics can be found in figure 4.1.

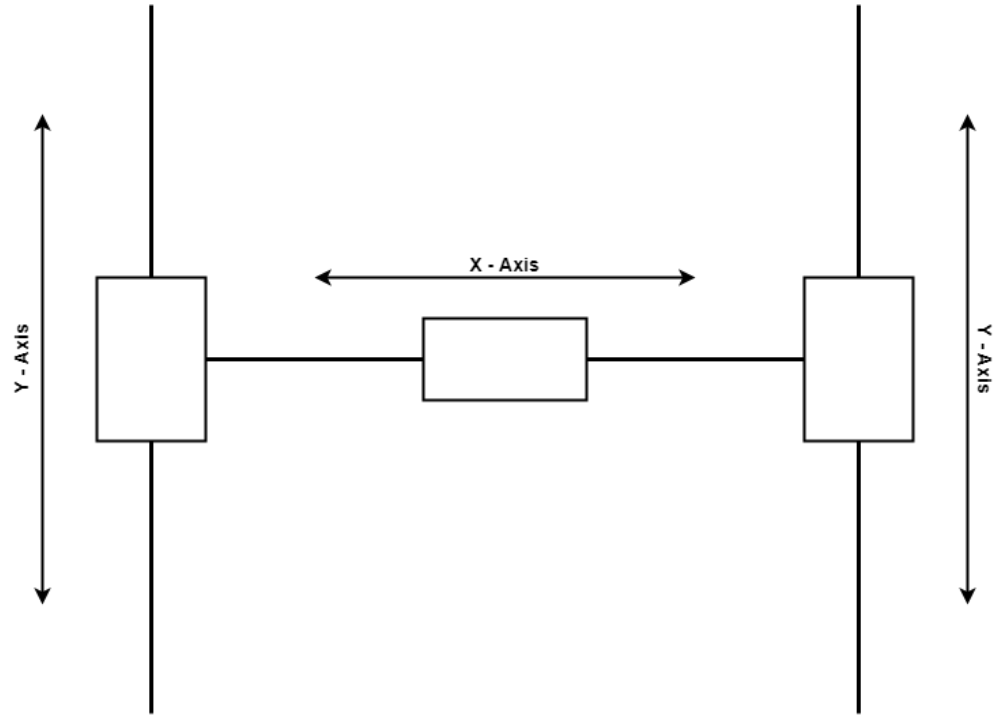


Figure 4.1: Cartesian robot joints

4.2 Motion method selection

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

Our choice for creating linear motion fell on the belt driven actuator, as this was already imple-

mented on the frame received from the school. For the vertical motion the ball screw actuator was considered, but as the objects are light weight and we need a fast acting motion, our choice landed on using a pneumatic cylinder.

4.3 Motor selection

As we chose belt drive as motion method, we were in need of actuators that could run the belts. Rotary motors were the obvious choice of actuator, as motion of the belts are generated by rotating the pulleys. While researching for the choice of rotary motors, two types of motors stood out. The first option was the stepper motor. Stepper motors are more suited for running at low speeds, and is not bounded to run a closed loop with encoders. A drawback considering stepper motors is the speed and torque characteristics. When the speed goes up, the torque is reduced drastically. The second option was the brushless motor. Burshless motors are made for high torque, high accuracy and high speed, and in combination with a encoder, we get a closed loop for position, velocity and acceleration feedback.

With the above taken into consideration, our choice fell on the brushless motor as actuator for our system. Burshless motor in combination with an encoder, will provide high accuracy at any speed.

4.4 Motor driver selection

When it came to motor drives, there were two categories to choose from. Servo drives for industrial use, and drivers for hobby use. In the hobby category, only the ODrive from ODrive Robotic satisfied the demands. It offers a dual channel configuration with Field Oriented Control (FOC).

FOC is a control method for controlling torque of 3-phase motors and stepper motors that offers high accuracy. To achieve this, it uses two components where one is in phase with the motors current flux and the other is orthogonal to the first and proportional to the motor's torque. In FOC, with having two PI controllers, the components can be regulated separately. In addition,

by adding speed and positional control loops, a fully functional servo control for positioning applications is accomplish[36].

The industrial servo drives are superior in quality, speed and precision positioning control, but it comes at a steep price. In comparison, the ODrive is more than enough for our application, and comes at a much smaller price. Servo motors are also heavier and bigger in size, making it less attractive for our lightweight robot construction. With this in mind we chose the ODrive motor driver.

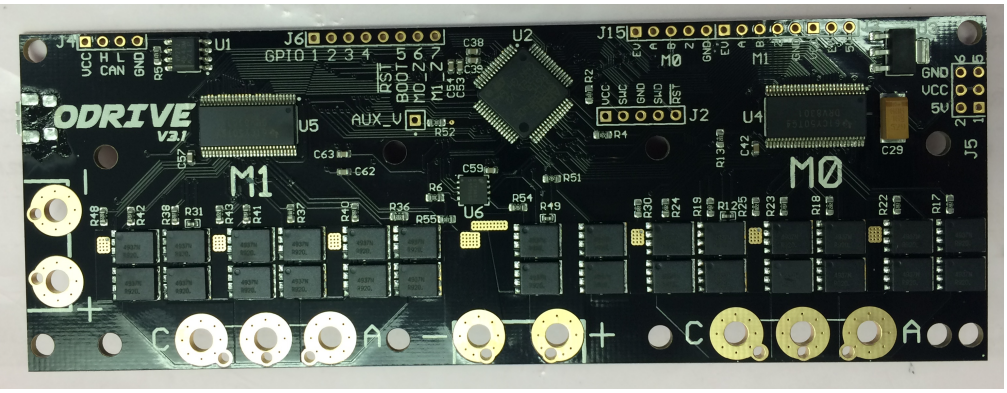


Figure 4.2: Odrive V3.6

4.4.1 Motors for horizontal motion

One of the motors are situated on the moving gantry, so weight was considered when choosing motors. Since the robot's task is to move quickly between locations it means the motors torque at lower speeds should be high.

With this in mind, we choose the ODrive Dual Shaft 270kv model. It offers nearly 2 Nm torque with nominal speed up to 8640 rpm. The acceleration is also enough as it only needs about 124 ms to reach its base speed [22]. It weighs in at only 500 gram, and that is including its enclosure with the encoder mounted.

Figure 4.3: Brushless motor

4.4.2 Cylinder for vertical motion

For the vertical motion we choose to use a pneumatic cylinder. The requirements for the cylinder are that it has fitting stroke length, and strength while also not being too big for the provided space, or too heavy for the rapidly moving endpoint.

The choice fell on a cylinder the school already had at hand. It was light weight and easy to mount on the robot. With a vertical travelling distance of 40mm when mounted it gives the robot enough clearance so it can pass over an object while holding another.

It also offers a bidirectional way of controlling and a t-slot design that can accommodate sensors for positional feedback. This was however not wanted as the complexity of the mount would increase as well as having a larger footprint on the already confined space. The cylinder is controlled via a 5/2 ported bistable pneumatic valve with a 24 VDC solenoid.

4.5 Choice of sensors

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

4.5.1 Mechanical endstop

Micro switches figure 4.4 were chosen for detecting the outermost points the robot can reach in x- and y-direction. Easy to install, minimum amount of wires and space made these the choice for sensors for the horizontal system.

Figure 4.4: Microswitch

4.5.2 Incremental rotary encoders

Photoelectric incremental rotary encoders were chosen for safely controlling the motion of the horizontal system. These encoders have a resolution of 8192 pulses per revolution and can be seen in figure 4.5.

Figure 4.5: Odrive encoder

4.6 Choice of controllers

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

4.6.1 Jetson Nano

The Jetson Nano will operate as our main controller. It runs on Linux and provide good support of different coding languages and compatibility for realtime operations, which is a critical part in our system. It also holds integrated options for USB, ethernet and GPIO pins for connecting hardware. The figure 4.6 shows a Jetson Nano.

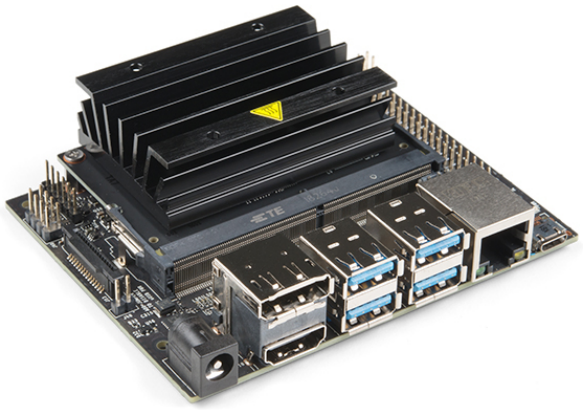


Figure 4.6: Jetson Nano

4.6.2 Teensy 4.0

Even though the Jetson Nano is a good choice for us, it lacks inputs and outputs (I/O). Therefore it was necessary to run a microcontroller alongside for handling input and output interaction. The microcontroller chosen was Teensy 4.0 for its low cost, and impressive performance. A Teensy can be seen in figure 4.7.

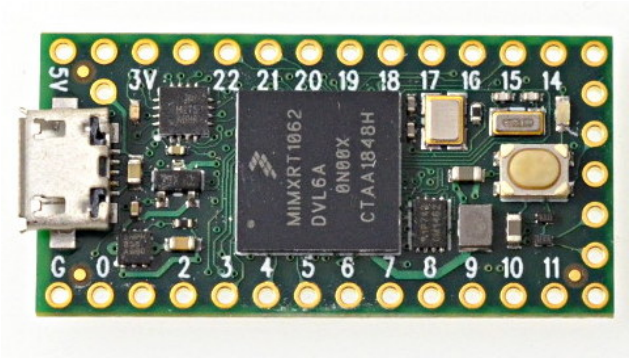


Figure 4.7: Teensy 4.0

4.7 Communication protocol

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

TCP/IP was the chosen protocol for the communication between the controllers remotely. This was chosen because its a standardized communication protocol, allows cross-platform and is a scalable, client-server architecture. This allows networks to be added without disrupting the current services.

4.8 Remote computer

The remote computer used for the convolution neural network to detection and classification is a "gaming computer" from a member of the group. The main components is:

- **CPU Intel i7-7700K** - 4.5 GHz clock speed. 4 cores and 8 Threads
- **GPU GeForce GTX 1080 Ti** - 11 GB memory and 3584 NVIDIA CUDA Cores
- **RAM Corsair Vengeance** - DDR4 3200Mhz 24GB

4.9 Camera

The camera chosen to use for the image capturing, is a Logitech C920 HD PRO web-camera figure 4.8. Logitech C920 is a Full-HD web-camera with 1920 x 1080 pixels and 78°field of view. The cameras high resolution and field of view makes it very convenient for capturing detailed pictures.

Figure 4.8: Logitech C920 HD PRO

4.10 Construction

As mention above, we were given the frame for our main construction. Furthermore, the rest of the construction consists of aluminium profiles, and self produced parts was 3D printed in plastic or cut in Plexiglas for the component housing.

4.11 Electrical components

The components used in the electrical part of the project were mainly composed of what could be gathered from the lab or bought online. How the components where connected, and how they where used can be seen in Appendix A.2.

4.11.1 Power supplies

When designing an electrical system, the supplies has to be chosen from the required voltages and power consumption of the electrical components in the system. In this system there is a range of different voltages. Because of this, multiple power supplies with different voltages are needed or voltage converters. Voltage converters can be single converter components or integrated circuits. To choose supplies we had to look at the consumption of the different voltage levels. The following currents in table 4.1 were calculated from RMS and is a theoretical maximum estimate of the power consumption of all the components combined. The idle or regular consumption will be lower than this estimate.

Power usage		
Component	Voltage	Ampere
Jetson Nano	5V	2.5A
Odrive V3.6	24V	100mA
Teensy 4.0	5V	100mA
Odrive 270kv	24V	5A
Odrive 270kv	24V	5A

Table 4.1: Power usage

The chosen supplies are the ones which were available to us from the lab inventory and delivered enough power. The supplies rating and their respective voltage rails can be seen in table 4.2.

Voltage	Ampere	Supply rating
5V	2.5A	2.5A
24V	7A	10A
5V	100mA	900mA

Table 4.2: Power supplies rating

4.12 Software and libraries

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

4.12.1 Software

- **JetBrains Intellij IDEA** - Is a free software development tool. It was used for developing Java software, which is its main supported language[10].
- **Overleaf** - Free web-based tool for writing in LaTeX. Some of its features are spell checking and cloud-based storing. The cloud-based storage makes real-time editing possible and thereof its possible to cooperate in the same files[24].
- **Arduino IDE** - Free environment for programming and interacting with the arduino microcontrollers. Released and made by the arduino corporation[1].

- **PCSchematic Automation** - Electrical CAD software. Has been used to draw all electrical schematics and our electrical layout diagrams[26].
- **Fusion 360** - 3D designing software[2].
- **Cura** - Slice STL files and generates G-code for 3D printing[37].
- **Draw.io** - For making simple diagrams and illustrations[6].
- **Visual Studio Code** - Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring[18].
- **Odrive Tool** - The ODrive Tool is the accompanying PC program for the ODrive. It's main purpose is to provide an interactive shell to control the device manually[21].

4.12.2 Libraries

The following libraries has been used in combination with Intellij IDEA, Visual Studio Code and Arduino IDE.

Java libraries

- **org.json** for parsing, decoding and processing of JSON objects [28].
- **SerialPort** for connecting and communicating over RX/TX [8].
- **JXInput** is an API that allows applications to receive input from the Xbox Controller for Windows. Controller rumble effects and voice input and output are supported [34].

Python libraries

- **OpenCV** is a library of programming functions mainly aimed at real-time computer vision [23].

- **Numpy** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays [20].

- **TensorFlow** is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks [35].

- **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries [27].

- **Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+ [16].

- **Keras** is an open-source neural-network library written in Python. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible [13].

C++ libraries

- **ArduinoJSON** is a C++ JSON library for Arduino and IoT (Internet Of Things) [3].

- **ODriveArduino** is a library used to operate the ODrive motor controller [38].

JavaScript libraries

- **AJAX** stands for Asynchronous JavaScript And XML. In a nutshell, it is the use of the *XMLHttpRequest* object to communicate with servers [12].

Chapter 5

Design

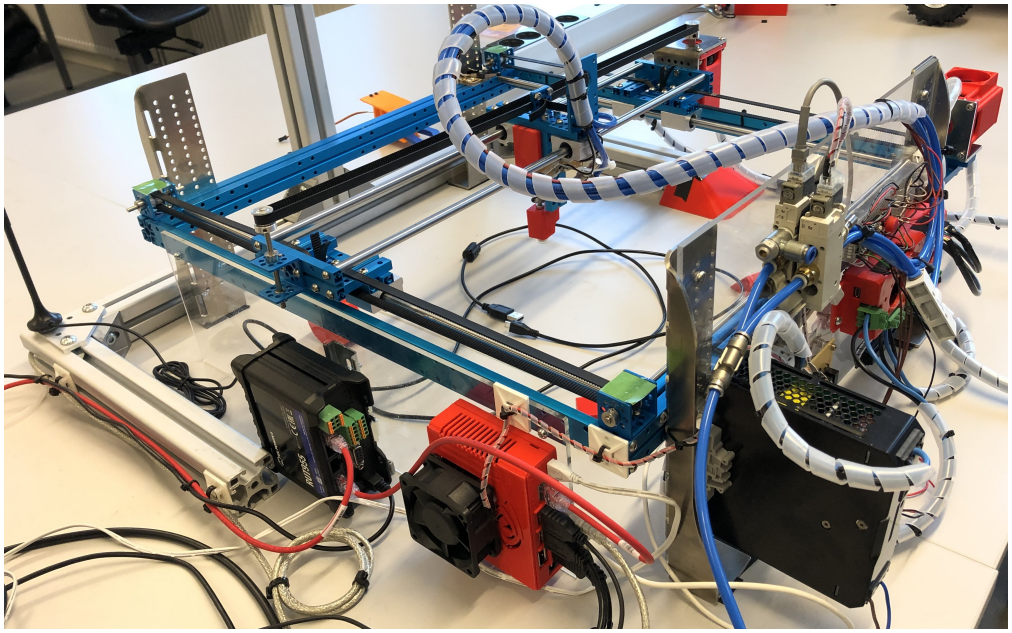


Figure 5.1: Pick and Sort Robot

This chapter explains how the most essential parts of the robot is designed and manufactured. All constructed parts are designed using Autodesk Fusion360. Custom parts were manufactured in the Tunglab at NTNU campus Aalesund. The Tunglab has 3D-printers and a CNC laser cutter.

5.1 Main frame

Figure 5.2: Original box

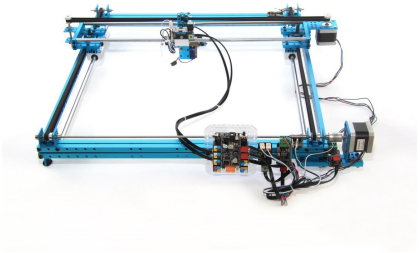


Figure 5.3: Original design

The main frame is further developed from a "XY Plotter Robot Kit" from the makeblock producers. The original box and design can be seen in the figure 5.2 and 5.3. The point of this machine is to either draw/write something using a pen, or it can be used as a laser engraver with the "Laser Engraver Upgrade Pack"[15].

5.1.1 Designed parts

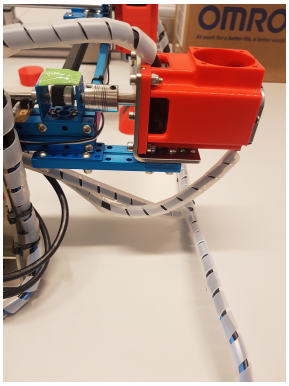


Figure 5.4: Motor for Y-axis

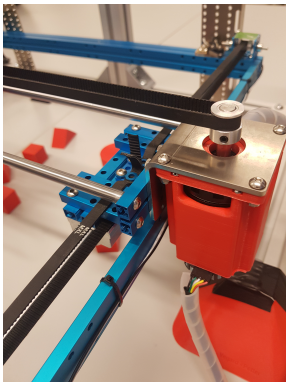


Figure 5.5: Motor for Y-axis

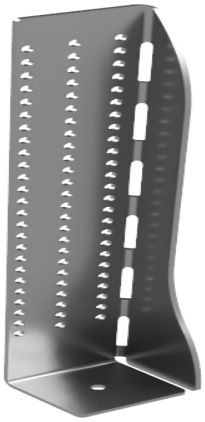


Figure 5.6: Robot foot

There are several designed parts in this projected. Some are designed by the group and others are 3D-printer templates. The first printer templates that were used was the housing for the Odrive and Jetson Nano. Further the housing for the motors was printed, this can be seen in figures 5.4 and 5.5. The group then design new legs in steel for the machine to get it taller as

shown in figure 5.6. Lastly the group design the cylinder holder and the holder for the suction cup, this is better explained in the chapter 5.2.2

5.2 System

5.2.1 Electrical layout

The electrical layout is shown at the bottom of the figure 5.1. A Plexiglas is attached vertical to the legs of the robot to create a foundation for the electrical equipment. The equipment are then attached on this plate with different methods, like glue or a DIN-shine. On the left part of figure 5.1 one can see the Jetson Nano and the router attached to another Plexiglas.

5.2.2 Festo vacuum generator

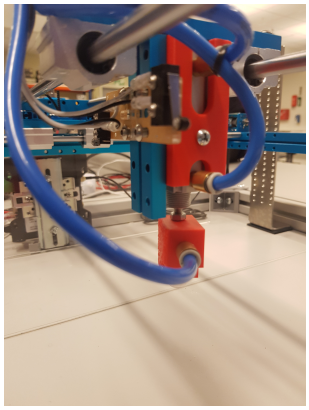


Figure 5.7: Cylinder with suction cup

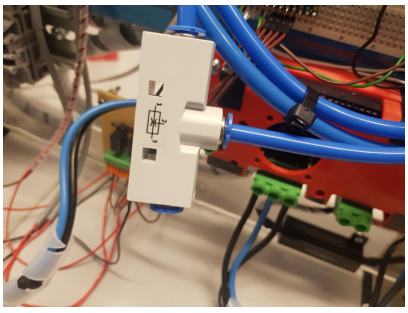


Figure 5.8: Vacuum generator

Figure 5.9: Designed parts

The overall design is shown in figure 5.9. This can be divided in to two parts. First the holder for the cylinder in the top section of the figure. The second part is the holder for the suction cup and its design is shown in the lower part of figure 5.9. The finished part put together are shown in figure 5.7. Figure 5.8 is showing how the vacuum is generated, and how this works is explained in chapter 2.6.

Chapter 6

Implementation

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

6.1 System overview

Figure 6.1 introduces the system in its entirety. It consists of several systems:

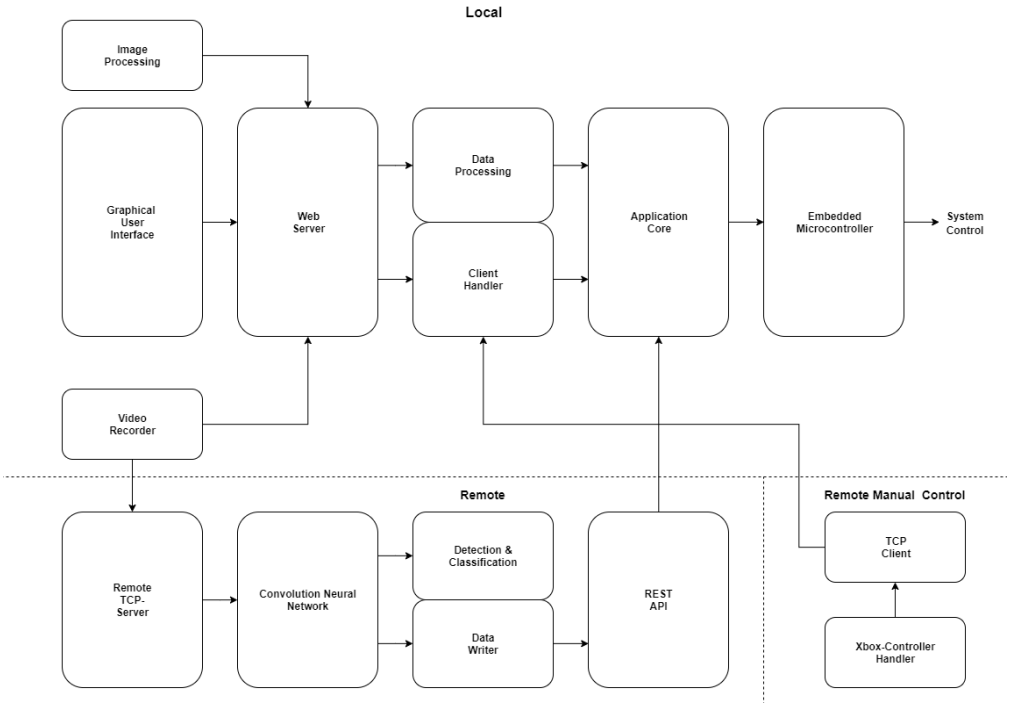


Figure 6.1: System overview

6.1.1 Graphical user interface (GUI)

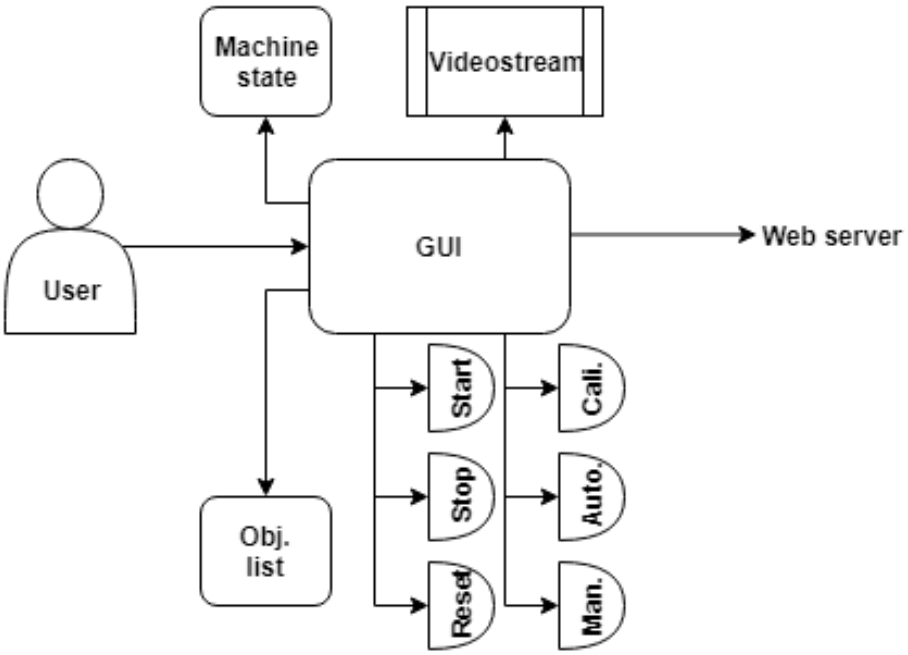


Figure 6.2: Graphical user interface

Graphical user interface figure 6.2 is a platform that interacts with the user. It provides the user with information from the video feed, state of the robot, and in addition the opportunity to calibrate, manual override, reset, start and stop the robot.

The GUI consists of different components that works concurrent with each other. The user experience would not be great, if for instance everything had the same update rate as the videofeed. Therefore has each component an internal update rate, and all the buttons are asynchronous for instant feedback to the system.

```
// Pick-And-Sort-Robot/remote/python/static/js/index.js
setInterval(function() {
  $("#state").load("/state");
}, 100); // In millis
$(function() {
        $('a#stop').bind('click', function() {
          $.getJSON('/start',
              function(data) {
```

```
        });
    return false;
});
```

6.1.2 Web server

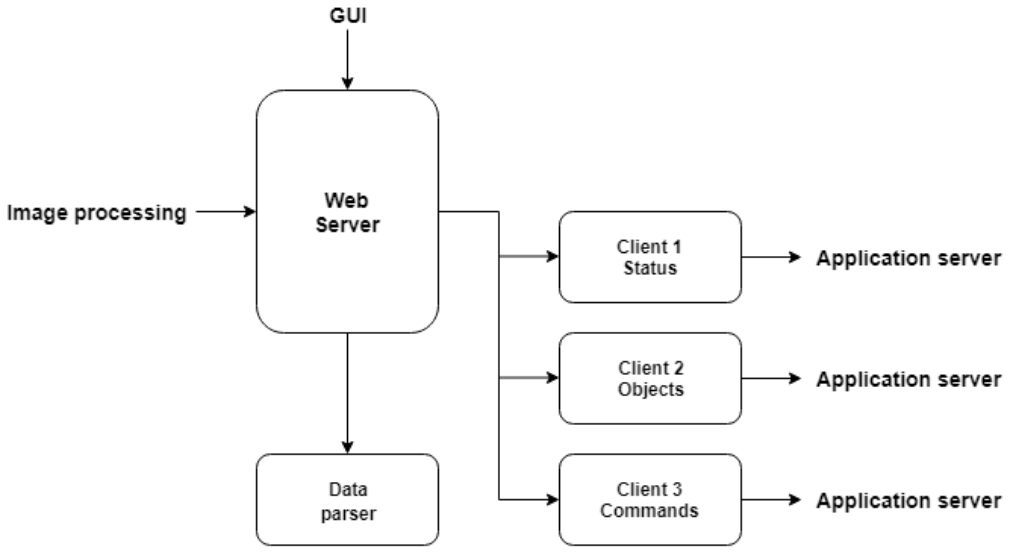


Figure 6.3: Web server

The web server figure 6.3 is the backbone of the GUI. All of the components communicates with the web server to handle the tasks. This means the web server has to communicate with the application server too get information from the database. This is done by creating two clients that *subscribe* on a topic from the database and pull information continuously within a given time-frame.

However all of the button presses from the GUI are handled by a single client. This client will only run periodically, and sends only data when a button is pressed. There is to be noted that all clients run in separate threads, and collaborates with the web server. This leads to better performance and the system can handle an "unlimited" amount of users on the front end, since the asynchronous calls and computations happen on client side, and they all fetch from same source as the three clients supply.

6.1.3 Video capture

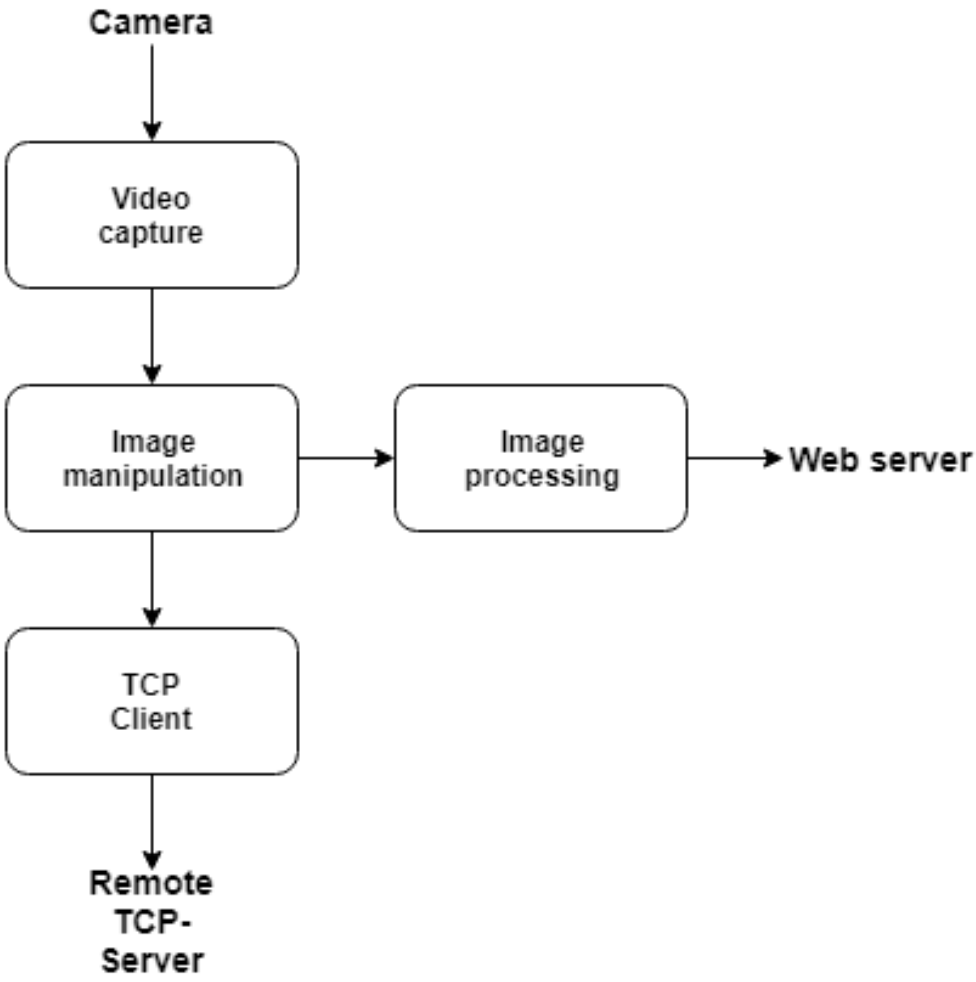


Figure 6.4: Video capture

The video capture system handles input from the web-camera attached to the robot. It does also work on the received input, processing it and manipulate it to fit it needs. This is done so the data sent to the remote server are optimized, sending only the necessary amount of data.

```
# Pick-And-Sort-Robot/remote/python/src/utils/object_detector.py

def send(self):
    """Convert and send frame as bytes."""
    self.frame = self.video_camera.run()
    # Create a region of interest that excludes the digital containers
    roi = self.frame[150: 480, 0: 460]
```

```
    data = pickle.dumps(roi)
    message_size = struct.pack("=L", len(data))
    self.write(message_size + data)
    _, jpeg = cv2.imencode('.jpg', roi)
    return jpeg.tobytes()
```

Concurrent with sending data to the remote server, it splits the image into two, and handle the other part separately too make it ready for the web server. This process draws the *digital containers* and uses data received from the *CNN* too accurately draw bounding boxes and labels on objects.

```
# Pick-And-Sort-Robot/remote/python/src/utils/visual.py

def draw_circles(self, frame, objects):
    """Draws circles around objects on frame."""
    result = frame
    for obj in objects:
        center_coordinates = (obj['x'], obj['y'])
        result = cv2.circle(result, center_coordinates,
                 self.radius, self.shapecolor, self.circlethickness)
        cv2.putText(result, obj['type'], center_coordinates,
                    self.font, self.fontscale*0.8, self.fontcolor, 1)
    return result
```

6.1.4 Remote computing

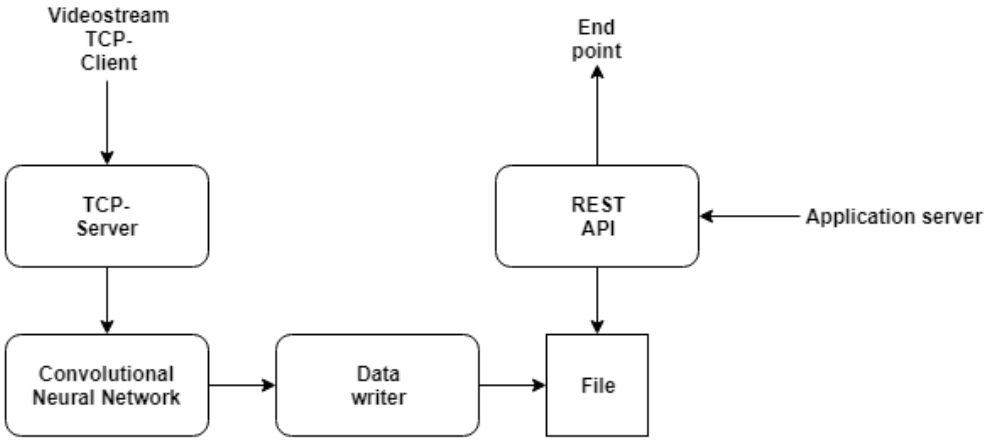


Figure 6.5: Remote computing

The remote server receives data from the video capture client continuously see figure 6.5. The data is decoded from bytes to an array again, and then served too the CNN model for further computation.

```
# Pick-And-Sort-Robot/remote/python/src/utils/server.py

    def get_frame(self):
        """Recieves frames as incoming bytestreams.
        Returns videoframe from stream."""
        while len(self.data) < self.payload_size:
            self.data += self.connection.recv(4096)

        packed_msg_size = self.data[:self.payload_size]
        self.data = self.data[self.payload_size:]
        msg_size = struct.unpack("=L", packed_msg_size)[0]

        while len(self.data) < msg_size:
            self.data += self.connection.recv(4096)

        frame_data = self.data[:msg_size]
        self.data = self.data[msg_size:]
        frame = pickle.loads(frame_data, encoding='latin1')
```

```
        return frame
```

The *CNN* will then give each object an id, center point, probability, and label. While this is happening another thread will write this information over too a text file as *JSON* format.
The text file will here work as a shared resource for multiple processes. The reason for this is that the *REST-API* reads the stored data and parses it as *JSON* before publishing it at an end-point where anyone that *subscribes* on the end-point will get the latest data.

```
# Pick-And-Sort-Robot/remote/python/src/end_point.py

# Endpoint to get all objects
@app.route('/', methods=["GET"])
def index():
    """Returns all found objects as JSON."""
    content = objects.read()
    return jsonify(content)
```

6.1.5 Core application

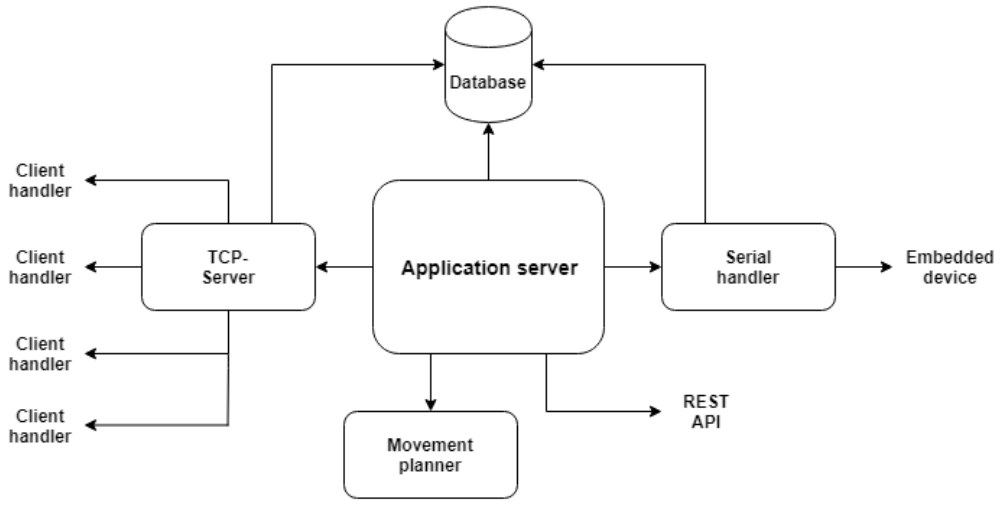


Figure 6.6: Core application

The Core application is where everything connects and the actions are managed. As shown in Figure 6.6, the core exists of different components that mainly takes care of several communication protocols and parsing data. The first and main protocol is the TCP - Server that secures

connection to the clients by a socket. Where as each client get a designated thread.

The second protocol is Serial RX/TX. The SerialHandler will search for a connection every 2000 milliseconds and need a connection established before proceeding to read or write any data. It implements *SerialPortEventListener* for incoming data, and sends data out on request.

The last protocol is the REST/API that fetches data from the remote stored textfile. It sends a GET request to the remote server, which in return reads from the stored textfile and sends the data back as JSON.

Besides that, does the core also handle movement planning based on the current configuration given in the graphical user interface, and what is available to be picked.

As stated earlier in *6.1.2 Web server* there are multiple clients connected to the *TCP - Server*. They are within the server handled each separately by starting a thread for each client, handling all their requests concurrently. This leads to a more responsive and optimized communication between the two systems.

6.1.6 Remote controller

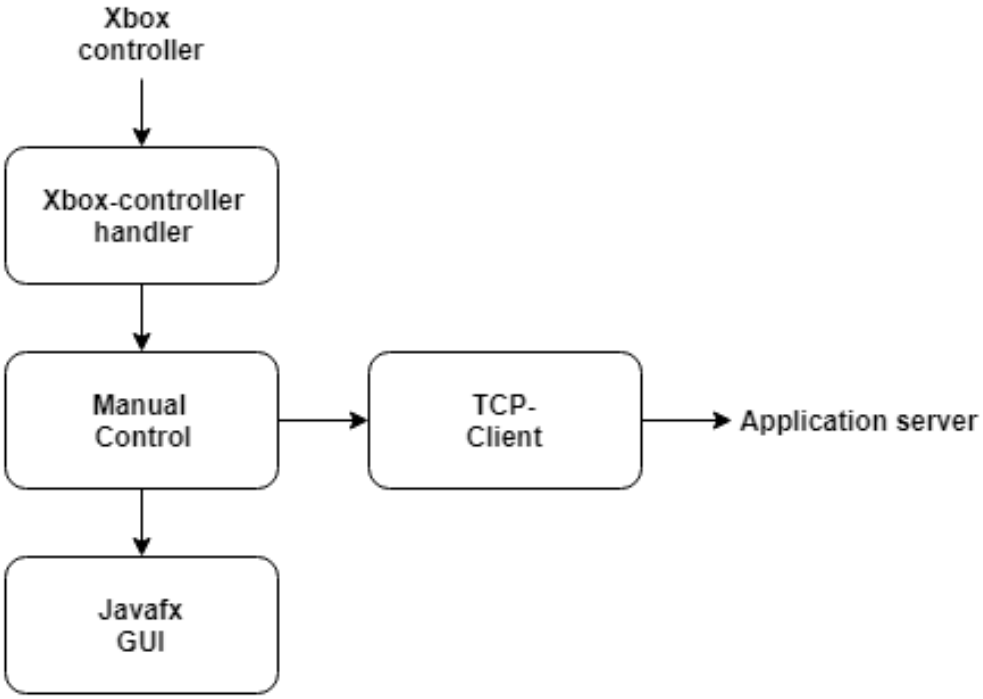


Figure 6.7: Remote controller

A remote controller application was developed so the robot could be controller manually. It can be helpful for speeding up the process of testing and debugging, and it also enables the user to move the robot if anything out of the ordinary should happen (e.g. in case of robot getting stuck. (this is more relevant for bigger robots that cannot be moved by hand)). The remote controller, when the robot is in manual mode, can control both x and y axes with a joystick, and the buttons can be set to execute different routines such as picking and placing objects.
In addition to the robot's integrated limit switches, safety can also be maintained by stopping all motion should the user release a dead man switch located on the remote controller.

A Class named XboxController, which manages the flow of information from the remote controller, implements a Runnable interface allowing it to be executed in an executor service. The ScheduledThreadPoolExecutor service is used to run the remote controller at a fixed rate. By setting the fixed rate at 100 ms, a maximum number of five clicks per button can be detected. Setting the time lower will decrease the response time even further.

By using a listener interface given in the XboxController constructor, we achieve a low coupling between the XboxController class and the side that receives data, e.i. the subscriber.

Data is given to the subscriber in JSON format stored in a StorageBox class. The storage box uses a producer-consumer implementation for safely exchanging information between multiple threads. This is done by using synchronized methods as well as not reading content before it becomes available. By also not allowing new content to be placed before the old is read, certainty that no content gets lost is given.
A separate TCP Client, running in the same executor service, connects to the server. When connected, a client.send() method is used for sending the JSON formatted string message. Before sending, the client appends a unique header, making the server able to parse the incoming message properly.

The figure 6.8 show the GUI that is used for connecting the controller to the robot.

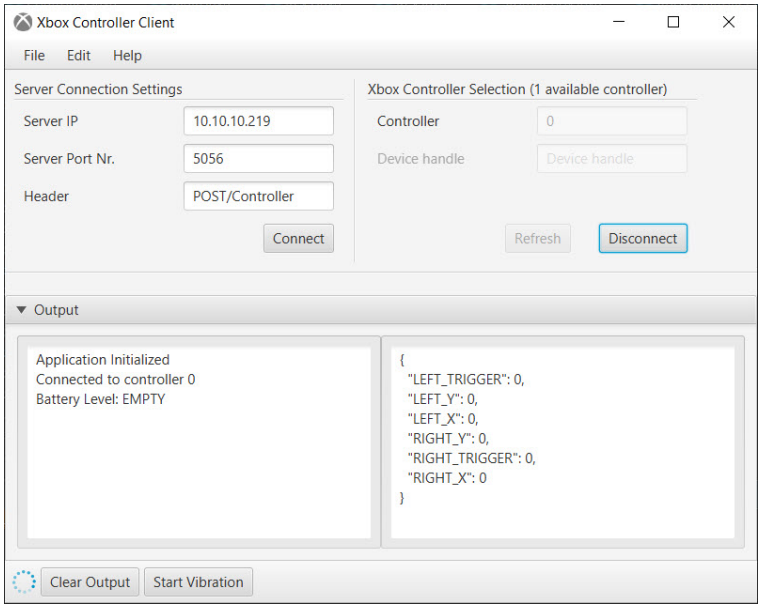


Figure 6.8: Xbox Controller Client GUI

6.1.7 Embedded state machine

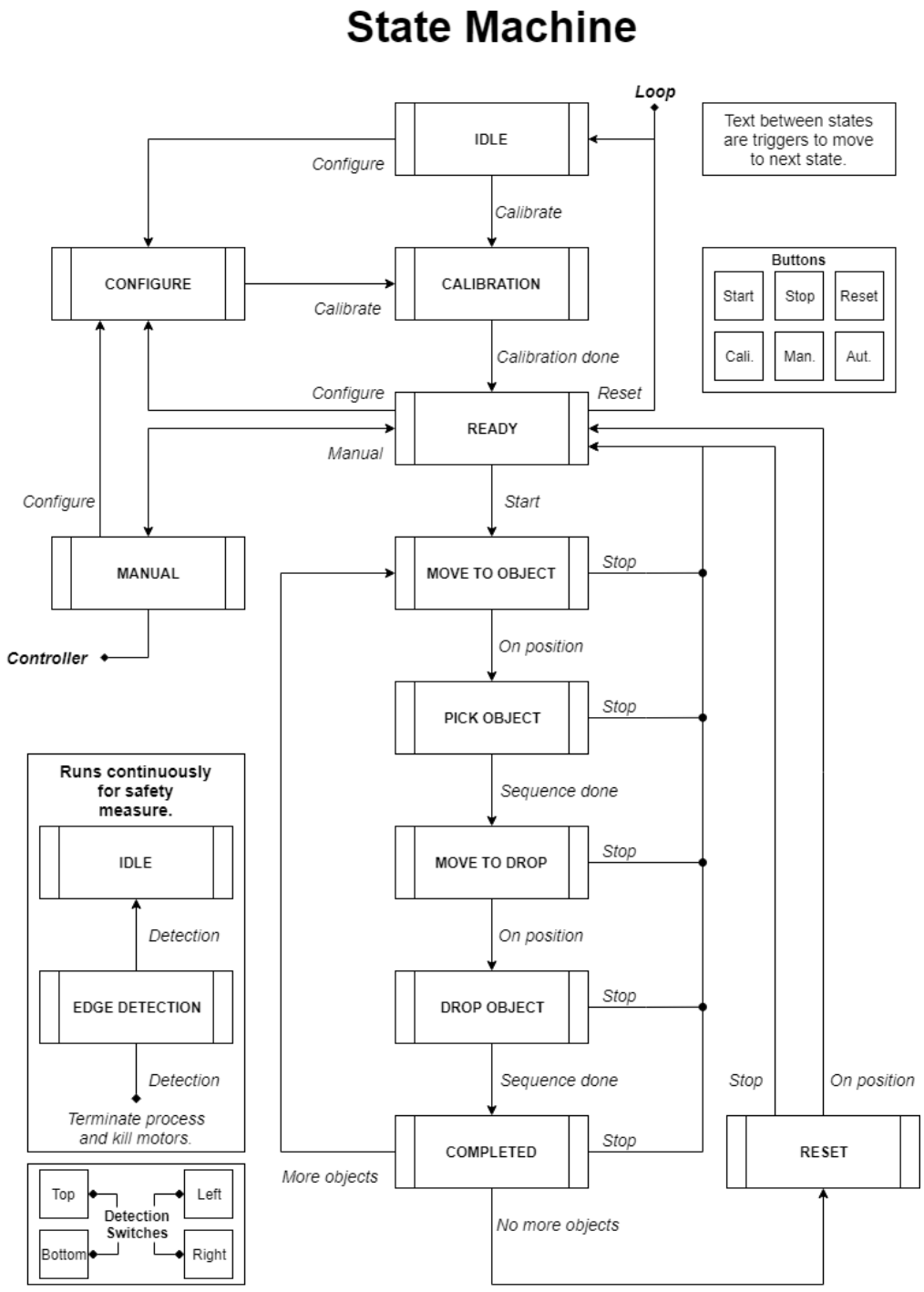


Figure 6.9: Robot state machine

The embedded state machine is the concept used to control the hardware of the robot. As seen in figure 6.9 the state machine consists of a state for every action. The reason for this is to have

a simple structure to follow, and minimizing processing for the microcontroller. As a result of this, the system can perform task without any input delay. Another thing to be mentioned, is that the microcontroller runs multiple task concurrent without implementing different threads. This is done by time management on different processes, and an example of this can be the implementation of the *pick and sort sequence* that have slow moving parts with no feedback control. With doing so, the information will go continuously without interruptions, while the pick or drop sequence is running.

```
boolean pickObject() {
  boolean picked = false;

  if (pickAndDropTimer.hasTimerExpired()) {
    digitalWrite(PISTON_DOWN, LOW);
    digitalWrite(PISTON_UP, HIGH);
    vacuumTimer.startTimer(VACCUM_DELAY_DURATION);
  } else {
    digitalWrite(PISTON_UP, LOW);
    digitalWrite(PISTON_DOWN, HIGH);
    digitalWrite(VACUUM, HIGH);
  }
  if ((pickAndDropTimer.hasTimerExpired()) && (vacuumTimer.hasTimerExpired
   ())) {
    picked = true;
  }
  return picked;
}
```

Where the function implements objects from the **Timer** class that was made for handling multiple time management tasks by using *millis()* as reference time.

```
void Timer::startTimer(unsigned long duration)
{
    this->nextTimeout = millis() + duration;
}
bool Timer::hasTimerExpired()
{
    return (millis() > this->nextTimeout) ? true : false;
```

```
}
```

6.2 Robot program

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

6.2.1 Program structure

The program structure is split up in different smaller structures. This means the system is developed with a high abstraction level in mind. In same way, it can handle interruptions, fails and other causes that will disrupt normal function of the program. In the end, is it the application server that runs on the Jetson Nano that binds everything together, and the other services/processes working collaboration with this. Working as branches for external input to be processed and parsed to the correct format.

All low-level logic, such as checking sensors and driving motors, will be handled by the Teensy. Interactions between the Jetson Nano and the Teensy are based on commands. The Teensy is responsible for carrying out the commands, and answer in form of statuses to the Jetson Nano. These statuses can relay information if that is requested. Statuses can be *current state, faults, position* and more. The commands include tasks like *start, reset, stop, manual override* and so on.

6.2.2 Operation flow

The robot operates in four states, see Figure 6.10. The first state that is *IDLE* is where the system waits for further instructions after being turned on. Second state is *CALIBRATE* which has to be started with a user input. This is where the motor, encoder and mapping calibrations takes place. However if the system is already calibrated, this step is ignored, and the system is already ready to operate. That mentioned, if any errors occurs during calibration, the system moves to *ERROR* state where actions from the user is required to proceed to *IDLE* state again. The last state is *OPERATE*, this is the state the robot is in when it picks and and sorts objects within the

workspace.

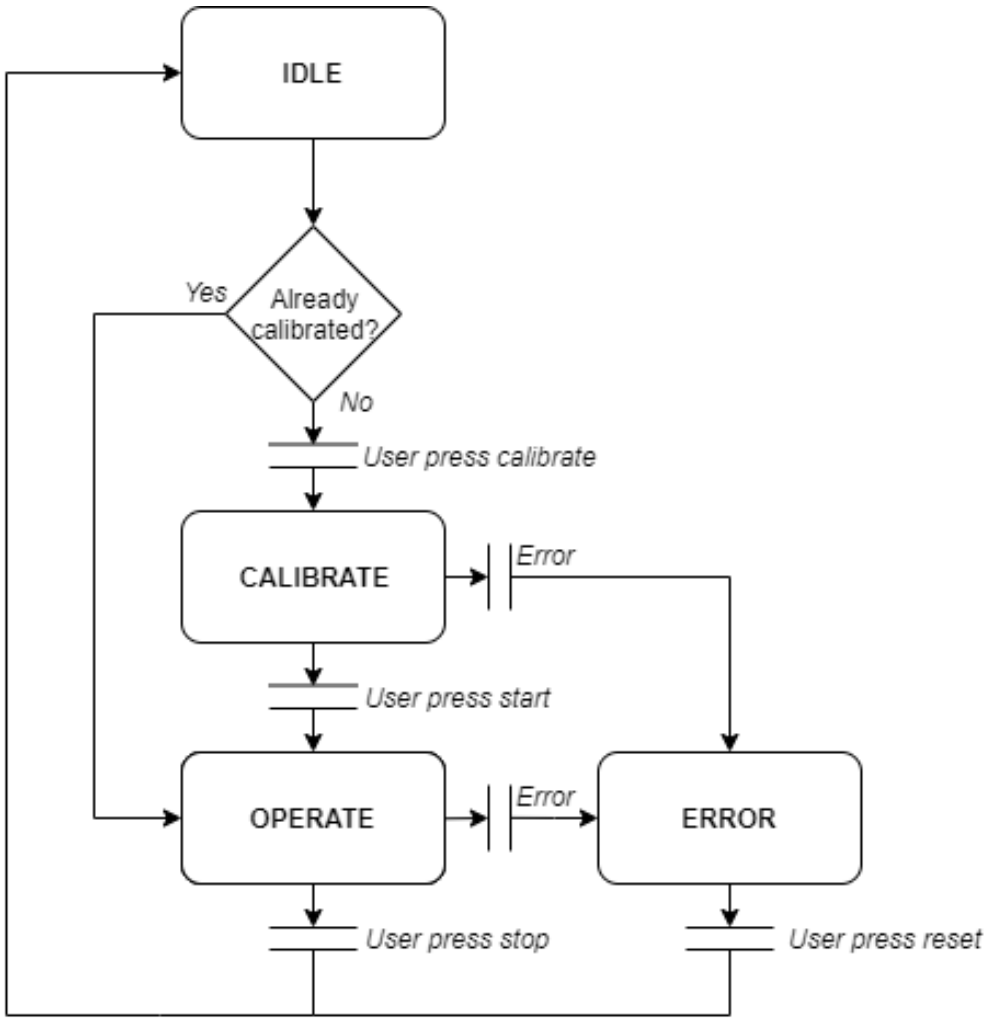


Figure 6.10: Main operation flow

6.2.3 Operation state flow

Figure 6.11 shows what happens from the second it is powered up and goes in depth when the calibration is done, the user has pressed start, and the main operation program enters the *OPERATE* state. As seen in figure 6.11 the *OPERATE* control the robots movements based on the object data. The search for objects happens continuously in order for the robot to always be up to date with the latest readings. Further, if the system detects an object, it will go in the following sequence seen in Figure 6.11. This will also be done correctly unless an error occurs. As the error checking also happens continuously, will the system respond to the error in any state, without

any delay. However, if everything runs smoothly, the robot will stop at the home position waiting for a new signal to before proceeding.

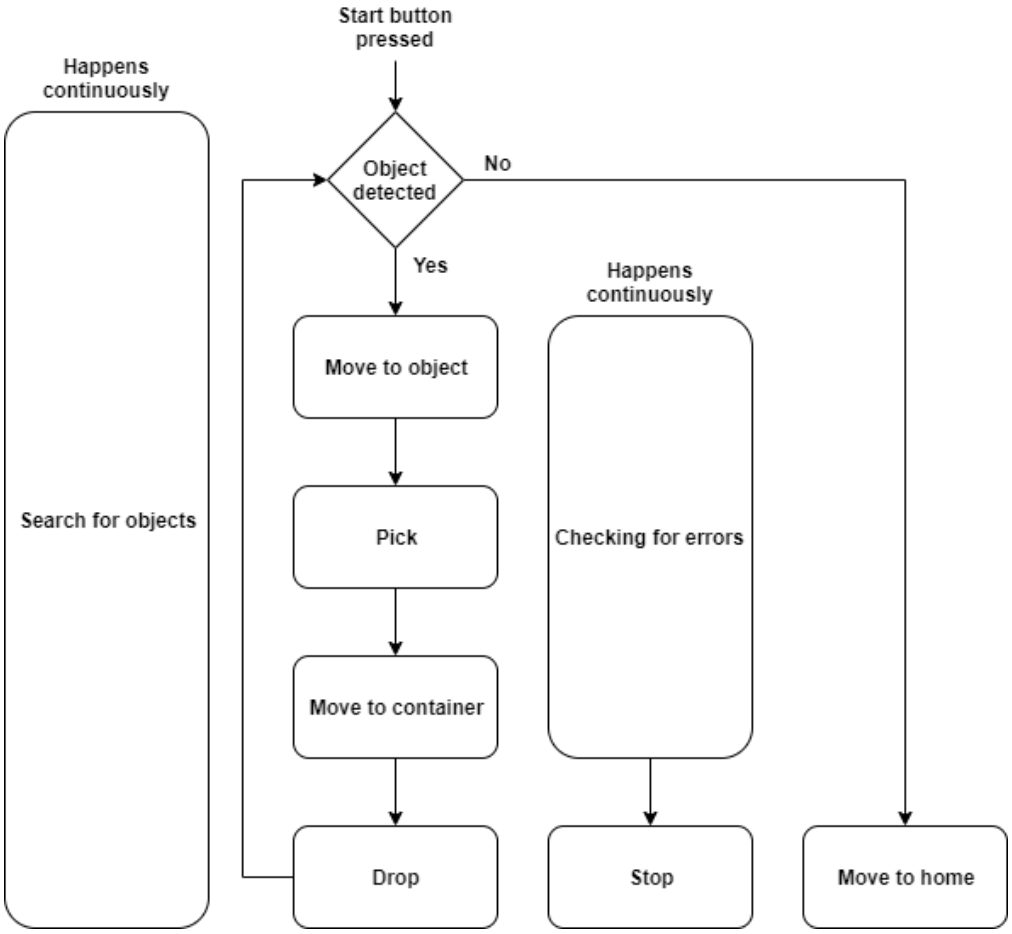


Figure 6.11: Main operation

6.3 Robot calibration

Calibrating the robot must be done before any other motion can be applied. The calibration gives the area for which the robot can move freely, but is also the foundation for the transformation from image coordinates to encoder coordinates which will be explained in 6.4.1 *Distance transformation* section.

Calibration is done by firstly setting the ODrive (Motor driver board) in a calibration mode where every single motor is turned approximately one revolution in each direction to measure motor

specific data (e.g. phase inductance and phase resistance). Afterwards a single axis at a time is driven to its home position and is stopped by the limit switches where the positional values of the encoders are referenced. The home position is in our case the corner that matches the (0, 0) pixel corner of the camera. From the home position the robot is then moved to the opposite limit switches to get the maximum travelling distance.

When both home position and maximum travelling distance in X and Y directions are known we can limit the robot's motion within that specific area. In this way the calibration is generalized to fit not only this robot, but robots at all sizes without any need for program changes.

6.4 Robot motion

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

6.4.1 Distance transformation

For transforming object coordinates the formula for a straight line is used to map coordinates from pixels to encoder counts. The map() function in Arduino is applied. By using encoder count values from calibration for defining the mapping range we can generalize the distance transformation to fit all robot sizes.

6.5 Convolution neural network

The *CNN* model is based on the famous Faster R-CNN ResNet50 model. In the Faster R-CNN setting, detection happens in two stages. In the first stage, called the *region proposal network* (RPN), images are processed by a feature extractor (e.g., VGG-16), and features at some selected intermediate level (e.g., "conv5") are later used to predict classagnostic box proposals.

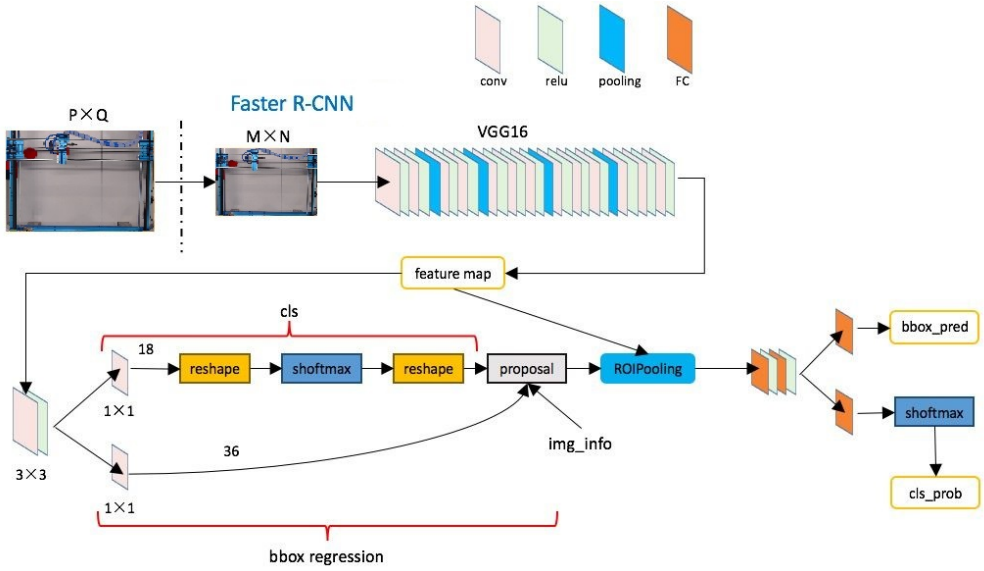


Figure 6.12: Faster R-CNN ResNet50 model

Collecting data

The whole group were needed to collect data for the CNN model. It was important to gather as much training data as possible since the CNN model's accuracy scales almost linear with the data given. As a result of this, were it collected 600 images containing the different object types. Aiming to capture the objects from multiple angles and surroundings to generalize the model, and making it more robust from external noise.

Figure 6.13: Training example 1

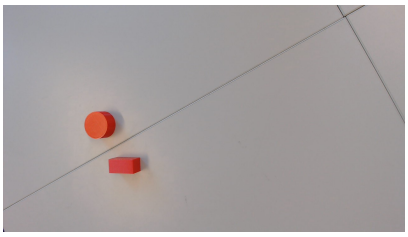


Figure 6.14: Training example 2

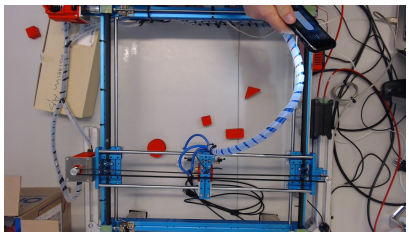


Figure 6.15: Training example 3

6.5.1 Labeling data

Labeled data is a designation for pieces of data that have been tagged with one or more labels identifying certain properties or characteristics, or classifications or contained objects. This is necessary to use a supervised machine learning setup.

Figure 6.16: Training example 1

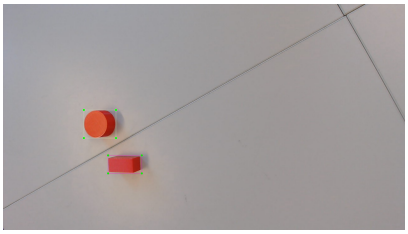


Figure 6.17: Training example 2

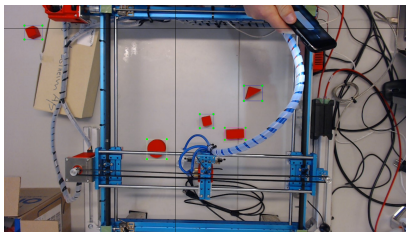


Figure 6.18: Training example 3

As shown in the figure 6.20 there is now a bounding box around the objects that our CNN should detect and classify. From the following images, a XML file is generated containing the data e.g. *filename, size, id, labels, bounding box coordinates.* This XML file needs to be converted to CSV-format before it can go into the model.

Filename	**Width**	**Height**	**Class**	**Xmin**	**Ymin**	**Xmax**	**Ymax**
filename1.jpg	1280	720	rectangle	578	282	763	447
filename2.jpg	1280	720	square	596	379	695	486
filename3.jpg	1280	720	circle	487	230	654	400
filename4.jpg	1280	720	rectangle	766	307	974	481
filename5.jpg	1280	720	triangle	398	409	518	518

6.5.2 Training

The training of the CNN model was done with Google's machine learning framework, Tensorflow. The training saved checkpoints of the model every 10000 epoch, keeping the top 5 newest if there would be any interruption during the training process. The training took approximately 9 hours to complete 200000 epochs. However, the training can be done in much less time, this will be stated in more depth in the results chapter.

```
INFO:tensorflow:global step 48173: loss = 0.0575 (0.945 sec/step)
INFO:tensorflow:global step 48174: loss = 0.0039 (0.948 sec/step)
INFO:tensorflow:global step 48175: loss = 0.0241 (0.948 sec/step)
INFO:tensorflow:global step 48176: loss = 0.0403 (0.945 sec/step)
INFO:tensorflow:global step 48177: loss = 0.0089 (0.946 sec/step)
INFO:tensorflow:global step 48178: loss = 0.0125 (0.947 sec/step)
INFO:tensorflow:global step 48179: loss = 0.0253 (0.942 sec/step)
INFO:tensorflow:global step 48180: loss = 0.0135 (0.952 sec/step)
INFO:tensorflow:global step 48181: loss = 0.0028 (0.945 sec/step)
INFO:tensorflow:global step 48182: loss = 0.0049 (0.945 sec/step)
INFO:tensorflow:global step 48183: loss = 0.0034 (0.950 sec/step)
INFO:tensorflow:global step 48184: loss = 0.0031 (0.949 sec/step)
INFO:tensorflow:global step 48185: loss = 0.0044 (0.949 sec/step)
INFO:tensorflow:global step 48186: loss = 0.0713 (0.948 sec/step)
INFO:tensorflow:global step 48187: loss = 0.0072 (0.947 sec/step)
INFO:tensorflow:global step 48188: loss = 0.0312 (0.952 sec/step)
INFO:tensorflow:global step 48189: loss = 0.0256 (0.950 sec/step)
INFO:tensorflow:global_step/sec: 0.979725
INFO:tensorflow:global step 48190: loss = 0.0111 (1.105 sec/step)
INFO:tensorflow:Recording summary at step 48190.
INFO:tensorflow:global step 48191: loss = 0.0084 (1.091 sec/step)
INFO:tensorflow:global step 48192: loss = 0.0080 (0.949 sec/step)
INFO:tensorflow:global step 48193: loss = 0.0410 (0.948 sec/step)
```

Figure 6.19: Training Tensorflow model

6.5.3 Evaluation

It is important to evaluate the model before it is being used in operation. The model was then introduced to new, never before seen pictures, and asked to perform the object detection and classification.

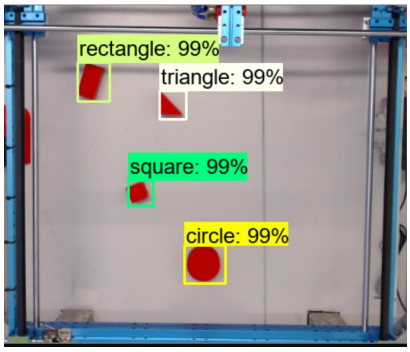


Figure 6.20: CNN Evaluation Example 1

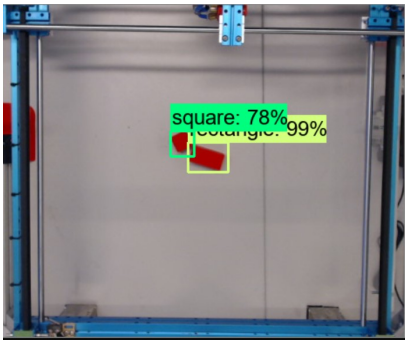


Figure 6.21: CNN Evaluation Example 2

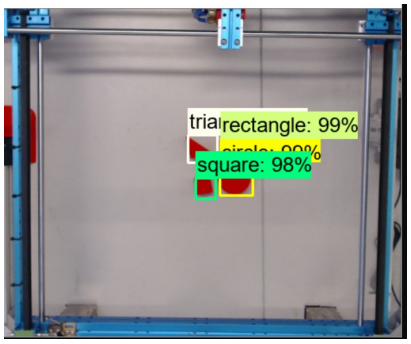


Figure 6.22: CNN Evaluation Example 3

6.6 Image processing

During the project there has been developed two image processing techniques. The first technique is meant to scale and rescale the image to match our dimensions. Not only for the CNN,

but also to send as little information as necessary to the remote server. The second method is encoding the image to bytes so it can be sent over TCP to remote server.

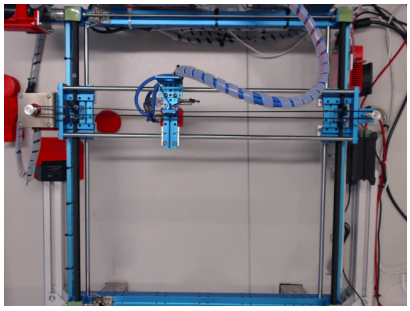


Figure 6.23: Original frame

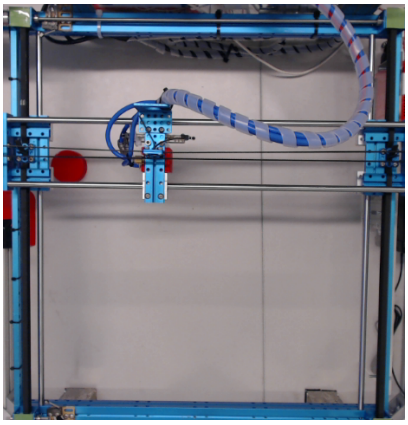


Figure 6.24: System frame

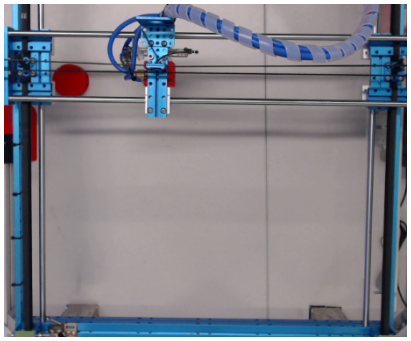


Figure 6.25: Detection frame

6.6.1 Digital containers

The digital containers are drawn directly on the frame before entering the web server. This containers are restricted areas where the robot will recognize the objects placed within the barriers as already sorted correctly, and will therefor discard them in the list of unsorted objects.

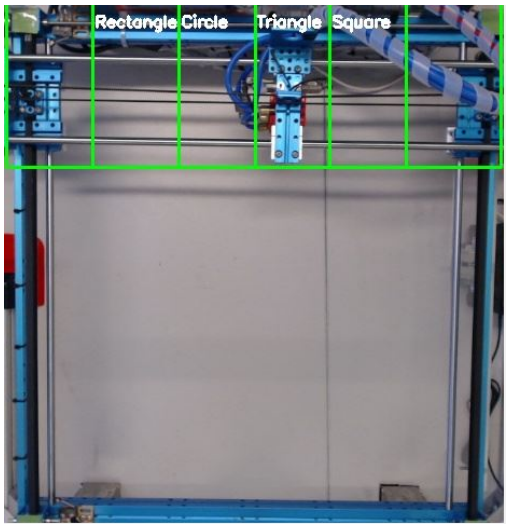


Figure 6.26: Digital containers

6.6.2 Data visualization

After the CNN has computed the frame and the REST-API uploaded the results, and later stored in the application database. It is time to visualize the results through the graphical user interface.

This is done in the back-end of the web server. The web server get the stored data through one of the TCP clients who continuously pull data for that subject. As a result of this, the data is passed in a JSON-parser who decode the data to JSON and sends this information to be drawn on the same frame as the digital containers are on.

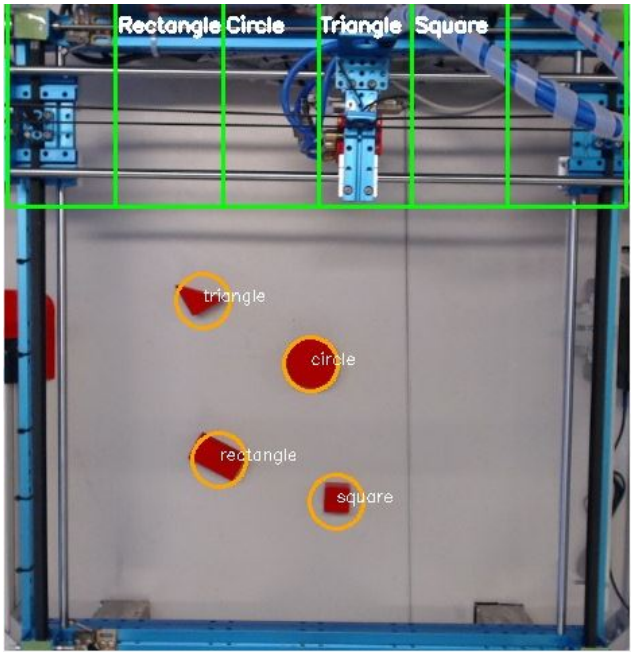


Figure 6.27: Data visualization on frame

6.7 Picking and sorting

Picking and sorting is done by utilizing a suction cup mounted on the piston end of the pneumatic cylinder. When the robots x and y positions is on the objects target center point, the vacuum is enabled while extending the cylinder to come in contact with. Thereafter, the cylinder is retracted and the object is picked up, after which it is placed in a available position withing it's object container.

By using timers to execute the cylinder and servo movements in the picking routine, we realize a multi-threading system allowing synchronization between the different aspects of the picking and sorting. Also, by using thread safe operations we manage not to corrupt any picking and sorting commands throughout the data flow.

6.8 Relays

Relays are used for controlling the coils in the pneumatic valves connected to the 24V supply with the 3V I/O on the Teensy 4.0. The relays needs to have the specifications as desired voltage and amount of contact sets for their intended purpose.

6.9 Emergency Stop

It is important that the emergency stop is rather physical power cutoff, than related to a software implementation to do its job. The emergency stop controls a relay which in turn are connected to the following components:

- Odrive V3.6 Motor controller
- Air cooling fan

6.10 Graphical user interface

A graphical user interface (GUI) has been developed. This interface will provide the user with option for controlling the robot, or just observe it operating. The GUI was created in JavaScript, HTML and CSS, which runs in the browser. When creating the GUI, the priorities was functionality and that it was easy to use, and available to anybody without any installation.

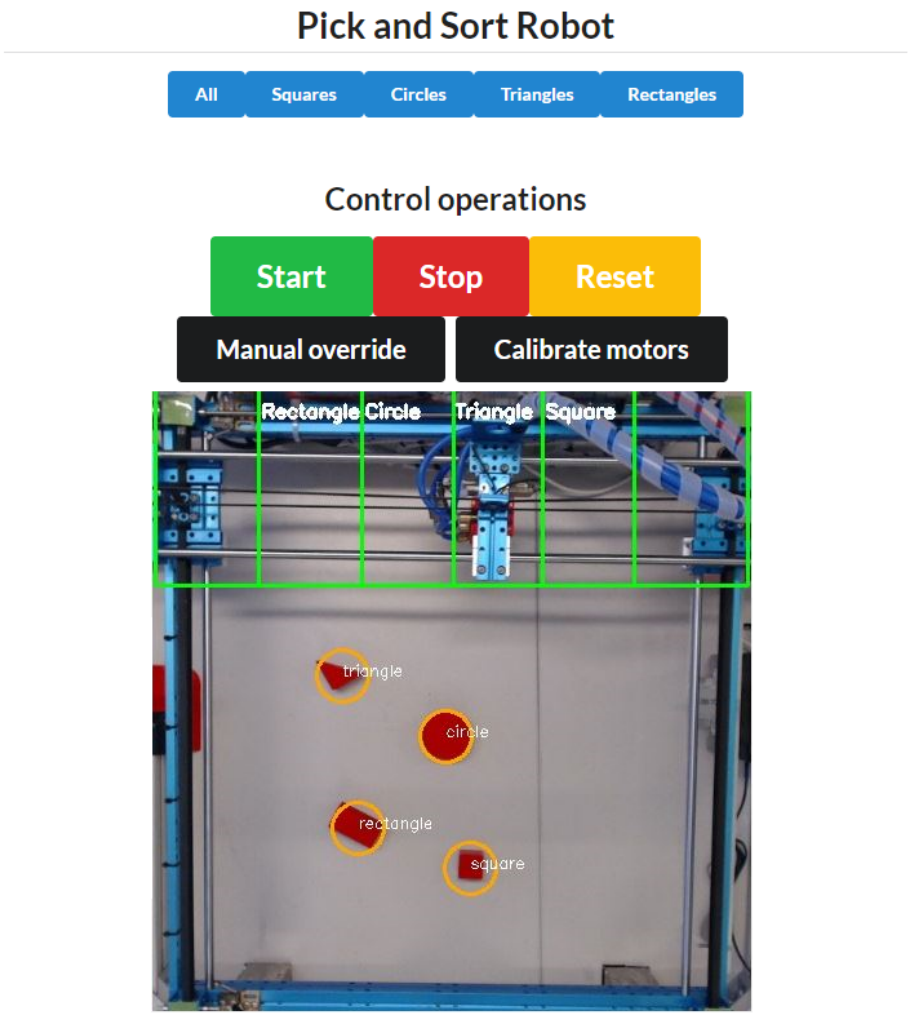


Figure 6.28: Graphical User Interface

The GUI is composed mainly of one frame, but do have separate hidden frames for each dynamic components. That mentioned, the user will only care about the main frame, as the dynamic components will update in real time, without needing to refresh the web page.

At last, as the GUI is rendered and all processed in the browser of the client. Thereby, eliminating any constraint on the web-server, and makes it that multiple can connect to the interface without any loss in performance.

Chapter 7

Reviews

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

7.1 Electrical testing

Before testing the robot, all the wiring of the components on the robot needed to be controlled. This was done to make sure the wiring was done correctly and that there were no short circuits that could potentially damage the electrical components. Consequently all connection points and wires were measured with a multimeter. After the test was completed with no faults, we applied the voltage and measured that all of the different voltages was correct.

7.2 System testing

Before assembling all the parts together, a set of test were done. These test made sure that all the connected components were working and configured correctly in the software.

Function	**Result**
Test emergency stop	Works
Test end switches	All four works
Test motors	Both works
Test encoders	Both works
Test valves	Both works

7.3 End switch testing

As a result of the end switches placements, they were a victim of electromagnetic noise generated from the motors. This made some of the end switches trigger, and resulted in terminating the motors power and configuring the Odrive to IDLE mode.

A filter was created to eliminate the unwanted end switch triggering. Therefor a button filter class was created to handle the obstacle. The filter simply check if the assigned button is continuously **HIGH** for longer than the set timeout.

```
void ButtonTimer::startButtonTimer(unsigned long duration) {
    this->nextButtonTimeout = millis() + duration;
}

bool ButtonTimer::isSwitchOn(int btn) {
    int btnState = digitalRead(btn);
    if (btnState != oldBtnState) {
        oldBtnState = btnState;
        startButtonTimer(this->timeOut);
    }
    return ((buttonTimerHasExpired()) && (btnState)) ? true : false;
}
```

7.4 Communication

7.4.1 Serial

The serial communication was divided into several smaller parts. First the initialize method was made, this method enable the usb connection. The next method was the serialEvent method. This method is receiving the JSON object from the Teensy when an event occurs. Further a method was made for sending a JSON object to the Teensy called sendData. The initialize procedure was also change at a later point to allow the usb connection to occur after the program had started. The rest of the code can be seen in the appendix B.1.

7.4.2 RX/TX

The sending from Teensy to the Odrive and the other way around are following a set of templates and already made methods from an existing library. The templates and methods that are used can be seen in the code B.1.

7.4.3 TCP/IP

There are running two separate TCP - Servers. One that sits on Jetson Nano running in Java that's communicating with the Python Web Server, and the second one that's on the remote computer. They are set up to handle completely different inputs, and the were some problems with the remote TCP - Server. The reason for this is the large amount of data that is sent through the socket. The buffersize is limited to *4096 bytes*, making it overflow when the bytestream from the client is significantly larger. The fix for the problem was to send a header which included the size of the image. Then on the receiving end add up the buffer until the whole image was sent.

7.5 Design reviews

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

7.6 Software reviews

Throughout the software progression and development of structures, a variety of different options has been considered, where some of them would work fine and as expected. By reviewing our code logic and structure and seeking guidance, different and better versions with looser coupling and a more professional finish has been developed.

7.6.1 Python, remote computing

In the first implementation of the software the information gained from the remote computer were return on the same socket that the picture were sent on. This solution turned out to not meet the required expectations. Therefore the implementation were change to use a REST-API to return the information back to the main application. This solution proved to meet the required expectation one can expect in a real-time system.

Chapter 8

Results

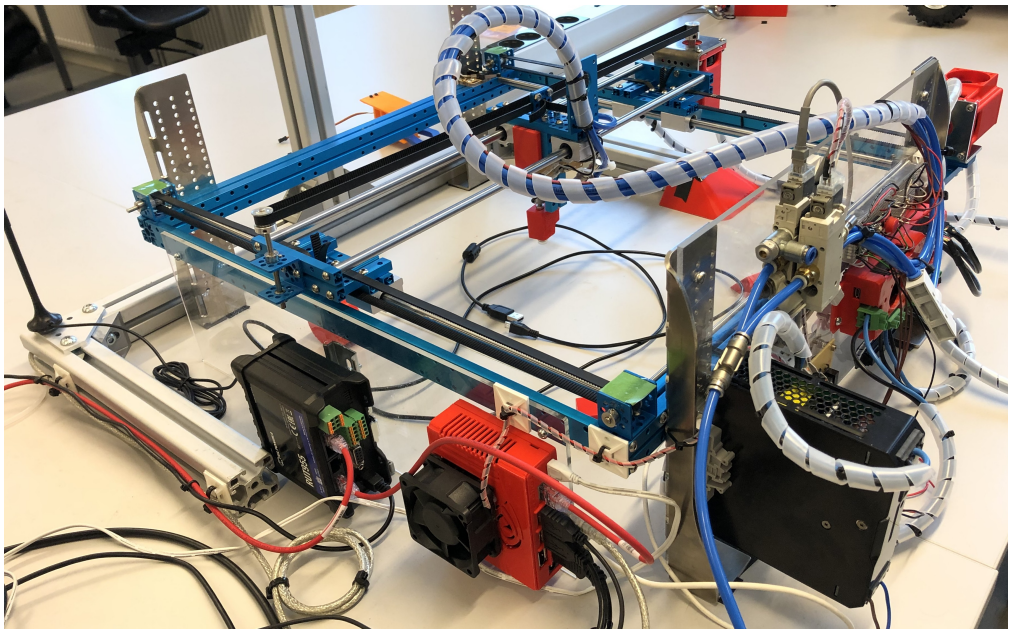


Figure 8.1: Pick and Sort Robot

This chapter elaborates on results of the various systems implemented in the robot and the robot in its entirety. A demo of the robot can be found here: Video demo. Figure 8.1 is a picture of the final system.

8.1 How the robot works

The robot detect and classify objects within the workspace of the frame. Furthermore will the robot move the mounted tool on top of one of the objects. With use of pneumatic cylinder and vacuum suction is it able to pick up the object and hold it. At last will the robot move towards the designated container and gently drop the object before moving towards a new object if more objects are present in the workspace. How the various operations explained above is developed can be read in chapter 5 and 6.

8.2 Workspace of the robot

The workspace of the robot can be seen in figure 8.2. The frame is 510mm x 560mm big. The picker can move left to right and up and down. The dotted lines on the lower part of figure 8.2 are the digital containers where the robot can put up to two objects. A camera is placed directly above the robot for the detection and classification of the objects.

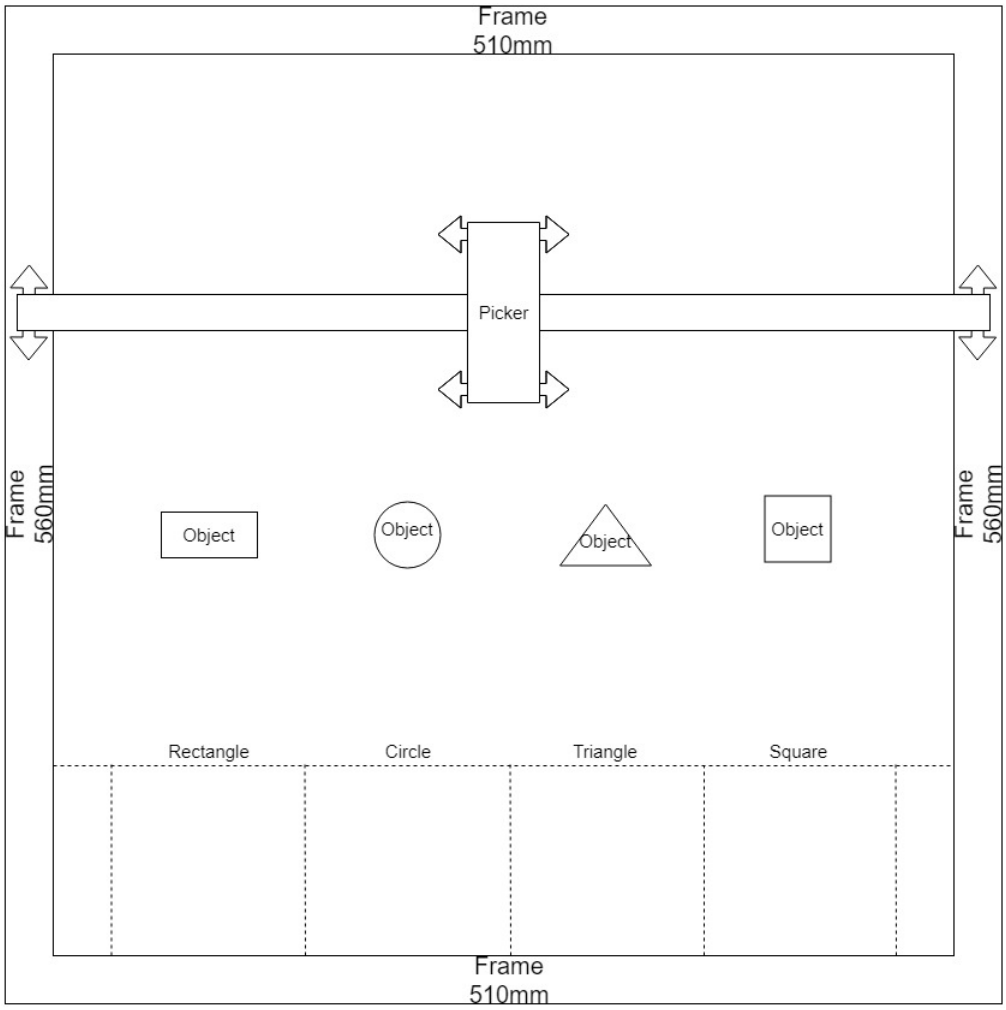


Figure 8.2: Workspace of the robot

8.3 System motion

As ODrive, that offers good integration with trajectory planning and acceleration profiles, is used for motion control, a good system motion is achieved. Steep acceleration and deceleration curves are taken advantage of, giving a high top speed rapid system response.

8.3.1 System movement

The overall system movement performs well. Picking can be executed for all, or just certain object types of object, and the system performs the movements necessary to achieve this. By utilizing a movement planner, both rapid and smooth trajectories are executed with good per-

formance.

8.3.2 System accuracy

The accuracy when picking up an object and placing it is measured to be no more than **1 mm** from target, giving us reason to believe the calibration routine and the transformation from image coordinates to real world coordinates are correct. Also, considering that the mechanical frame does not offer excellent rigidity, but is quite flexible, parts of the inaccuracy must originate from the construction itself.

8.4 Real time performance

As the subject where this project is hold is *Real time programming*, is the processing time for the different system crucial, and has to be within the time margins to be called a real time system.

Graphical user interface

The GUI with its asynchronous calls for the buttons gives response to the web server in a time of **3 milliseconds**. This was tested with various buttons, and in multiple scenarios to get an accurate average of the input delay.

Web Server

The Web server consists of multiple parts, where multiple processes run in their own threads. The measurements shows that preparation to send images takes **30 milliseconds** before it is ready to be sent to the remote TCP - Server.

Concurrently is the communication with the application server in **30 milliseconds**, in addition to this uses the image processing **10 milliseconds** to draw the digital containers and the results on the frame.

At last, the objectlist component in the GUI is updated every **10 milliseconds**, and the system status every **100 milliseconds** as this is limited to the send frequency from the Teensy.

Remote computation

The remote TCP - Server is able to receive the full image in **1 millisecond**, before the CNN uses in average of **62 milliseconds** to detect and classify the different objects in the image.

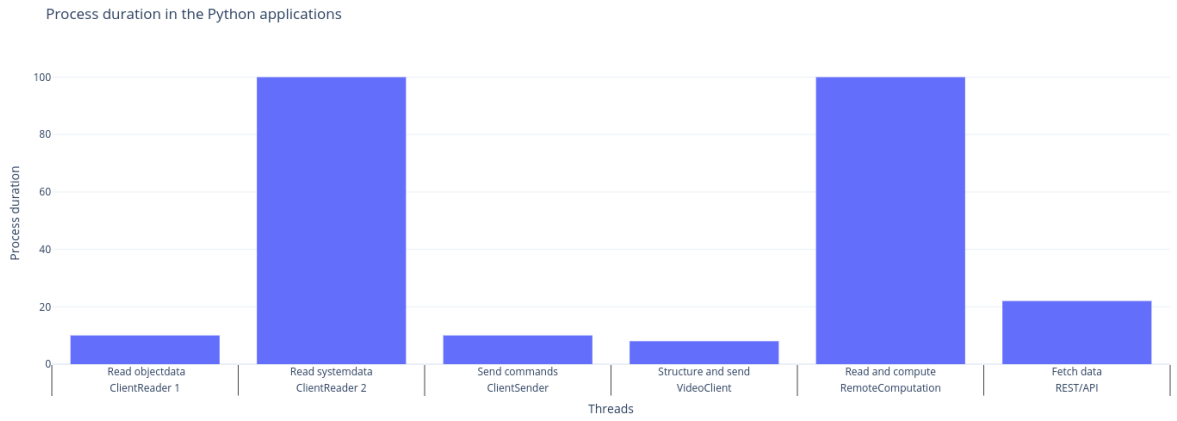


Figure 8.3: Python application time usage

Application core

As stated earlier, the TCP communication takes **10 milliseconds** to receive input and return a result to the clients. The serial communication runs in a separately thread and reads data on requests.The Teensy sends every **100 milliseconds**, and the time it takes the thread to save data to the database is lower than **1 milisecond**. Concurrently to reading, the sending to Teensy is set to be at **20 milliseconds**, making it slow enough so the Teensy is always ready to receive data without overwriting its buffer. At last the REST/API fetches data from the remote stored file on a interval of **40 milliseconds**.

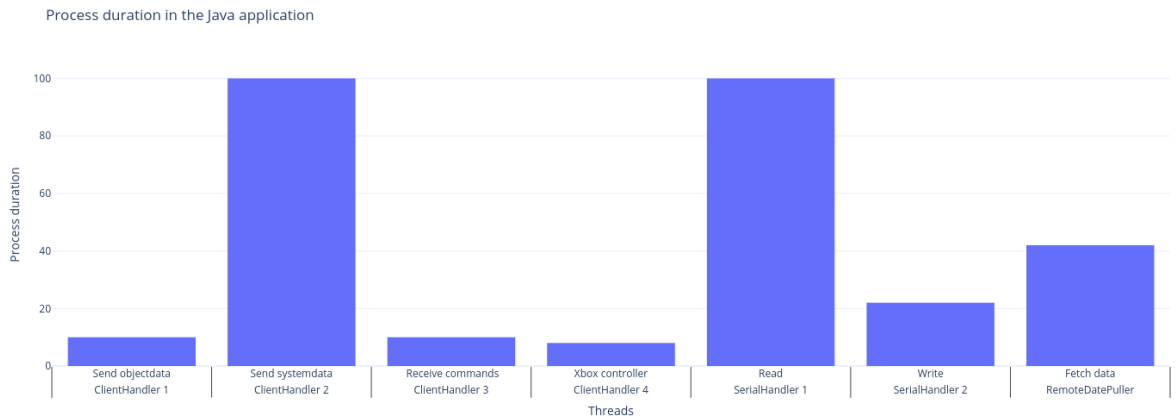


Figure 8.4: Java application time usage

By just looking at the figures 8.5 and 8.6, it looks like some of the threads uses a lot of time on their task, however most of it is I/O operations, meaning they are in a sleeping mode while other processes work to complete their tasks.

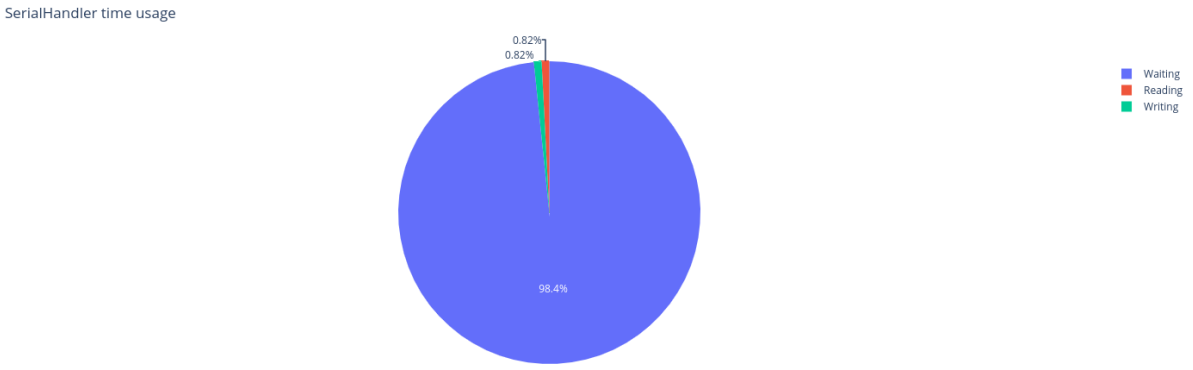


Figure 8.5: SerialHandler time usage

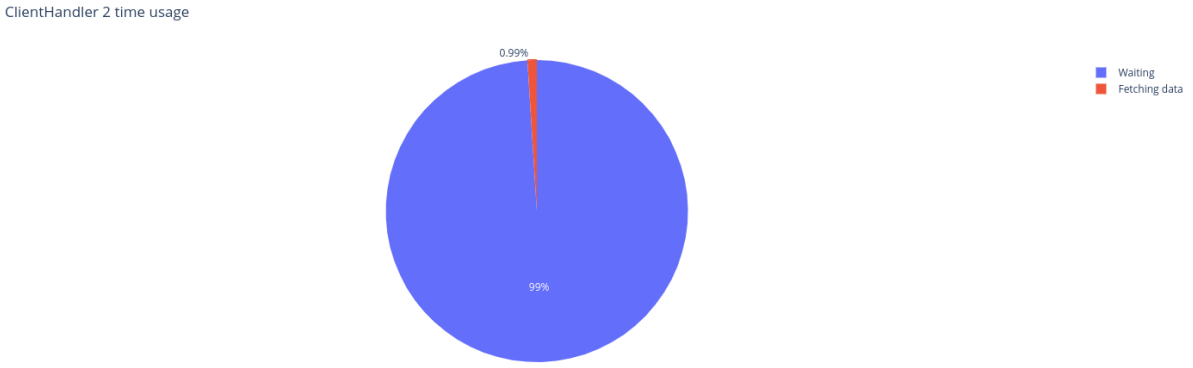


Figure 8.6: ClientHandler time usage

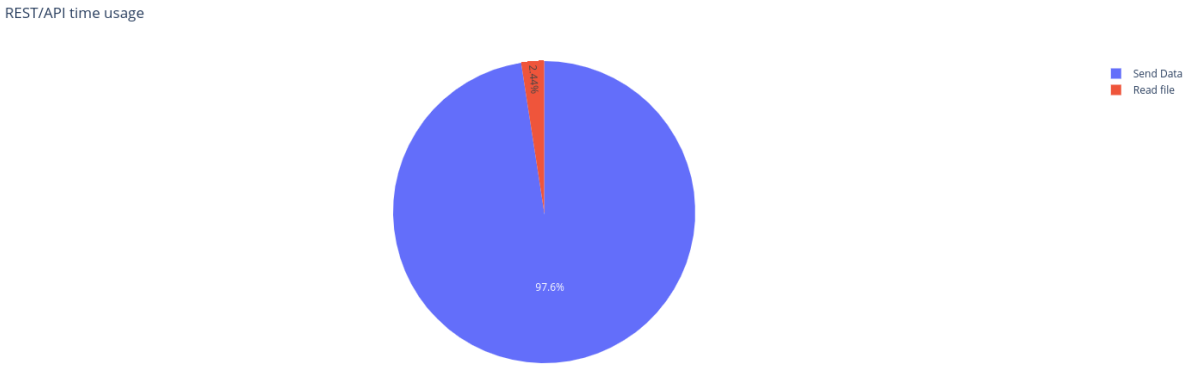


Figure 8.7: REST/API time usage

Embedded device

With multiple measurements in different states, the average loop time on the Teensy lands on **3 milliseconds**, making it running acceptable to always handle incoming data from serial and detect error/faults close to instantly.

8.5 Convolution neural network

The CNN developed was trained on a large variety of taken images. This resulted in a total of 800 labeled objects for training, and 200 for testing. As figure 8.3 shows, the data was sufficient enough to train the model to get a very low loss on the training data within 90000 epochs. This proves the robustness of the model, and it can with very little training data be retrained to any object in just a couple of hours.

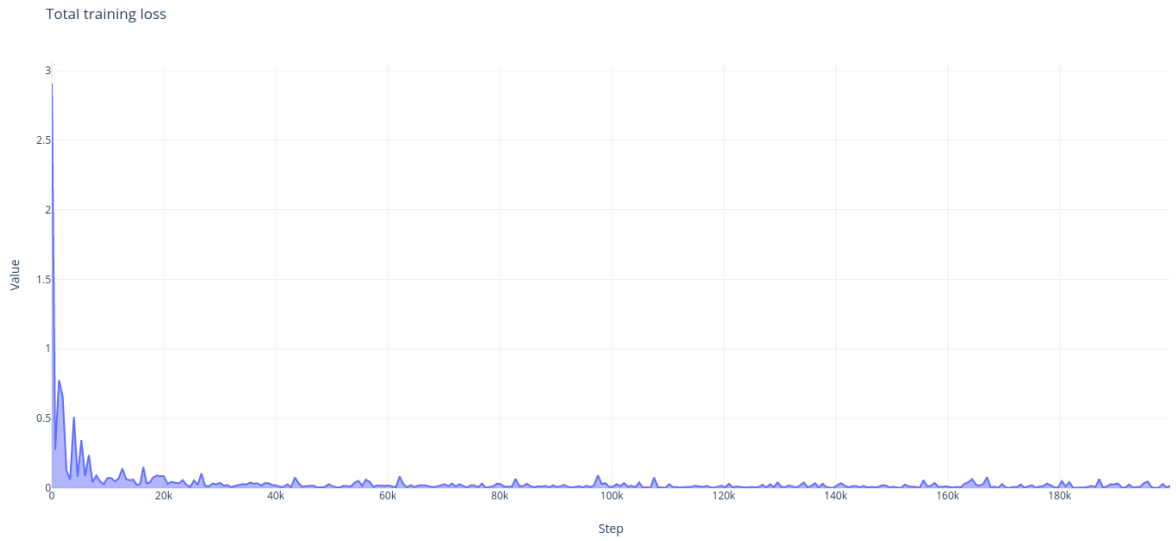


Figure 8.8: Total training loss

8.5.1 Accuracy

The CNN were later tested in order to verify its accuracy. The CNN model was tested with a number of never before seen sampled images. By knowing the correct labels in each image, a confusion matrix for the accuracy were made.

Confusion matrix

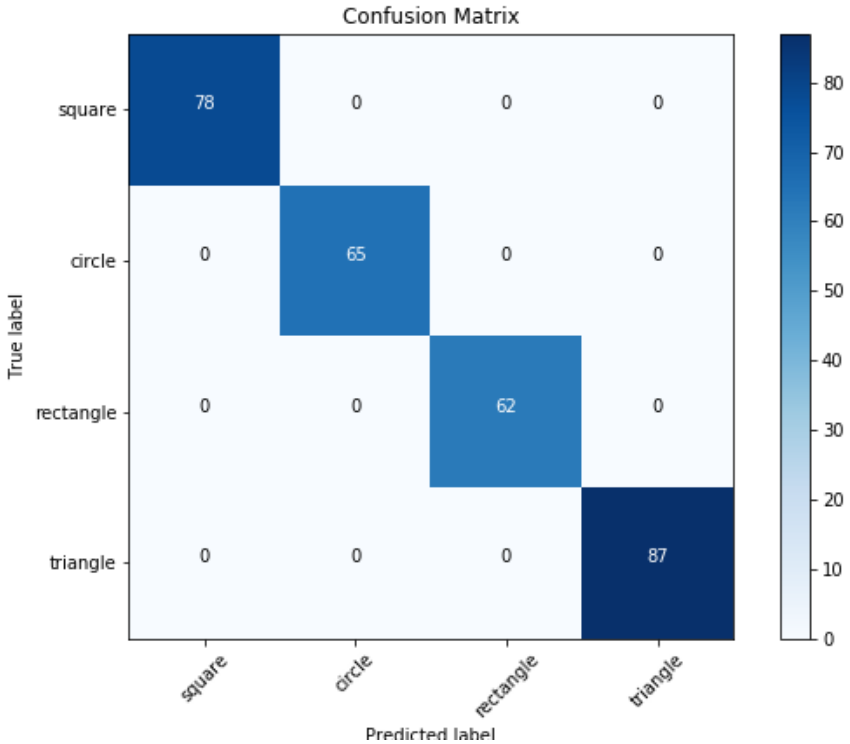


Figure 8.9: Confusion matrix

The confusion matrix figure 8.9, reveals the accuracy of the CNN model. As the confusion matrix shows, 100% of the objects in the images were detected and classified correct.

Chapter 9

Discussion

In this chapter we discuss the results obtained in the previous chapter and the reliability of the data collected. Including our experiences during the project.

9.1 Results from testing

Here we discusses the results obtained from the previous chapter.

9.1.1 Motion

The system has an acceleration that is around the maximum of what the mechanical frame can handle without causing too much flexing. By modifying the acceleration curve, perhaps a more rapid motion could be achieved, but would likely yield low improvements because of the small robot size. Also, the cylinder for the vertical pick and place movement is noticeably slower then the servo motors. This would improve with feedback sensors, giving it less waiting time between operations. Maybe, changing the cylinder with a servo operated linear actuator would be the best direction to take. However, this would probably be too heavy for the current hardware.

9.1.2 Real time performance

The real time performance were up on part with what we wanted to achieve. That is, seamless connection between the different systems without any noticeable delay. In combination with a

CNN that performs remarkable, doing the classification and detection at a speed compared to some image processing algorithms while being robust to light conditions. Similarly the Teensy 4.0 handle the load of operations with ease, running the loop close to 400Hz, making it suitable for much harder tasks. All this holds every operation thread safe with no corrupt memory or any deadlocks interfering with the systems.

9.1.3 Convolution neural network

As our first intuition was to have everything run locally on the Jetson Nano, where multiple CNN models trained and tested before it landed on the ResNet50 model. Many of the early models didn't deliver on performance, and the once who did, used to much resources. Nevertheless did we not manage to run any CNN that gave a respectable result locally on the Jetson Nano. With this mentioned, it will probably be possible in the near future with the use of *Deepstream SDK*, developed by Nvidia, which is a promising new technology meant to optimize the models for single board computers.

9.1.4 Remote computing

The remote computing solution worked surprisingly well. The delay is about half a second, and that is not much considering the computational work the computer do and the distance. The only problem that was encountered with the remote computer, was one day when it was power outage where the computer is located. The result of this is that the robot is not able to detect objects, and therefore unable to act. One other problem that can occur is loss of internet either on the school or at the remote place, but this was never encountered. Solutions to these problems are explained in chapter 9.4.

9.2 Software

Regarding the software, the GUI has not been prioritized. This does not mean it has bugs, but rather lack some features. After the biggest bugs were fixed, the program functioned as it was supposed to. All the big parts of the program was developed separately and then put together as

a whole, with the *ClientHandler* in the Application Core as its main logic. This made it possible to develop the CNN, motion planner, GUI, Web server, state machine and the communication classes individually and connecting them through the API.

9.3 Physical structure and design

In the beginning of the project, an extensive amount of time was used for designing the parts of the robot in 3D and finding suited components. This paid off when we were assembling the robot. Most of the parts and components were installed without any problems and no major changes were needed. The finished prototype is solid with both the horizontal and vertical motion being stable during operation.

9.4 Improvements for the future

The following points are suggestions for further development of the Pick and Sort Robot.

- **Hardware**
  - Equip a larger vacuum suction cup.
  - Change the vacuum suction to a gripper.
  - Implement a depth camera for optimal gripping.
  - Use absolute encoders to eliminate any need of calibration at every start-up.
  - Increase the area of the workspace.
  - Add feedback control on cylinder position.
  - Add a rotational axis on the picker so objects can be stacked closer together when placed.
  - Mount camera stand to workspace, to get everything straight.
- **Software**
  - Optimize the PID-controller for higher motor speeds.

- Eventually move the CNN to local, too eliminate external disruptions.
- Implement a smaller, slower CNN model locally, to take over the computation if the robot looses contact with the remote server.
- Train the CNN model on even more objects.

9.5 Experiences

9.5.1 Planning

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

9.5.2 Division of labor

All group members have different job and field experience. The different backgrounds have been used in the work distribution to take advantage of each members expertise. In a multidisciplinary project, as this turned out to be, with planning, constructing, and building the robot, led to self-learning and everyone developing new skills during the project.

Chapter 10

Conclusion

The project aims to develop a robot prototype for pick and sort of different shaped objects. The group is very pleased with the result, as the goal was to create a robot that was fast and accurate in both motion and with the CNN's detection and classification in real time.

Our prototype has accomplished the required accuracy and repeatability we set from start. Utilizing concurrent processes and artificial intelligence, the robot manages to detect and classify all the objects in the workspace even in various lighting conditions, and sort them efficient with high speed and accuracy.

This mentioned, there are many improvements to be made on both the design and software side that can increase the reliability and performance of the system. All the improvements that can be made are listed in section 9.4.

Bibliography

[1] Arduino. Arduino ide, 1 2019. URL https://www.arduino.cc/.

[2] Autodesk. Fusion 360, 1 2019. URL https://www.autodesk.com/products/fusion-360/students-teachers-educators.

[3] Benoît Blanchon. Arduinojson: Efficient json serialization for embedded c++, 11 2019. URL https://arduinojson.org/.

[4] Ivvar Blindheim. Real time programming course, 09 2019.

[5] Jason Brownlee. A gentle introduction to computer vision, 07 2019. URL https://machinelearningmastery.com/what-is-computer-vision/.

[6] Draw,io. Draw.io, 1 2019. URL https://about.draw.io/.

[7] Encoder. What is an encoder?, 02 2016. URL http://encoder.com/blog/company-news/what-is-an-encoder/.

[8] fazecast. jserialcomm, 11 2019. URL https://fazecast.github.io/jSerialComm/.

[9] Festo. Vacuum generator, pneumatic vn, 01 2019. URL https://www.festo.com/us/en/p/vacuum-generator-pneumatic-id_VN/.

[10] JetBrains. Intellij idea, 10 2019. URL https://www.jetbrains.com/idea/features/.

[11] JIMBLOM. Serial communication, 11 2019. URL https://learn.sparkfun.com/tutorials/serial-communication/all.

[12] jQuery. jquery.ajax(), 11 2019. URL https://api.jquery.com/jquery.ajax/.

[13] Keras. Keras documentation, 11 2019. URL https://keras.io/.

[14] Dhairya Kumar. Introduction to data preprocessing in machine learning, 12 2018. URL https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d.

[15] Makeblock. Xy plotter robot kit, 10 2019. URL https://www.makeblock.com/project/xy-plotter-robot-kit.

[16] Matplotlib. Matplotlib: Python plotting, 11 2019. URL https://matplotlib.org/.

[17] Steve Meyer. Linear motion, 4 2009. URL https://www.therobotreport.com/linear-motion/.

[18] Microsoft. Visual studio code, 1 2019. URL https://code.visualstudio.com/.

[19] Chris Nicholson. A beginner's guide to neural networks and deep learning, 1 1990. URL https://skymind.ai/wiki/neural-network.

[20] Numpy. Numpy, 11 2019. URL https://numpy.org/.

[21] Odrive Robotics. Odrive tool, 1 2019. URL https://odriverobotics.com/.

[22] Odrive Community open source. Odrive motor guide, 11 2019. URL https://docs.google.com/spreadsheets/d/12vzz7XVEK6YNIOqH0jAz51F5VUpc-lJEs3mmkWP1H4Y/edit#gid=0.

[23] OpenCV. Opencv, 11 2019. URL https://opencv.org/.

[24] Overleaf. Overleaf, 1 2019. URL https://overleaf.com.

[25] pc control. Incremental encoders, 01 2008. URL https://www.pc-control.co.uk/incremental_encoders.htm.

[26] PCSchematic. Pcschematic automation, 1 2019. URL https://www.pcschematic.com/en/electrical-cad-design-drawing-software/electrical-cad/electrical-cad-user-perspective.htm.

[27] The Pallet Projects. The pallet projects, 11 2019. URL https://www.palletsprojects.com/p/flask/.

[28] Maven Repository. Json in java, 08 2019. URL https://mvnrepository.com/artifact/org.json/json.

[29] Margaret Rouse. Tcp/ip (transmission control protocol/internet protocol), 10 2008. URL https://searchnetworking.techtarget.com/definition/TCP-IP.

[30] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 12 2018. URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[31] Sam. Motor drivers vs. motor controllers, 08 2019. URL https://core-electronics.com.au/tutorials/motor-drivers-vs-motor-controllers.html.

[32] Dipanjan (DJ) Sarkar. A comprehensive hands-on guide to transfer learning with real-world applications in deep learning, 11 2018. URL https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in

[33] sas. Machine learning what it is and why it matters, 11 2019. URL sas.com/en_us/insights/analytics/machine-learning.html.

[34] StrikerX3. Jxinput, 08 2018. URL https://github.com/StrikerX3/JXInput.

[35] Tensorflow. Tensorflow, 11 2019. URL https://www.tensorflow.org/.

[36] Trinamic. Trinamic foc, 11 2019. URL https://www.trinamic.com/technology/std-technologies/field-oriented-control/.

[37] Ultimaker. Cura, 1 2019. URL https://ultimaker.com/software/ultimaker-cura.

[38] Oskar Weigl. Odrive, 11 2019. URL https://odriverobotics.com/.

[39] Wikepedia. Brushless dc electric motor, 11 2019. URL https://en.wikipedia.org/wiki/Brushless_DC_electric_motor.

[40] Wikepedia. Cascading style sheets, 11 2019. URL https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[41] Wikepedia. Camera, 11 2019. URL https://en.wikipedia.org/wiki/Camera.

[42] Wikepedia. Cartesian robot, 10 2019. URL https://en.wikipedia.org/wiki/Cartesian_coordinate_robot.

[43] Wikepedia. Cloud computing, 11 2019. URL https://en.wikipedia.org/wiki/Cloud_computing.

[44] Wikepedia. C++, 11 2019. URL https://en.wikipedia.org/wiki/C%2B%2B.

[45] Wikepedia. Html, 11 2019. URL https://en.wikipedia.org/wiki/HTML.

[46] Wikepedia. Digital image processing, 11 2019. URL https://en.wikipedia.org/wiki/Digital_image_processing.

[47] Wikepedia. Industrial robot, 10 2019. URL https://en.wikipedia.org/wiki/Industrial_robot.

[48] Wikepedia. Java (programming language), 11 2019. URL https://en.wikipedia.org/wiki/Java_(programming_language).

[49] Wikepedia. Javascript, 11 2019. URL https://en.wikipedia.org/wiki/JavaScript.

[50] Wikepedia. Pid controller, 11 2019. URL https://en.wikipedia.org/wiki/PID_controller.

[51] Wikepedia. Python (programming language), 11 2019. URL https://en.wikipedia.org/wiki/Python_(programming_language).

[52] Wikepedia. Representational state transfer, 11 2019. URL https://en.wikipedia.org/wiki/Representational_state_transfer.

[53] Wikepedia. Region of interest, 11 2019. URL https://en.wikipedia.org/wiki/Region_of_interest.

[54] Wikepeida. Robot kinematics, 6 2019. URL https://en.wikipedia.org/wiki/Robot_kinematics.

Appendices

Appendix A

First appendix

A.1 Progress schedule

Prosjektstatus

100% FULLFØRT

AKTIVITET	PROSENT FULLFØRT
Prosjektplanlegging og dokumentasjon	100 %
Idemyldring	100 %
Arbeidsfordeling	100 %
Elektrisk dokumentasjon	100 %
Mekanisk dokumentasjon	100 %
Prosjektrapport	100 %

AKTIVITET	PROSENT FULLFØRT
3D - Modelling	100 %
Motorbokser	100 %
Figurer	100 %
Gripefeste	100 %
Motorfester	100 %
Odrive kasse	100 %
Jetson Nano kasse	100 %
Overgang for kabelføring	100 %

AKTIVITET	PROSENT FULLFØRT
Konstruksjon	100 %
Feste motorer	100 %
Koble styrestrøm	100 %
Koble hovedstrøm	100 %
Lage griper	100 %
Pnaumatisk utstyr	100 %
Kretskort for releer	100 %

AKTIVITET	PROSENT FULLFØRT
Java	100 %
Kommunikasjon mot arduino	100 %
Server	100 %
Klient	100 %
Databehandling	100 %
Ruteplanlegging	100 %
Manuell styring	100 %
Automatisk sortering	100 %
Xbox kontroller styring	100 %
Felles ressurs (Database)	100 %
Hente data fra REST API	100 %

AKTIVITET	PROSENT FULLFØRT
Python	100 %
Objekt klassifisering	100 %
Objekt tracking	100 %
Databehandling	100 %
Klient	100 %
Webserver	100 %
REST API Server	100 %
Skyløsning for AI	100 %
Datavisualisering	100 %
Virtuelle containere	100 %

AKTIVITET	PROSENT FULLFØRT
Arduino	100 %
PID kontroller	100 %
JSON behandling	100 %
Kommunikasjon mot Odrive	100 %
Kommunikasjon mot Server	100 %
Tilstands maskin	100 %
IO - handtering	100 %
Tidsstyring	100 %
Motorkalibrering	100 %
Plukk og slipp sekvens	100 %
Styrelogikk	100 %
Kommandohåntering	100 %
Enkoderkalibrering	100 %
Filter for endebryter	100 %

AKTIVITET	PROSENT FULLFØRT
Grafisk brukergrensesnitt	100 %
Lese fra server	100 %
Skrive til server	100 %
HTML (Layout)	100 %
CSS (Style)	100 %
JavaScript (Funksjonalitet)	100 %
Videostream	100 %
Objektliste	100 %
Status	100 %

AKTIVITET	PROSENT FULLFØRT
Testing og optimalisering	100 %
C++	100 %
Java	100 %
Python	100 %
GUI	100 %

A.2 Electrical drawings

Skoleversion

Pick-And-Sort-Robot

Pick-And-Sort-Robot

Component	Article	Type	Manufacturer	Supplier	Description	Position
		230VAC/24VDC				
		Jetson nano				
		Teensy 3.6				
		C930 webcam				
		2x Dual shaft Motor				
		2x 8192 CPR Encoder				
		4x Limit switch				
		4x 10k Resistance				
		2x 5/2 Valve				
		3x Relay				
		Odrive				
		50WR5J Resistance				
		Switch				
		Suction cup				
		Vaccum generator				
		Cylinder				

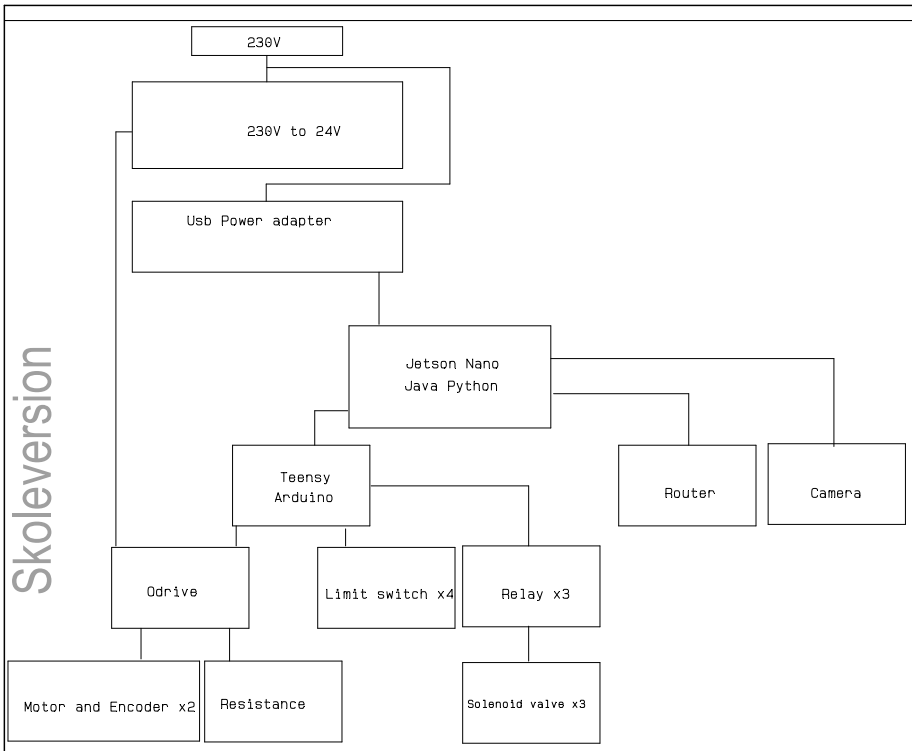
Skoleversion

PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**	**Project rev.:**	**Page**	**1**
Customer:	NTNU	DCC:		Scale:	1:1
Page title:	Component list	Dwg. no.:	Page rev.:	Previous page:	0
File name:	Pick-And-Sort-Robot	Eng. (proj/page): Petter	Last print: 19.11.2019	Next page:	2
Page ref.:		Appr. (date/init):	Last edit: 19.11.2019	Total no. of pages:	9

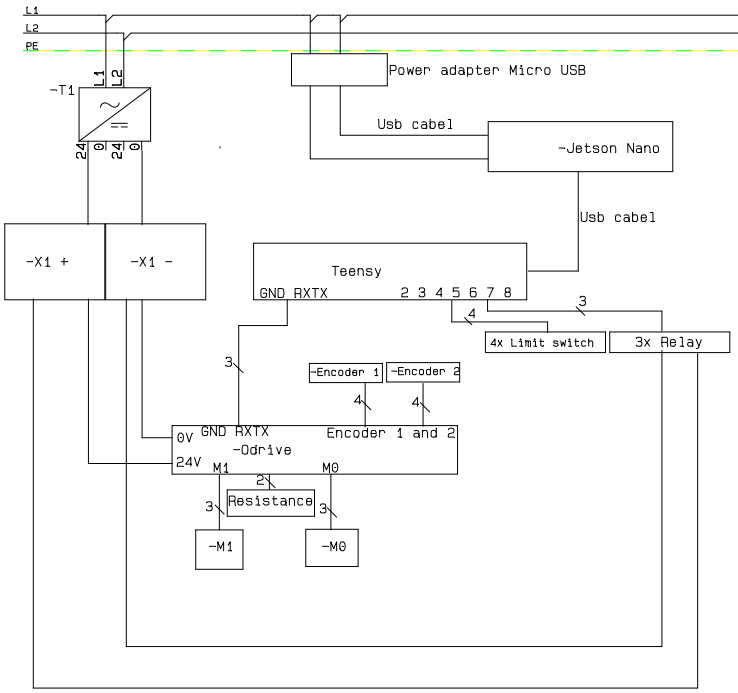
Skoleversion



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**		**Project rev.:**	**Page** 2
Customer:	NTNU	DCC:			Scale: 1:1
Page title:	Overview	Dwg. no.:		Page rev.:	Previous page: 1
File name:	Pick-And-Sort-Robot	Eng. (proj/page):	Petter	Last print: 19.11.2019	Next page: 3
Page ref.:		Appr. (date/init):		Last edit: 19.11.2019	Total no. of pages: 9



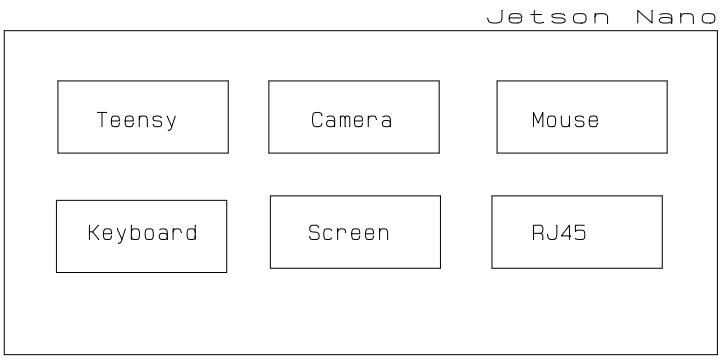
Skoleversion

PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**	**Project rev.:**	**Page**	**3**
Customer:	NTNU	DCC:		Scale:	1:1
Page title:	Control power	Dwg. no.:	Page rev.:	Previous page:	2
File name:	Pick-And-Sort-Robot	Eng. (proj/page): Petter	Last print: 19.11.2019	Next page:	4
Page ref.:		Appr. (date/init):	Last edit: 19.11.2019	Total no. of pages:	9

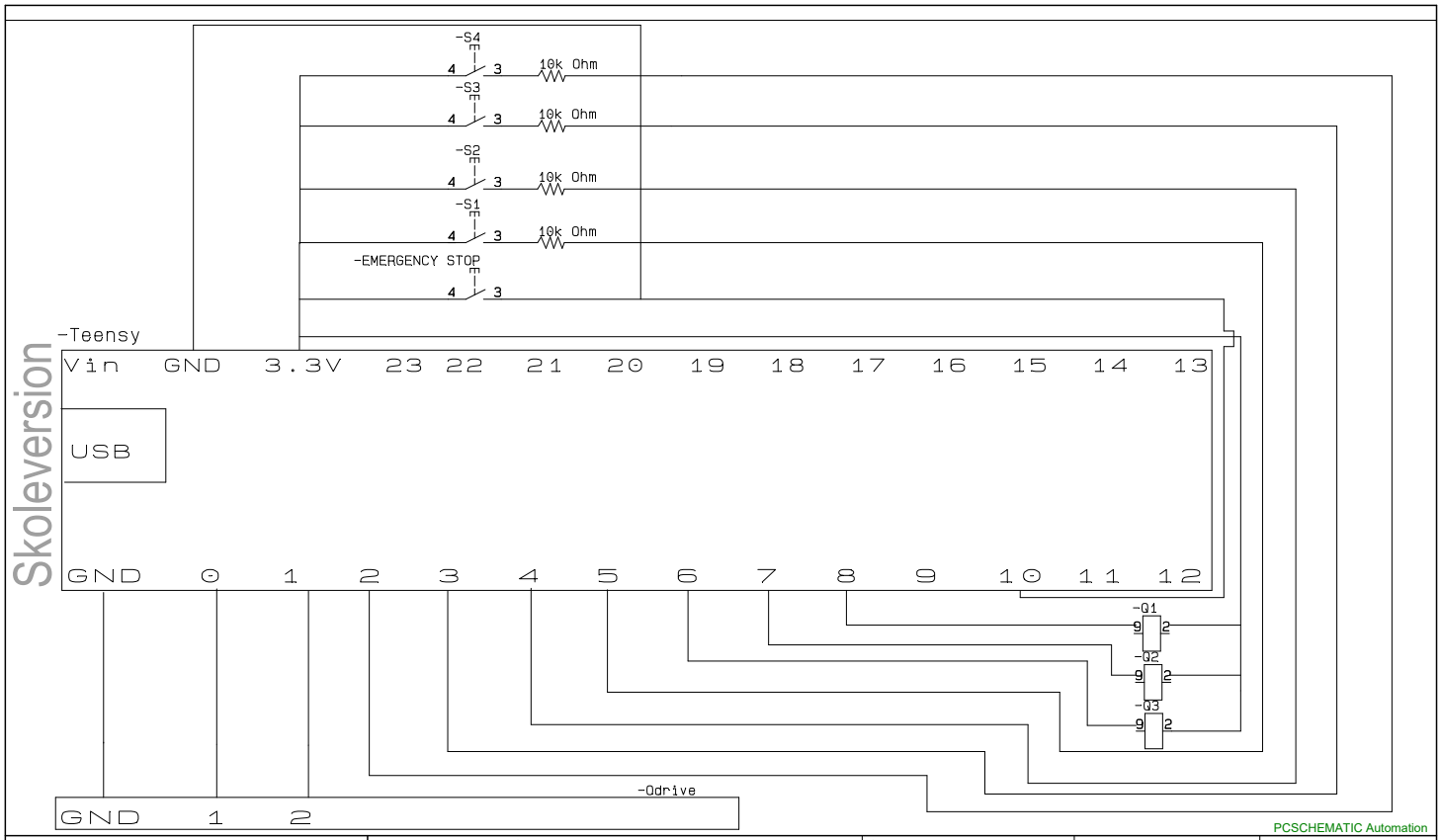
Skoleversion



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**		**Project rev.:**	**Page** 4
Customer:	NTNU	DCC:			Scale: 1:1
Page title:	Jetson Nano	Dwg. no.:		Page rev.:	Previous page: 3
File name:	Pick-And-Sort-Robot	Eng. (proj/page):	Petter	Last print: 19.11.2019	Next page: 5
Page ref.:		Appr. (date/init):		Last edit: 19.11.2019	Total no. of pages: 9

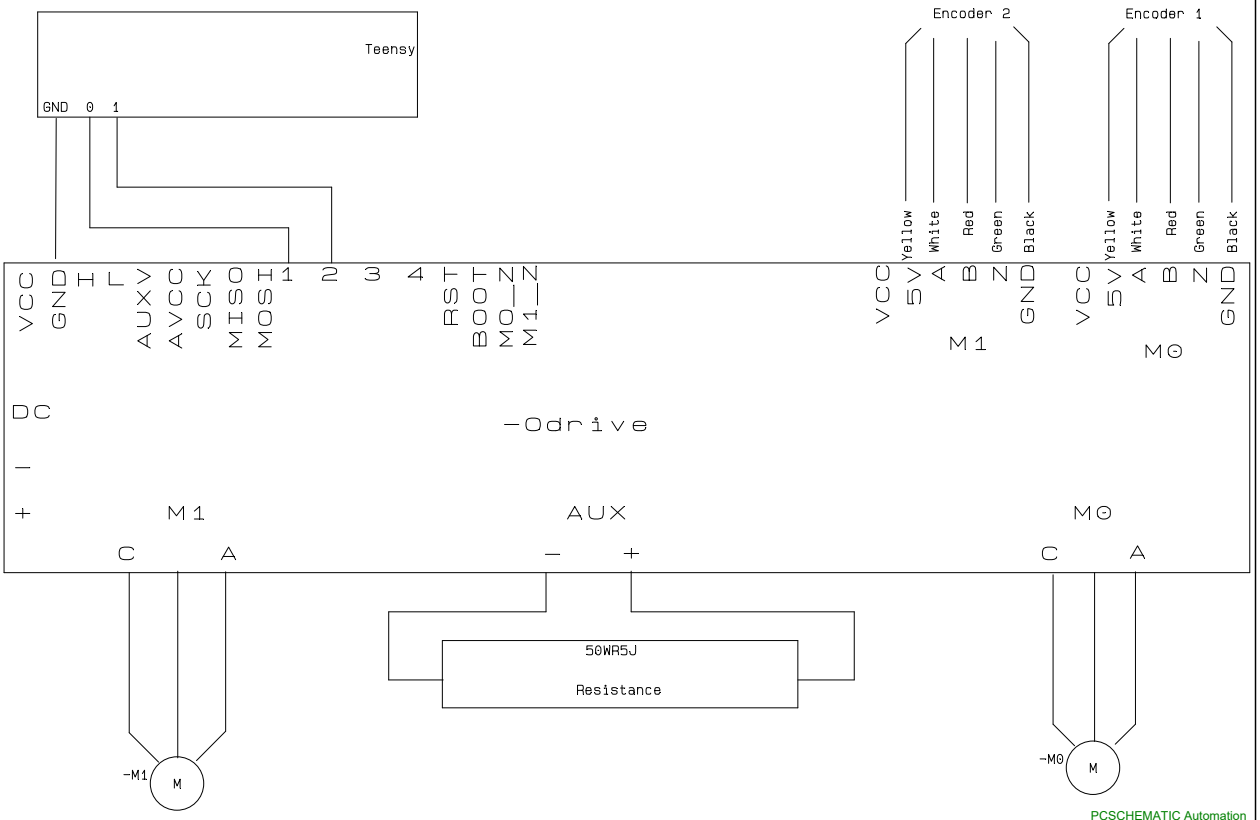


PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**		**Project rev.:**	**Page** 5
Customer:	NTNU	DCC:			Scale: 1:1
Page title:	Teensy	Dwg. no.:		Page rev.:	Previous page: 4
File name:	Pick-And-Sort-Robot	Eng. (proj/page):	Petter	Last print: 19.11.2019	Next page: 6
Page ref.:		Appr. (date/init):		Last edit: 19.11.2019	Total no. of pages: 9

Skoleversion

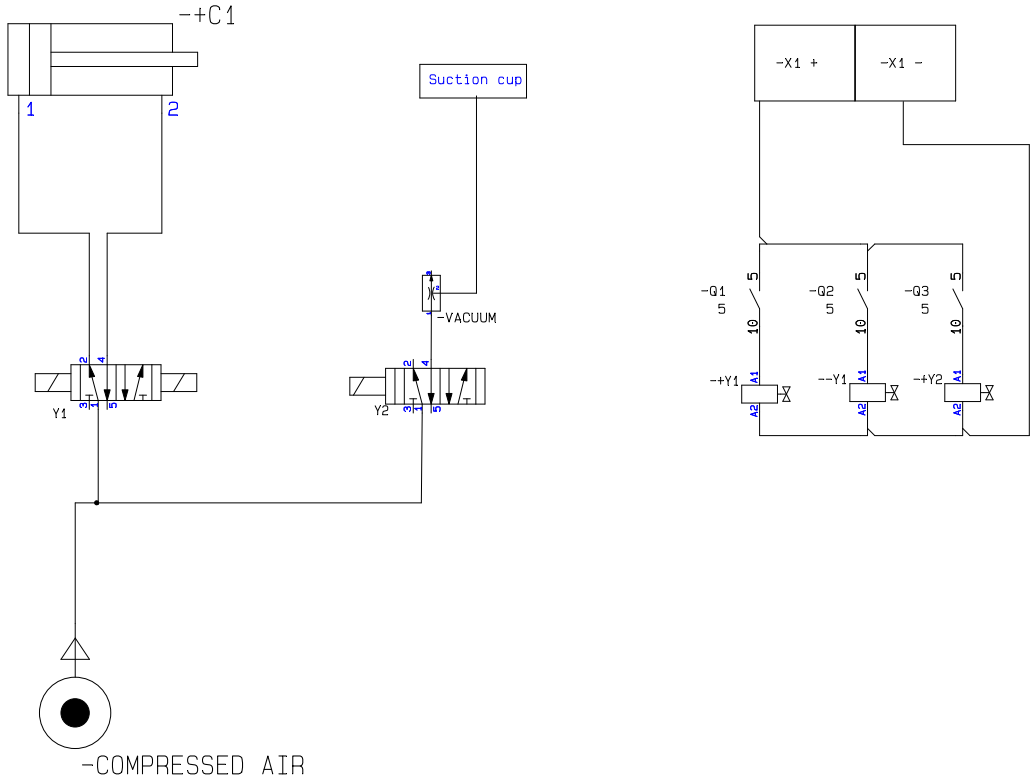


PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**		**Project rev.:**	**Page** 6
Customer:	NTNU	DCC:			Scale: 1:1
Page title:	Odrive	Dwg. no.:		Page rev.:	Previous page: 5
File name:	Pick-And-Sort-Robot	Eng. (proj/page):	Petter	Last print: 19.11.2019	Next page: 7
Page ref.:		Appr. (date/init):		Last edit: 19.11.2019	Total no. of pages: 9

Skoleversion



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Pick-And-Sort-Robot**	**Case no.:**		**Project rev.:**	**Page** 7
Customer:	NTNU	DCC:			Scale: 1:1
Page title:	Pneumatics	Dwg. no.:		Page rev.:	Previous page: 6
File name:	Pick-And-Sort-Robot	Eng. (proj/page):	Petter	Last print: 19.11.2019	Next page:
Page ref.:		Appr. (date/init):		Last edit: 19.11.2019	Total no. of pages: 9

Appendix B

Second appendix

B.1 Source Code

The source code can be found here Github