

Erlend Lillevik  
Andreas Nonslid Håvardsen

# Active control of an ankle using linear actuators and load cells

May 2021

**NTNU**

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Department of ICT and Natural Sciences

**Bachelor's thesis**

**2021**







Erlend Lillevik  
Andreas Nonslid Håvardsen

# Active control of an ankle using linear actuators and load cells

Bachelor's thesis  
May 2021

**NTNU**

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Department of ICT and Natural Sciences



Norwegian University of  
Science and Technology



# Active control of an ankle using linear actuators and load cells

Erlend Lillevik

Andreas Nonslid Håvardsen

May 2021

PROJECT / BACHELOR THESIS

Supervisor 1: Øystein Bjelland

Supervisor 2: Aleksander Larsen Skrede

## **Preface**

This bachelor thesis is written by two students from Automatiseringsteknikk at NTNU Ålesund. The group consists of members with a varying degree of practical experience in the field, as well as varying fields of interest.

The goal of this project is to see how one can use actuators and load cells to actively control an ankle and get useful data from the testing. This type of technology is quite interesting because it can be used to get objective data about a ankle joint and for determining the efficiency of different surgical procedures. Because of this, there are large potential pitfalls when creating such a system.

All of this combined made the thesis quite intriguing, and made working on the project an interesting endeavor for the project members.

## **Acknowledgement**

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

- Our supervisors for all the help and guidance through the project
- Family and friends for supporting us through the semester
- Ålesund Biomechanical Laboratory for equipment and expertise
- Andreas Fagerhaug Dalen for medical knowledge and guidance through the project
- Anders Sætersmoen for ordering and supplying the parts needed for the project

## Summary

This report concerns the development of a prototype system to perform stability tests on an ankle. The purpose of this project is to investigate how to implement such a system, and to make a simulation and visualization of an ankle. The prototype uses four linear actuators and four load cells to pull and measure the load on a tendon. For this, an ankle model with simulated tendons was made. The results from testing the prototype show that the system is capable of moving the ankle in a realistic movement pattern and accurately measure the load. It would however benefit from better software and controller integration. The simulation and the visualization gives a good approximation to a real ankle. It does however use a simplified mathematical model and could benefit from more optimization. Even still, the prototype shows great promise and worked well as a proof of concept.



# Contents

Preface . . . . .	i
Acknowledgement . . . . .	ii
Summary and Conclusions . . . . .	iii
Acronyms . . . . .	2
<b>1 Introductions</b>	<b>8</b>
1.1 Background . . . . .	8
1.2 Problem Formulation . . . . .	9
1.3 Limitations . . . . .	10
1.4 Requirements . . . . .	10
1.5 Structure of the Report . . . . .	11
<b>2 Theoretical basis</b>	<b>12</b>
2.1 Joint . . . . .	12
2.1.1 Ankle . . . . .	12
2.1.2 Ankle tendons . . . . .	14
2.1.3 Ankle joint stability . . . . .	14
2.1.4 Testing joint stability . . . . .	15
2.2 Linear actuator . . . . .	16
2.2.1 Belt driven actuators . . . . .	16
2.2.2 Ball screw actuators . . . . .	17
2.2.3 Rack and pinion actuators . . . . .	18
2.2.4 Controlling the linear actuator . . . . .	19
2.2.5 Voltage clamp . . . . .	19

2.3	Load cell	19
2.3.1	Strain gauge	19
2.3.2	Strain gauge based load cells	20
2.3.3	Load Cell Accuracy Classes	21
2.3.4	Load cell amplifier	22
2.4	Micro controller	22
2.5	Serial communication	23
2.6	State-space	23
2.6.1	System identification	24
2.6.2	Basic control theory terminology	25
2.6.3	PID regulators	25
2.7	Kinematics	28
2.8	Mass-spring-damper systems	29
<b>3</b>	<b>Method</b>	<b>31</b>
3.1	Method	31
3.1.1	Project Organisation	31
3.1.2	Ankle testing rig	32
3.1.3	Ankle model	33
3.1.4	Simulation	34
3.1.5	Visualization	35
3.2	Materials	37
3.2.1	Data collection	37
3.2.2	Construction	37
3.2.3	Ankle model	38
3.2.4	Linear motion	39
3.2.5	Linear actuator controller	40
3.2.6	Load cell	41
3.2.7	Load cell amplifier	42
3.2.8	System controller	43

3.2.9	Communication protocol	44
3.2.10	Power supply	44
3.2.11	Cable and pulley	46
3.2.12	End switches	47
3.2.13	Software and libraries	47
<b>4</b>	<b>Result</b>	<b>49</b>
4.1	Design	49
4.1.1	Main frame	50
4.1.2	Linear actuators	51
4.1.3	Load cells	52
4.1.4	Cable and pulley	53
4.1.5	Complete Design	54
4.1.6	Electrical layout	55
4.2	Build	57
4.2.1	Main frame demo	57
4.2.2	Linear actuators	57
4.2.3	Load cells	58
4.2.4	Cable and pulley	59
4.2.5	Ankle model	60
4.2.6	Electrical	61
4.2.7	Completed build	62
4.3	Implementation	64
4.3.1	Arduino code	64
4.4	Simulation	69
4.4.1	Model	69
4.4.2	Code	74
4.5	Visualization	76
4.5.1	Code	76

<i>CONTENTS</i>	1
<b>5 Discussion</b>	<b>79</b>
5.1 Rig . . . . .	79
5.1.1 Design and materials . . . . .	79
5.1.2 Hardware . . . . .	80
5.1.3 Software . . . . .	80
5.1.4 Testing . . . . .	80
5.2 Simulation . . . . .	81
5.2.1 Different stages throughout the project . . . . .	81
5.2.2 Deciding on level of complexity in the model . . . . .	82
5.3 Visualization . . . . .	83
5.3.1 Different available solutions for visualizing the data . . . . .	83
5.4 Personal Experiences . . . . .	84
5.4.1 Planning . . . . .	84
5.4.2 Division of labor . . . . .	84
5.4.3 Data collection . . . . .	85
5.4.4 Covid-19 . . . . .	85
<b>6 Conclusions</b>	<b>86</b>
6.1 Further work . . . . .	87
<b>7 Appendices</b>	<b>88</b>
<b>Appendices</b>	<b>88</b>
A Code . . . . .	88
B Data-sheets . . . . .	108
C Preproject report . . . . .	150
D Gantt diagram . . . . .	161
E Meeting reports . . . . .	163
F Electrical drawings . . . . .	172
<b>Bibliography</b>	<b>174</b>

## Terminology

**PID** Proportional integral derivative controller

**UI** User Interface, makes it possible to interact with a computer

**CAD** Computer aided design, technology for 3D-design and modelling.

## Notation

$K_p$  Proportional term of a PID controller

$K_i$  Integral term of a PID controller

$K_d$  Derivative term of a PID controller

**kg** System International unit for Kilogram

$\Omega$  Ohm, system International unit for electrical resistance,.

**W** Watts, system International unit for power.

**V** Volts, system International unit for electric potential.

**A** Ampere, system International unit for electric current.

**k** Spring constant used in mass-spring-damper system.

**c** Damping constant used in mass-spring-damper system.

**m** Mass constant used in mass-spring-damper system.

**u** Input(s) to system.

## **Abbreviations**

**IEEE** Institute of Electrical and Electronic Engineers

**Gnd** Ground in electrical circuits

**DOF** Degrees of Freedom, number of configurations for a object

**DC** Direct current

**SISO** Single input single output

**MIMO** Multi input multi output

# List of Figures

2.1	Healthy ankle, seen from above and from the side. . . . .	13
2.2	The most important tendons in the ankle. . . . .	14
2.3	Kuka robot performing tests on a sholder joint. . . . .	15
2.4	Belt driven actuator. . . . .	16
2.5	Ball screw actuator. . . . .	17
2.6	Rack and pinion actuator. . . . .	18
2.7	Compression load cell. . . . .	20
2.8	Tension load cell. . . . .	21
2.9	Beam load cell. . . . .	21
2.10	State-space in a block figure representation . . . . .	24
2.11	A example of a PID regulator used to regulate a basic 2nd order system, made in Matlab . . . . .	26
2.12	The PID regulated plant response at $t = 10$ . . . . .	27
2.13	The PID regulated plant response at $t = 300$ . . . . .	28
2.14	Force-velocity, force-displacement, and impedance translational relation- ships for springs, viscous dampers, and mass . . . . .	30
3.1	Example of aluminium profiles. . . . .	38
3.2	Ankle model. . . . .	38

3.3 RS Pro Electric Linear Actuator ID10, 24V DC. . . . .	40
3.4 RoboClaw 2x15A. . . . .	41
3.5 RS PRO Alloy Steel S Beam Load Cell. . . . .	42
3.6 Seed hx711 load cell amplifier. . . . .	42
3.7 Arduino Mega and Sreed Shield. . . . .	43
3.8 Cosel 1500W power supply. . . . .	45
3.9 VClamp and resistor . . . . .	45
3.10 Petzl Aluminium Pulley . . . . .	46
4.1 Current lab. . . . .	49
4.2 Main frame CAD model. . . . .	50
4.3 Actuators CAD model. . . . .	51
4.4 Load cells CAD model. . . . .	52
4.5 Cable and pulley CAD. . . . .	53
4.6 Complete design CAD. . . . .	54
4.7 Wiring diagram control circuit. . . . .	55
4.8 Wiring diagram main circuit. . . . .	56
4.9 Main frame demo. . . . .	57
4.10 Actuators build. . . . .	58
4.11 Load cells build. . . . .	58
4.12 Cable and pulley build. . . . .	59
4.13 Modified ankle model. . . . .	60
4.14 Modified ankle model . . . . .	61
4.15 Electrical connections. . . . .	62
4.16 Ankle model during testing . . . . .	62



4.17 Completed build . . . . .	63
4.18 Function from Arduino code in prototype without comments and compressed, function: PID() . . . . .	66
4.19 Function from Arduino code in prototype without comments and compressed, function: roboclawMovement() . . . . .	67
4.20 A typical two boned joint . . . . .	69
4.21 A lateral view of a elbow . . . . .	70
4.22 Kinematic model of ankle with two degrees of freedom . . . . .	70
4.23 Kinematic model of ankle limited to one degree of freedom . . . . .	71
4.24 Mass-spring-damper representation of the simplified ankle . . . . .	72
4.25 The input and ankle height generation code from the simulation, compressed and without comments . . . . .	74
4.26 The state-space model generation in Matlab, compressed and without com- ments . . . . .	75
4.27 The code the simulation is run, " <b>Y</b> " stores the simulation results, " <b>sys</b> " is the state-space model, " <b>u</b> " is the input and " <b>t</b> " is the simulation time references, without comments . . . . .	75
4.28 The line bundles being generated, each line generates a line bundle with it's own color and points array, without comments . . . . .	76
4.29 The way the <b>a1</b> point array is filled, without comments . . . . .	76
4.30 The geometric logic used to find all points for visualization, where every cor- ner of the two rectangle's angles are $\frac{\pi}{2}$ rad . . . . .	77
4.31 The code used to calculate the geometric points to "draw" for the visualiza- tion, without comments . . . . .	78

# List of Tables

3.1	Arduino models . . . . .	43
4.1	Tendons resistance . . . . .	61

# Chapter 1

## Introduction

This report will review all the different stages of development for the project. All principles that have been considered will be drawn out, as well as what was chosen as the final implementation and the theory behind this.

### 1.1 Background

Hospitals are becoming increasingly more technological, and medical procedures and operations are becoming increasingly higher in quality. By combining engineering and medical expertise using modern technology, researchers and doctors can do experiments together that further improves the quality of these procedures. One medical field that is implementing new technology to increase the knowledge and quality offered is the field of orthopedics.

Orthopaedic surgery or orthopaedics, is the branch of surgery concerned with conditions involving the musculoskeletal system. Orthopaedic surgeons use both surgical and nonsurgical means to treat musculoskeletal trauma, spine diseases, sports injuries, degenerative diseases, infections, tumors, and congenital disorders. [15]

To achieve this the orthopedic department of the hospital in Ålesund (Ålesund Sjukehus) and NTNU have created a collaborative project called Ålesund Biomechanical Laboratory. The goal of this project is for engineers and researchers from NTNU and orthopedic surgeons from Åle-

sund Hospital to use robots in researching better surgical methods. This kind of robot lab is the first of its kind in Scandinavia.

The medical research currently conducted is based on testing new orthopedic surgical methods on corpse samples. To test the results of the new operating methods, for example joint stability, a Kuka KR-6 industrial robot with a 6DOF load cell is used. The next step is to also include active muscle management in stability tests.

This project will therefore aim to create a system to be able to actively control a ankle joint for usage in stability tests using linear actuator and load cells. The focus will lie in making a robust and modular system.

## 1.2 Problem Formulation

The basic problem for this project is to create a system that is able to actively control a ankle joint and make the ankle have a realistic movement pattern. This should be done by pulling on the tendons located in the ankle joint. The system should also be able to measure the load being applied to the specific tendon. Furthermore, the system should be powerful and modular enough so that it can later be used for testing other types of joints.

Additionally there should be a simulation that can be run separately from the prototype, and a visualization of the simulation results.

### Problems to be addressed

- Designing a rig
- Design and implement a controller for the linear actuators
- Implement a system for reading the load cell data
- Creating a ankle model to be used for testing
- Create a simulation and visualization for testing without the prototype

### 1.3 Limitations

In this particular project the limitations mainly derives from the group members limited knowledge of the medical aspects related to the task.

Another limitation is the current Covid-19 pandemic. Since the laboratory is located at a hospital, the regulations are strict and changes rapidly based on the current infection rate. This adds a lot of uncertainties and means that the group can lose access to the lab overnight. Therefore the group will try to do the majority of the work on the project at NTNU, even though this can have a impact on the result. Another effect of the pandemic is the availability and delivery times of parts needed for the project.

No matter what the effects of the pandemic, the main focus of the group will always be that of safety.

### 1.4 Requirements

The requirements for the project is as follows;

- Design a modular system
- Create a functioning prototype
- Make the ankle have a realistic movement pattern
- Design a appropriate model for the simulation with regards to ankle testing
- Have a visualization that clearly highlight the simulation results

## 1.5 Structure of the Report

The rest of the report is structured as follows.

**Chapter 2 - Theoretical basis:** Chapter two gives an introduction to the theoretical background needed in this report, with proper references to each literature reference used. Theoretical basis may involve concepts, definitions, methods, standards and theory to explain specific system behavior.

**Chapter 3 - Method:** Contains a description of the methodology and materials that were considered throughout the project. It also contains important information of what needs to be considered for building rig for active ankle control. In the end of the chapter there is a description of how the tests will be executed.

**Chapter 4 - Result:** Contains a description of the finished prototype, the material for building it and the software developed for the project. In addition to this it contains results from the tests.

**Chapter 5 - Discussion:** An assessment of the chosen methods and achieved results. As well as limitations, changes done and sources of error.

**Chapter 6 - Conclusions:** This chapter present an overall conclusion and the experiences gathered trough out the project.

# Chapter 2

## Theoretical basis

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

### 2.1 Joint

Joints are the areas where 2 or more bones meet. Most joints are mobile, allowing the bones to move. Joints consist mainly of the following: [1]

- Cartilage. This is a type of tissue that covers the surface of a bone at a joint. Cartilage helps reduce the friction of movement within a joint.
- Ligaments. Strong ligaments (tough, elastic bands of connective tissue) surround the joint to give support and limit the joint's movement. Ligaments connect bones together.
- Tendons. Tendons (another type of tough connective tissue) on each side of a joint attach to muscles that control movement of the joint. Tendons connect muscles to bones.

#### 2.1.1 Ankle

In medicine, the ankle is defined as the joint that connects the bones in the lower leg to the foot bones. It can be divided into two parts: the upper and lower ankle. [21]

The upper ankle allows us to move our feet upwards, downwards, and a little to the side. It is made up of three bones:

- The tibia (shinbone): the main bone in the lower leg.
- The fibula (calf bone): a second, thinner bone on the outer side of the lower leg.
- The talus (anklebone): the foot bone that connects to the shinbone and calf bone.

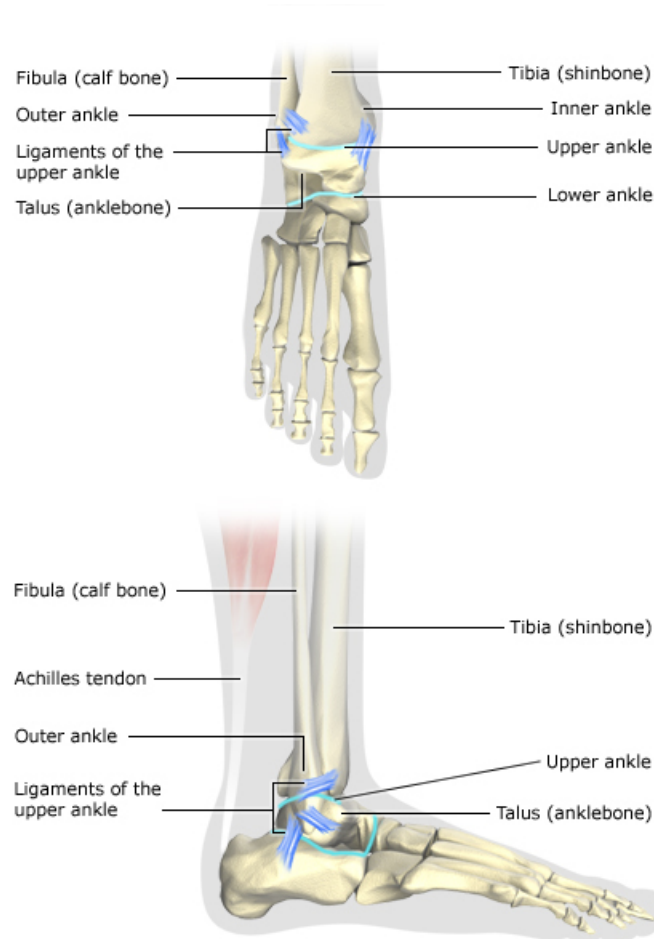


Figure 2.1: Healthy ankle, seen from above and from the side.

Source: [www.ncbi.nlm.nih.gov](http://www.ncbi.nlm.nih.gov)

The lower ankle connects the anklebone to the bones in the tarsus (the midfoot and hind-foot) and the heel bone. It does not move as much as the upper ankle. The lower ankle allows the foot to tilt to the side a bit and also turn inwards and outwards. Turning your foot outwards is known as pronation, and turning it inwards is called supination. [21]



### 2.1.2 Ankle tendons

The ankle joint is also supported by nearby tendons. The large Achilles tendon is the most important tendon for walking, running, and jumping. It attaches the calf muscles to the calcaneus (heelbone) and is used for raising the heel. The posterior tibial tendon attaches one of the smaller muscles of the calf to the underside of the foot. This tendon helps support the arch and allows us to turn the foot inward. The anterior tibial tendon allows us to raise the foot. Two tendons run behind the outer bump of the ankle. These two tendons, called peroneus brevis and peroneus longus, help turn the foot down and out. [18]

In summary:

- Achilles, raises the heel.
- Posterior tibial, turns the foot inward.
- Anterior tibial, raises the foot.
- Peroneus brevis, turns the foot down and out.

### 2.1.3 Ankle joint stability

The ankle is not as mobile as some other joints, however due to the need for the foot and ankle to move both up and down but also inwards and outwards, there is an innate tradeoff for stability. The primary stabilizers of the ankle are the ligaments around the ankle, however the secondary stabilizers are the tendons that cross the ankle joint. This means that when the ligaments are injured, the tendons often have to work harder to

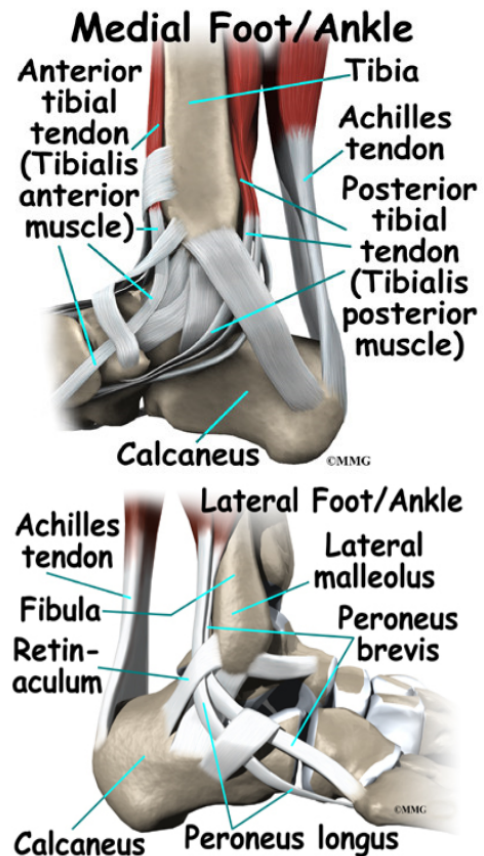


Figure 2.2: The most important tendons in the ankle.

Source: [www.eorthopod.com](http://www.eorthopod.com)

maintain stability, thereby putting them at risk for injury as well. [2]

### 2.1.4 Testing joint stability

At Ålesund Biomechanical Laboratory the current system for testing joint stability is performed using a robot to determine the biomechanical stability. As of now, the tests are run on a shoulder joint. The tests are performed by applying a force along two axes, horizontally and vertically, and at the same time measuring how far the joint ball moves. These tests are performed four times. The first test is performed on a intact and "healthy" carcass to determine a base measurement, later the test is performed on the carcass after the surgeons have inflicted it with an injury. The remaining tests are performed after the surgeons have corrected the injury by two different surgical methods. This kind of testing provides data that is used to evaluate new and existing surgical methods.



Figure 2.3: Kuka robot performing tests on a sholder joint.

Source: [www.tu.no](http://www.tu.no) - Jørn-Arne Tomasgard

## 2.2 Linear actuator

Linear actuators are a type of actuator that convert rotational motion in motors into linear or straight push/pull movements. There are several ways of achieving this conversion. The most commonly used techniques are either by belt-driven actuators, ball screw driven actuators or rack and pinion driven actuators. These three different actuator implementations all apply linear motion by rotary motors, but they differ in strengths and limitations. [7]

### 2.2.1 Belt driven actuators

A belt-driven actuator converts rotary motion to linear motion by means of a timing belt connected between two pulleys at either end of the drive. The belt contains teeth that mesh with the pulleys to efficiently transfer torque and prevent slipping. A belt drive is normally enclosed within a body with the load carrying carriage riding on top along rails. [14]



Figure 2.4: Belt driven actuator.

Source: [www.isotechinc.com](http://www.isotechinc.com)

Pros:

- High speeds and acceleration
- High duty cycle

- Low noise levels

Cons:

- Not as accurate as other types
- Low load carrying capability

### 2.2.2 Ball screw actuators

A ball screw actuator consists of a ball screw and ball nut with recirculating bearings that roll in the grooves formed by the screw and nut. [7] Rotating the screw will move the nut up and down the length of the screw.

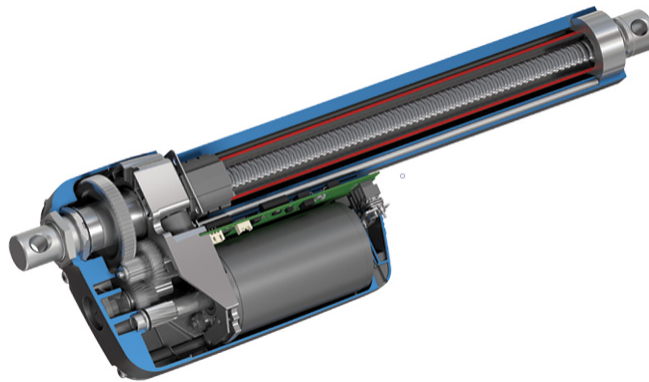


Figure 2.5: Ball screw actuator.

Source: [www.thomsonlinear.com](http://www.thomsonlinear.com)

Pros:

- High load carrying capability
- High levels of accuracy
- High mechanical efficiency
- Reduced power requirement

Cons:

- More expensive than other types
- Prone to be noisy
- Slower than other types

### 2.2.3 Rack and pinion actuators

Rack and pinion gear sets consist of a circular gear called a pinion that engages the teeth of a linear gear called a rack. [7] Rotating the pinion will move it up and down the rack.

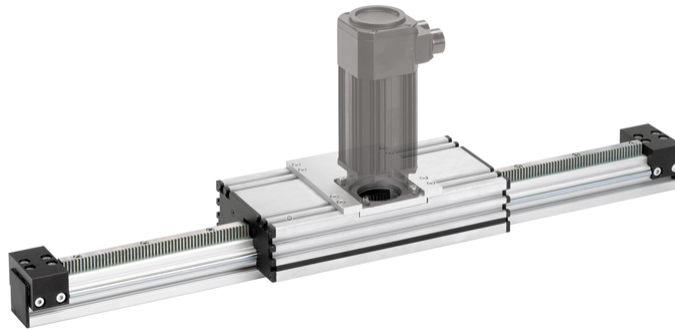


Figure 2.6: Rack and pinion actuator.

Source: [www.rk-rose-krieger.com](http://www.rk-rose-krieger.com)

Pros:

- High speeds
- High accuracy over long travel lengths
- Few components

Cons:

- Need active lubrication
- High friction
- Need clearance, therefore backlash is a possibility

## 2.2.4 Controlling the linear actuator

To control a linear actuator, a DC-motor driver is used. A motor driver 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. These can either be a stand-alone unit or part of the linear actuator itself. [3]

## 2.2.5 Voltage clamp

A voltage clamp is a circuit used with a motor controller to divert excess energy away from the control circuitry and power supply to prevent damage. The primary purpose of a voltage clamp is to protect a motor controller's power supply from the regenerative power created when a motor is stopped or slowed down quickly. A motor spins when power is applied to it, however it also acts as a generator when slowing down or suddenly stopped. This energy opposes the flow of current to the motor and is directed back to the motor controller and power source. Switching power supplies are not designed to handle this power surge. A voltage clamp detects this power surge as a rise in voltage and sends the excess current to a resistor where it's turned in to heat. [16]

## 2.3 Load cell

A load cell is a sensor or a transducer that converts a load or force acting on it into an electronic signal. This electronic signal can be a voltage change, current change or frequency change depending on the type of load cell and circuitry used.[13] The inner working of a load cell differs based on the load cell that you choose. There are hydraulic load cells, pneumatic load cells, and strain gauge load cells. Strain gauge load sensors are the most commonly used among the three. [5]

### 2.3.1 Strain gauge

A strain gauge is a device that measures electrical resistance changes in response to and proportional of the strain applied to the device. The most common strain gauge is made up of very

fine wire, or foil, set up in a grid pattern in such a way that there is a linear change in electrical resistance when strain is applied in one specific direction, most commonly found with a base resistance of  $120\Omega$ ,  $350\Omega$ , and  $1,000\Omega$ . [4] This change in resistance then leads to a change in output voltage when a input voltage is applied. [5]

### 2.3.2 Strain gauge based load cells

#### Compression Load Cells

Compression load cells measure the pushing or squashing force, and are typically installed underneath what is to be measured or weighed. Compression load cells are unidirectional, only designed to measure the downward compression. They do this with strain gauges attached to the body of the load cell. When under compressive load, the body of the load cell deforms slightly. [12] The strain gauges, which are bonded to the load cell body also distort, giving a change in resistance.



Figure 2.7: Compression load cell.

Source: [www.eilersen.com](http://www.eilersen.com)

#### Tension Load Cells

Tension load cells (also known as S-beam load cells) are predominantly used for measuring tension or pulling force. Many can also measure compression, which makes them very versatile due to having bi-directional sensitivity. When experiencing a load, the main body of the load cell deforms slightly. [9] The strain gauges, which are bonded to the load cell body also distort, giving a change in resistance.

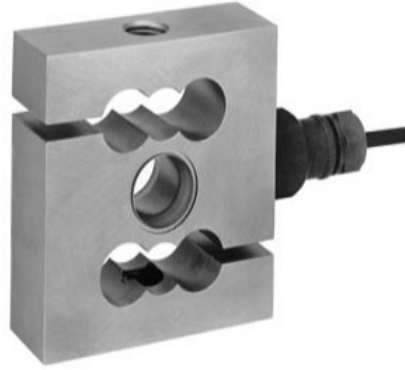


Figure 2.8: Tension load cell.

Source: [www.synectic.co.uk](http://www.synectic.co.uk)

### Beam Load Cell

Beam load cells operate as simple cantilevers that flex slightly when subjected to a force or weight. They are typically fixed at one end and free at the other, behaving much like a diving board. When a load is applied, the body of the load cell will flex due to the elastic properties of the metal material that it is made from. [8] The strain gauges, which are bonded to the load cell body also distort, giving a change in resistance.



Figure 2.9: Beam load cell.

Source: [www.pavonesistemi.com](http://www.pavonesistemi.com)

### 2.3.3 Load Cell Accuracy Classes

Load cells are highly accurate devices but have different accuracy classes, these classes can give more information on precisely how accurate the load cell is which in turn can help with selec-



tion for certain applications. [6]

**D1 – C2:** This load cell accuracy class is relatively low. It is sufficient for example applications in simple building materials scales which are used to weigh cement, sand or water as well as non-trade applications.

**C3:** The accuracy class C3 offers an accuracy of 0.0230% and is the most common accuracy class. Typical applications include; belt scales, platform scales and other industrial weighing devices.

**C4 – C5:** The accuracy class C4-C5 offers an accuracy of 0.0174% - 0.0140%. This accuracy class is higher and is usually used for shop counter scales.

### 2.3.4 Load cell amplifier

A load cell amplifier is a device that can increase the strength of the signals coming from a load cell. Sometimes the signals produced by the load cell can be too weak to read directly. A load cell amplifier resolves this issue by taking the signal and amplifying it, resulting in a higher strength output signal that can be read directly. [10]

## 2.4 Micro controller

A microcontroller is an integrated circuit (IC) device used for controlling other portions of an electronic system, usually via a microprocessor unit, memory, and some peripherals. These devices are optimized for embedded applications that require both processing functionality and agile, responsive interaction with digital, analog, or electromechanical components. [11]

## 2.5 Serial communication

Serial communication is a simple means of sending data to long distances quickly and reliably. The most commonly used serial communication method is based on the RS232 standard. In this standard, data are sent over a single line from a transmitting device to a receiving device in bit serial format at a prespecified speed, also known as the Baud rate, or the number of bits sent each second. Typical Baud rates are 4800, 9600, 19200, 38400. [20]

Serial communication is a form of asynchronous data transmission where data are sent character by character. Each character is preceded with a start bit, seven or eight data bits, an optional parity bit, and one or more stop bits. The most commonly used format is eight data bits, no parity bit, and one stop bit. The least significant data bit is transmitted first, and the most significant bit is transmitted last. [24] For this a minimum of three lines are used: transmit (TX), receive (RX), and ground (GND). Some high-speed serial communication systems use additional control signals for synchronization. [20]

## 2.6 State-space

The following was found in Norman S. Nise's book, "Control systems engineering, 7th edition" [25].

A state-space model is a model that uses state variables to describe a system through a set of first-order differential equations, instead of a higher order differential equation. The typical state-space representation of a linear system is written as:

$$\dot{x} = Ax + Bu \quad (2.1)$$

$$y = Cx + Dy \quad (2.2)$$

Where  $A$  is a  $n$  matrix where  $n$  is the number of states.  $B$  is the  $n$  input matrix where  $p$  is the number of inputs.  $C$  is the  $q$  output matrix where  $q$  is the number of outputs.  $D$  is the feedthrough matrix and is a  $q$  matrix. In system models without feedthrough,  $D$  is the zero matrix. All these

matrices and/or values can vary with time. A block figure representation of the state-space model can be shown in the figure below:

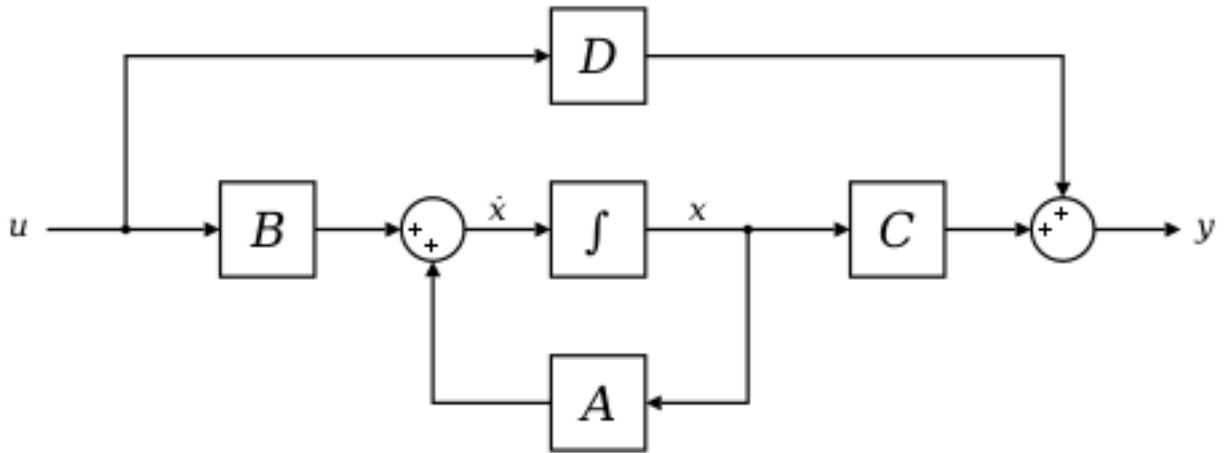


Figure 2.10: State-space in a block figure representation

Source: [https://en.wikipedia.org/wiki/State-space\\_representation](https://en.wikipedia.org/wiki/State-space_representation)

### 2.6.1 System identification

The following was found in Norman S. Nise's book, "Control systems engineering, 7th edition" [25].

System identification is a general way to gain a model through experimental means. There are many ways to identify a system experimentally, but the general rule is that it always includes monitoring the systems response to inputs. The step response is typically used and can for example see if the system is a close match to a first-order or second-order system. From observation one can then measure specific parts of the response to reverse engineer the system approximately. This is often used on systems that are not in ideal circumstances or generally complex systems to find a practical model instead of an overly complex and computationally impossible solution.

Different system identification cases can generally be categorized into three groups, the white box, grey box and black box models. The white box cases are where you have complete information and could build it based on first principles, for example Newton equations. These are the cases where usually the models can become overly complex and even impossible to ef-

fectively use. The black box cases are the systems where we have no information between input and output. The grey box model is a middle-ground between these two cases.

### 2.6.2 Basic control theory terminology

The following was found in Britannica's definition of control systems[17].

In control theory there are many ways to control a system. Generally these can be categorized into two main groups, the open-loop and closed-loop systems. The system is the whole process with its controllers, inputs and so on. The process to be controlled is called the plant. The actuator is the input into the plant. The actuator isn't the input, but rather the response of a error through a regulator, where the error is the difference between the output and the input (desired value).

Another way to categorize different types of systems is by splitting into SISO (Single input single output) and MIMO(Multi input multi output) systems. The difference is obviously the amount of inputs and outputs, but also the ramifications of multiple inputs and outputs are significantly more difficult equations for solving, especially by hand.

The two main ways of representing systems in control theory is in the frequency domain with transfer functions, and in state-space. A transfer function is a function describing the relation between its input and the output. The use-cases vary, and there are different strengths to using either representation, but a important limitation of the transfer functions is that they are only usable with SISO systems.

### 2.6.3 PID regulators

The following was found in National Instrument's page on PID-Control[22].

PID (Proportional Integral Derivative) regulators are a type of control regulators that subtracts the feedback from the input creating a error value as the input to the proportional gain,

integrator and derivative and using the responses into as input to the actual plant (system being regulated). A PID controller can contain any combination of the three control elements, for example a PD controller with a proportional and derivative part, but without a integral part. This would simply mean setting the integral gain equal to 0. A example of a PID regulated plant is shown in fig. 2.11.

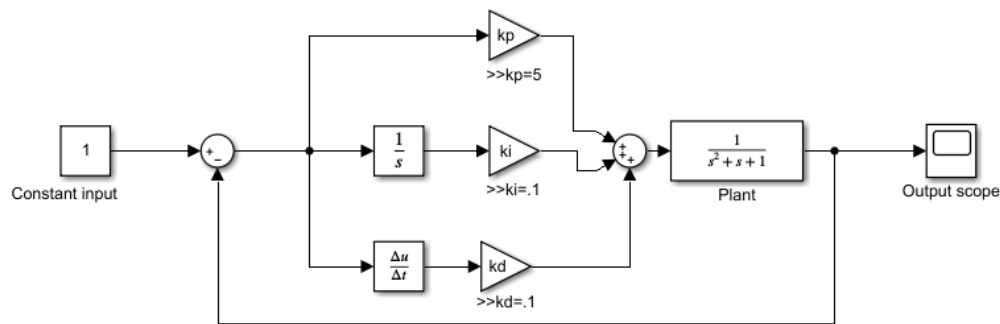


Figure 2.11: A example of a PID regulator used to regulate a basic 2nd order system, made in Matlab

The way we would expect the response to be here would be a summation of the responses from the different parts of the PID regulator.

1. Proportional feedback corrects the input proportionally by the output. With only this type of control the output will never reach the exact desired value since the applied correction approaches 0 as the output reaches the desired value. In this example, the expected output from the proportional controller would be a increase of 5, which will decrease quickly when reaching the desired value, but still overshooting since the initial speed will be quite large, and also therefore oscillate.
2. Integral feedback accumulates the error and grows bigger when there is continuous error. This removes steady-state (value at  $t = \infty$ ) error since if there is a error, the integral feedback will gradually grow bigger to reduce the error until it is exactly 0. The expected response from the integrator would be a small value, constantly increasing for a quite long time until finally the error is equal to 0.
3. Derivative feedback depends on the error's rate of change. If the error is quickly reaching zero, the derivative feedback will slow it down, reducing the overshoot. The expected

response in this example would then be a large initial value comparatively to the rest of the response, but still small in comparison to the proportional response. Therefore there will still be overshoot, but it will be somewhat reduced and continuously work against the overshoots the proportional response will have.

The resulting figures at  $t = 10$  and  $t = 300$  is shown in figures 2.12 and 2.13.

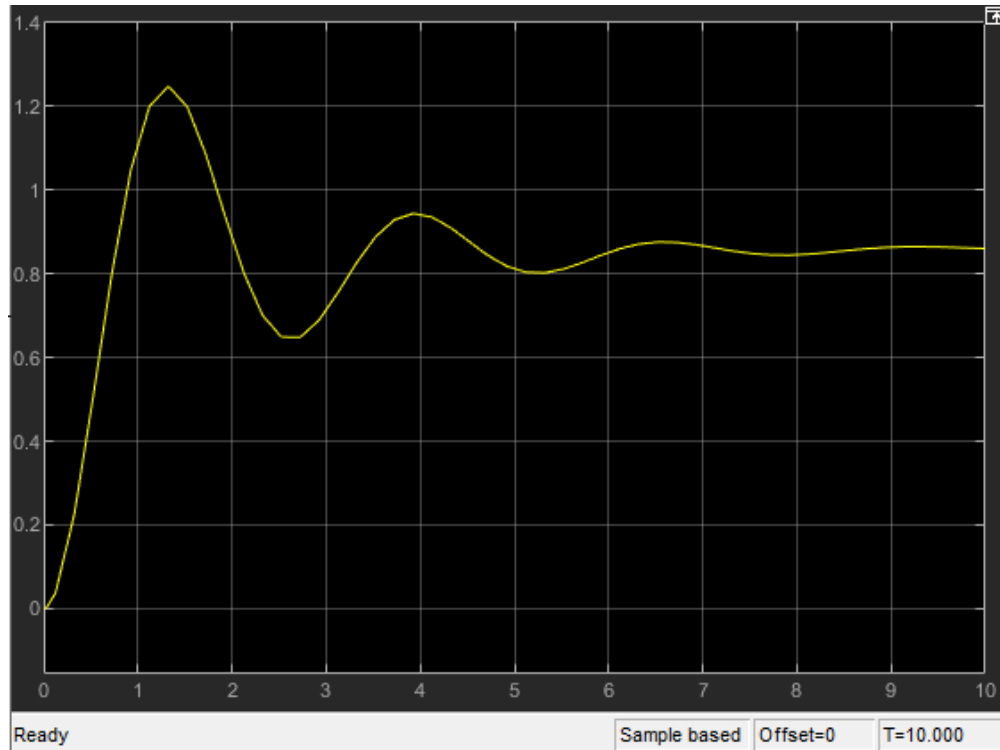


Figure 2.12: The PID regulated plant response at  $t = 10$

In figure 2.12 the proportional and derivative responses are highlighted as the response quickly grows towards the desired value, and overshoots less and less until finally settling at somewhere just above 0.85 which is not too far off, but still not the desired value. It is also very clear that the proportional and derivative responses cancel each other out since the integrator has not yet been able to grow very large, and the response is settling on a more or less constant value.

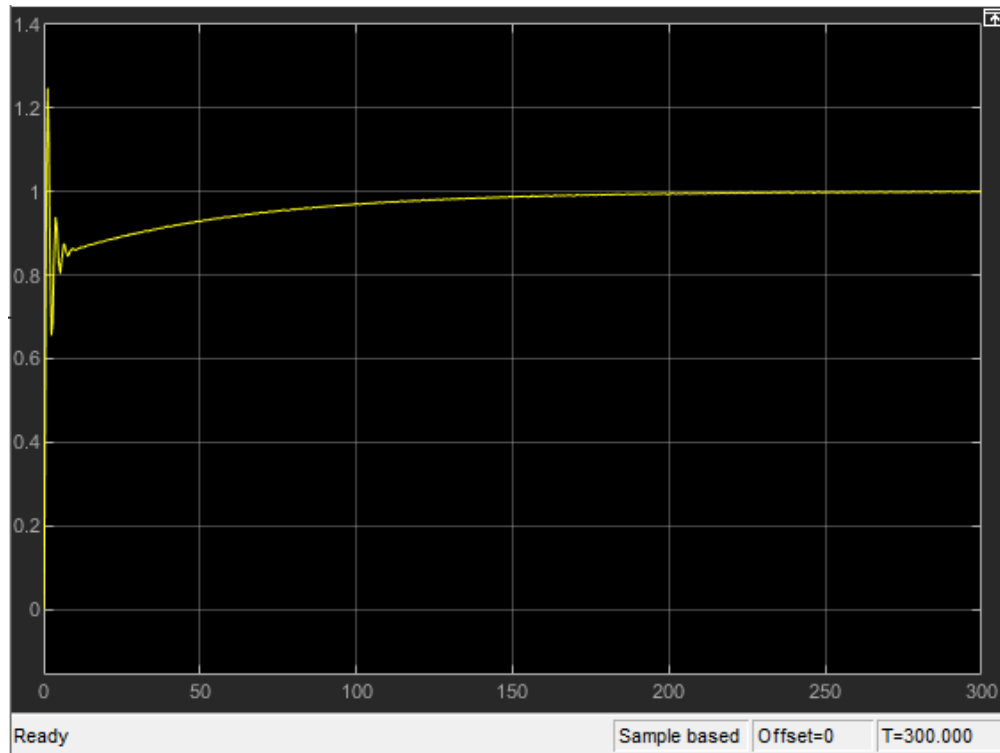


Figure 2.13: The PID regulated plant response at  $t = 300$

In figure 2.13 the integral response is highlighted, since at around  $t = 10$  the derivative and proportional responses started cancelling each other out leaving a steady-state error for the integrator to get rid off. Slowly the constant error makes the integral response grow and reduce the error towards 0.

## 2.7 Kinematics

Kinematics is the study of the motion of points, objects, and groups of objects without considering the causes of its motion [23]. This branch of physics is used to then understand the motion of a object or set of objects, and be able to use this understanding afterwards to control it or predict it using forces and applying other fields of physics on the model or representation. The understanding of the motion of a object also means understanding it's limits, and being able to then use these limits to further simplify or reevaluate a given system. A limit in how many directions a object can move independently is called the degree of freedom of said object.

A example that would be relevant to this project could be a elbow joint that can only rotate in one direction, albeit with positive or negative velocity, which means it has one degree of freedom [27]. It has a circular motion, meaning that it will have a velocity that changes in two axis, but they are co-dependant. This therefore mean that the system only has one degree of freedom. Other kinematic analysis that could be done would be how far the elbow would have to rotate to end in a specific angle.

## 2.8 Mass-spring-damper systems

The following was found in a article written by Prof. Larry Francis Obando called "Mass-Spring-Damper System Dynamics" [26].

A mass-spring-damper system is a system that consists of a combination of masses, springs and dampers.

These components each have specific properties. These properties are defined as follows:

- Mass is the object or elements innate ability to resist change in motion
- Springs are a elastic type system component able to deflect under load, and store up energy.
- Dampers are components that is restraining of vibratory motion, such as mechanical oscillations, noise, and alternating electric currents, by dissipation of energy.

Mathematically speaking the way these components function in a system is as shown in figure 2.14



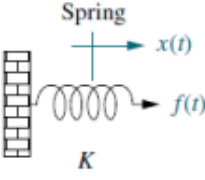
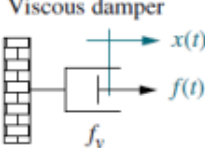
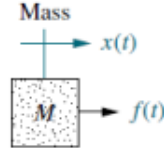
Component	Force-velocity	Force-displacement	Impedance $Z_M(s) = F(s)/X(s)$
	$f(t) = K \int_0^t v(\tau) d\tau$	$f(t) = Kx(t)$	$K$
	$f(t) = f_v v(t)$	$f(t) = f_v \frac{dx(t)}{dt}$	$f_v s$
	$f(t) = M \frac{dv(t)}{dt}$	$f(t) = M \frac{d^2x(t)}{dt^2}$	$Ms^2$

Figure 2.14: Force-velocity, force-displacement, and impedance translational relationships for springs, viscous dampers, and mass

Source: <https://dademuch.com/2018/12/28/mass-spring-damper-system-dynamics/>

Therefore when analyzing a system consisting of all three, the system will work with regards to both position, velocity and acceleration. The mass resists acceleration, the damper resists velocity and the spring resists displacement.

# Chapter 3

## Methods and materials

### 3.1 Method

#### 3.1.1 Project Organisation

The group will have one project leader and one secretary. As the team only consists of two members, these are the only two roles in the group. Once a week the project leader and the secretary will book a meeting and write a summary of the previous week. Even though the group has a leader, the group will work as a flat organization where decisions and agreements are taken in unison. Each team member will be responsible for one or more main tasks, this to evenly distribute the work.

Every other week the group will have meeting with the supervisors. During these meetings, the group will discuss the progress, solutions to possible challenges, and suggestions for possible improvements. In the early stages of the project, the group will establish a plan based on the task provided by the employer. To ensure a good result and efficient approach a project plan has been made. This plan contains information about when subtasks will start and be completed. A short version of the plan is listed below. The thesis will be written in parallel to the tasks.

- Make a detailed concept drawing of the total system. This drawing will contain the measurements and basic design. This will give the team members an overview of the tasks needed to be done and a idea of what the result should be in the end. The concept draw-

ing will be made as a 3D-CAD model.

- Select the needed components for the project.
- Test the individual components before configuring and assembling the system.
- Perform a complete system check.
- Develop the necessary control software to control the actuators.
- Design and calibrate control software specifically for ankle testing.
- Design a mathematical ankle model, and use it for simulating a walking motion.
- Create visualization for the simulation results.

### **3.1.2 Ankle testing rig**

To design and build a testing rig for ankle joint stability, several elements need to be taken into consideration. The general specifications are explained beneath.

#### **Rig design**

The rig needs to be designed in a way suited for the facilities found at Ålesund Biomechanics Laboratory. The laboratory has a limited amount of space, this means that the rig needs to be as compact and modular as possible. As the laboratory is also being used to perform other tests, the rig needs to be designed in a way that its not a nuisance to the personnel performing these tests.

The rig needs to be able to handle heavy loads. It must be capable of withstanding the loads that are exerted on the large tendons in the body, i.e. the Achilles tendon that can bear loads in excess of 3500N. [19]

#### **Linear motion**

For controlling the tendons, a method for providing motion is needed. A tendon only has one degree of freedom, thus the only movement needed is extending and retracting. When choosing

the method for creating the linear motion there are two main criteria that needs to be met. First criteria being accuracy in operation. The second criteria is being able to handle heavy loads. For controlling the tendons, the accuracy needs to be high enough to be able to make small adjustments, at the same time the maximum load abilities needs to be in the region of 3500N. [19]

For generating this motion, some kind of actuator is required. To find a suitable actuator, it is important that it is capable of meeting the requirements for accuracy and force needed. When a suitable actuator has been selected, a compatible motor driver is necessary for operation.

### **Measuring the load**

For measuring the load that is applied to the tendons, some sort of load cell is required. Since the tendons behave like a rope, the load cell only needs to be able to measure pulling force. When choosing the type of load cell, there are two main criteria that needs to be met. First criteria being accuracy, the second criteria is being able to handle high loads. For measuring the load on the tendons, the accuracy needs to be high enough measure small differences in pulling force, at the same time the maximum load capabilities needs to be the same or higher than the chosen actuator.

### **System controller**

For controlling the system, some kind of microcontroller is needed. The suitable controller would need to be able to read and write to all the components in the rig, these signals may be both analog, digital and serial communication. The controller would therefore need to have enough inputs and outputs to match the other chosen components. It would also be beneficial if the controller used a well-established programming language, and preferably something with supported libraries for the chosen components.

### **3.1.3 Ankle model**

To be able to test if the rig is able to perform ankle tests, a model of an ankle is needed. This model needs to be as similar as possible in form and function to a real ankle. The model would

also need to be equipped with some form of tendons. The model used for testing purposes should also be affordable and easily modifiable. It would also be beneficial if the model was not easily broken or distorted if problems with the rig were to arise during testing.

### **3.1.4 Simulation**

The simulation of a ankle is initially a highly complex task. To simulate a system one first needs to approximate a ankle as a physical system for analysis. To do this, the separate parts of the ankle need to be defined into the model or ignored for simplification. Both the specific skeletal structure of the ankle and the general complexity from nerves, muscle and other anatomical factors complicates the system dramatically. Reducing the number of factors simplifies the system down to a system that is possible to analyze using fundamental laws of physics and generate models from.

#### **Creating a model**

There were five steps to create the model used in the simulations in this project.

1. Simplify the anatomical complexities.
2. Create a kinematic model.
3. Transform the kinematic model into a analyzable physical system.
4. Find the equation(s) that describes the physical system.
5. Translating the equation(s) into a state-space model representation of the system.

The complete ankle system was first simplified anatomically. The complexities of the nerves, muscle etc. were evaluated at this stage. The primary goal of the simplification was to have minimal effect on the kinematics of the system, while still simplifying it as much as possible.

The next goal is to make a kinematic model from the anatomic one and simplify it as much as we can without losing the qualities necessary for the overall goal of the model's predictions.

Then the kinematic model is transformed or rather physically represented through a approximately equivalent system using analyzable components. These analyzable components could've been any standard component that has real-world properties like mass, dampers, springs etc.

From analyzing the physical system, we then find the equation(s) that describes the system. This is done with the fundamental laws of physics, like Newton's laws or Kirchhoff's laws.

When the system's equation(s) are found, the last step is to translate the equation(s) into a form that can be simulated. To do this the equation(s) are algebraically put into state-space form with,  $\dot{x} = Ax + Bu$  and,  $y = Cx + Dy$ .

### **Implementing the simulation with Matlab**

Implementing the simulation with Matlab is simply inserting the model and using the built-in `lsim()` function with the system, time range of the simulation and the inputs at all times in the simulation. Of course there is no need to use the built-in function, but the quality and flexibility of it makes the added benefit of a custom-function negligible.

### **3.1.5 Visualization**

The goal of the visualization is to show the results from the simulation in a meaningful way. To do this, the simulation data need to be translated and then put into figures that updates to become a video-like representation.

#### **Translating the results into figures**

The translation from the simulation results into a visualization figure is completely dependent on the simulation output. The output needs to be understood and then logically reconstructed in the figures. Using this project as an example, the logical visualization of the simulation would be to visualize a ankle movement, which then means to draw a ankle, and change it's position or angle to accurately represent the simulation results. Therefore the simulation was first made,

then the visualization was designed afterwards to satisfy any need for data translation.

### **Creating the moving image of the simulated ankle**

To create a moving image (video) of the simulated ankle the figures need to be simulated and visualized in many "frames" that can be cycled through to create the illusion of movement. There are many programming languages available that can do this, and all that is necessary is to either save and replay the visualization frames, or to live-play them while simulating. After deciding on a programming language and whether to replay or do a live visualization the geometric logic in the visualization shapes needs to be implemented.

### **Implementing the visualization with Matlab**

Implementing this in Matlab requires the use of a figure and clearing it before generating the next figure and either storing them to replay after the complete simulation and visualization data is generated, or updating them in real-time with the simulation. Dependent on the function used for creating the figures, and the quality of the figures that are generated, the speed of the updates may vary greatly. With very high quality comes more data and more calculations for the generation onto the figure. The choice between real-time and replays are therefore completely dependent on the specific situation.

## **3.2 Materials**

This chapter gives insight and explanation of the materials in the project. It will also elaborate on the choices made and the components that were selected. Since the project is a proof of concept, all of the components are chosen based on a combination of availability and price, while still meeting the requirements specified in the previous section.

### **3.2.1 Data collection**

Since the provider for this project is Ålesund Biomechanical Laboratory, the group had access to resources that would otherwise be difficult to obtain. At the laboratory the group had an orthopedist to help with the challenges regarding the medical field. The lab also had a selection of literature and scientific papers covering the relevant subjects. The engineers at the laboratory also shared their experience and knowledge.

### **3.2.2 Construction**

When choosing material for the main construction there were some options. The two main alternatives were stainless steel or aluminum. Stainless steel would need to be cut and welded. This, in comparison to aluminum, which can be bought as finished profiles, and therefore do not require any fabrication.

The group decided that the best material for constructing the rig would be to make it out of aluminium profiles. These types of profiles are strong, lightweight and very modular. Aluminium profiles are also available in a huge variety of dimensions and lengths. The aforementioned points are all highly desired attributes for this kind of rig, and the reason why aluminium profiles would be ideal for this application.





Figure 3.1: Example of aluminium profiles.

Source: [www.industry-plaza.com](http://www.industry-plaza.com)

### 3.2.3 Ankle model

When choosing what type of ankle model to be used for testing, the group consulted a orthopedic surgeon at the laboratory. Together it was decided that the "Fot med ankelledd, fleksibel, model 6056" (Foot with ankle joint, flexible) would be ideal. This is a model normally used for medical education. In this model the skeletal parts are made of plastic, and are connected with elastic strings.



(a) Front.



(b) Side.

Figure 3.2: Ankle model.

Source: [www.gymo.no](http://www.gymo.no)

### 3.2.4 Linear motion

For creating the linear motion to pull on the tendons the group considered using ball screw driven actuators, belt driven actuators or rack and pinion driven actuators.

Belt drive actuators are best suited for moving lightweight objects over long distances at various speeds. A drawback to belt drive is the stretching effect of the belt making for imprecise movements under heavy loads. The next alternative was rack and pinion actuators. These can move heavy weighted objects over long distances with high accuracy. The main drawbacks are the excessive need for lubrication and the complicated mounting method, which would make it challenging to fit in the space available. The last alternative are ball screw actuators. By using ball screw actuators it is possible to move heavy loads with high accuracy. Some other advantages is the low need for maintenance and the compact dimensions. The main disadvantage of ball screw actuators are the high price and relative low operating speeds.

By considering the limitation set by the available space, and the need for precise movements under potentially heavy loads, the group's choice for linear motion were ball screw actuators. When selecting the specific actuators for the task, there were many factors that needed to be addressed. Firstly the actuators needed to be able to move loads in the area of 3500N, and preferably at a reasonably fast speed. Secondly the project needed four actuators, one for each of the tendons to be controlled. This was of course a big factor when considering cost, because any potential price difference between different actuators would effectively be increased fourfold.

When considering our available suppliers and available budget the choice finally fell on the "RS Pro Electric Linear Actuator ID10, 24V DC" from RS Components. This is a ball screw actuator with a travel range of 305mm, a force capacity of 2500N and a speed of 47.2 mm/s at full load. There were several other actuators available within the price range, but no other actuator had the satisfactory combinations of the desired specifications.



Figure 3.3: RS Pro Electric Linear Actuator ID10, 24V DC.

Source: [www.rs-online.com](http://www.rs-online.com)

### 3.2.5 Linear actuator controller

For controlling the actuator the group chose to use the "RoboClaw Motor Controller". These were chosen because of their high performance and their excellent selection of compatible libraries. Since the linear actuator of choice has a power draw of 13.2A at 24VDC, the 2x15A model of the RoboClaw was selected. As an added bonus each motor controller can run two actuators, so only two RoboClaws were needed for the project.

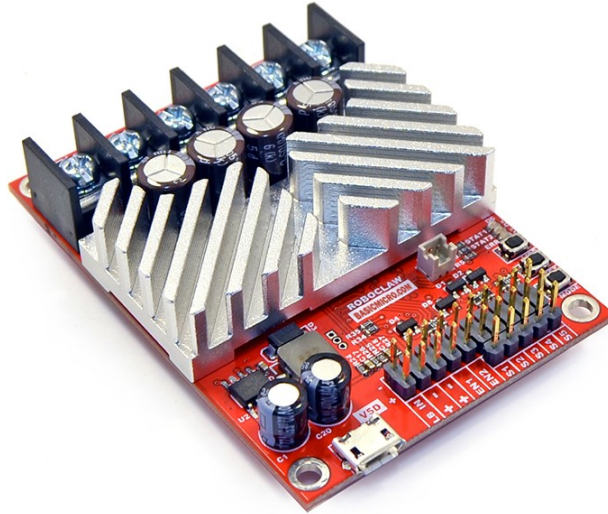


Figure 3.4: RoboClaw 2x15A.

Source: [www.basicmicro.com](http://www.basicmicro.com)

### 3.2.6 Load cell

When selecting the type of load cell to use in the system the group considered compression load cells, beam load cells and tension load cells.

Since the task of the load cell is to measure the pulling force on the tendons, the compression load cell is unusable, as it only measures compression forces. The beam load cell can be used for measuring tension and is by far the most affordable alternative. However, using a beam type load cell to measure tension requires a very specific way of mounting, and that can be a big issue in this type of application. Tension load cells are made for measuring tension, and is often formed in an S-shape, making it ideal for in-line load measuring. The drawbacks of the tension load cell is the somewhat higher price compared to beam load cells. The group therefore decided to go with a tension load cell.

When considering our available suppliers and available budget the choice finally fell on the "RS PRO Alloy Steel S Beam Load Cell" from RS Components. This is a tension load cell with a load capacity of 300Kg and a C3 accuracy class. There were a few other load cells available within the price range, but no other load cell had the desired accuracy and a 300kg weight capacity.



Figure 3.5: RS PRO Alloy Steel S Beam Load Cell.

Source: [www.rs-online.com](http://www.rs-online.com)

### 3.2.7 Load cell amplifier

For a load cell amplifier the group chose to go for the Seed HX711 24-bit. The HX711 is an affordable yet accurate 24-bit analog to digital converter made for strain gauge based load cells. It has a plethora of libraries and examples, and works with just about any load cell and controllers.

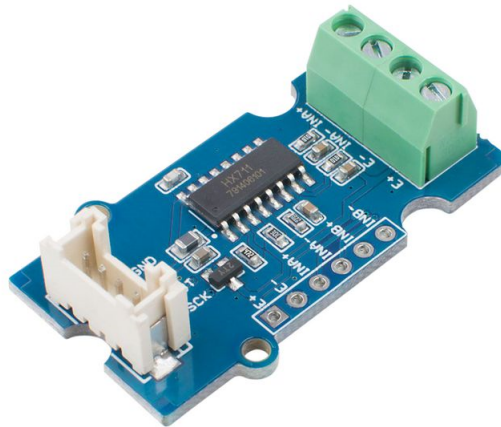


Figure 3.6: Seed hx711 load cell amplifier.

Source: [www.elfadistrelec.no](http://www.elfadistrelec.no)

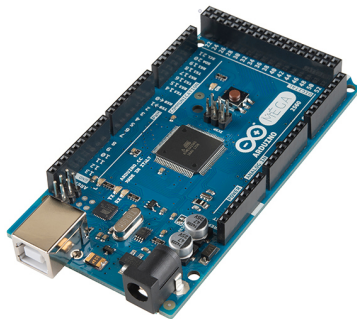
### 3.2.8 System controller

For handling input and output interaction it was necessary to run a microcontroller . This interaction was needed to control both the load cells and the actuators. All group members were familiar with the Arduino series of microcontrollers, this was therefore the obvious choice.

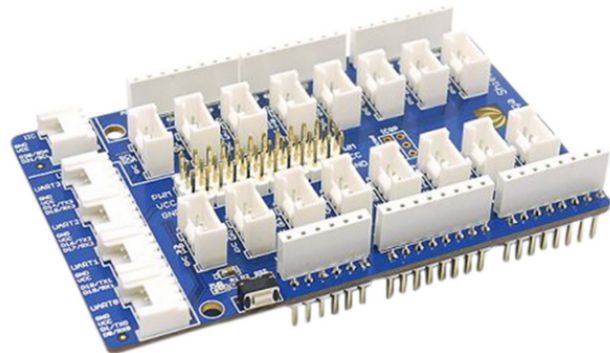
Arduino			
Model	Digital I/O	Analog I/O	Processor
Arduino Mega	DI/O 54	AI 16 / AO 15	ATmega2560
Arduino Uno	DI/O 14	AI 6 / AO 6	ATmega328P
Arduino Nano	DI/O 14	AI 8 / AO 4	ATmega328V
Arduino Micro	DI/O 20	AO7 / AI14	ATmega32U4

Table 3.1: Arduino models

The selected Arduino must have enough I/O for controlling the complete system. To ensure compatibility with potential future expansion of the system, the group decided to go with the Arduino Mega. To make the connection to the load cells as hassle free as possible, a shield made by Seeed studios was also selected.



(a) Arduino Mega.



(b) Seeed shield.

Figure 3.7: Arduino Mega and Seeed Shield.

Source: [www.crazyipi.com](http://www.crazyipi.com) & [www.distrelec.biz](http://www.distrelec.biz)

### 3.2.9 Communication protocol

The group chose to use serial using TTL for the communication between the controllers. This was chosen because its supported by all our controllers. With serial there is no need for a bus, and a point to point connection is easy to maintain and handle. Both the Arduino and the RoboClaw have libraries for serial communication, making the process easier. For communicating with the Arduino and for the initial configuration of the RoboClaws the group uses a USB connection and the manufacturers software.

### 3.2.10 Power supply

For powering the system, a power supply was needed. When choosing the power supply, calculations were made based on the specifications for already selected the components.

Firstly the the linear actuator is 24V DC, meaning the a 24V power supply is needed. Each actuator is also rated at 13.4A at full load and the controllers are rated at 15A per channel. Some calculations then show that the minimum rating for the power supply would be 1440 Watts.

$$(15A * 4) * 24V = 1440W$$

The closest power rating is 1500W. After considering availability and price the group chose to go for the "Cosel 1.5kW Embedded Switch Mode Power Supply, 24V DC". This is a compact and with a high efficiency of 88%, and a low idle powerdraw of only 1.5 Watts at idle.





Figure 3.8: Cosel 1500W power supply.

Source: [www.rs-online.com](http://www.rs-online.com)

To protect the power supply from the regenerative power that occurs when braking the actuator, the Basicmicro VClamp board was chosen. From reading the VClamp documentation it was recommended using a 50W 2Ω resistor.



(a) Basicmicro VClamp.



(b) 50W Resistor.

Figure 3.9: VClamp and resistor

Source: [www.robotshop.com](http://www.robotshop.com) & [www.elfadistelec.no](http://www.elfadistelec.no)

A fuse for the 230V input on the power supply was needed to protect against any shorts or potential overloads. Some calculations then show that the required rating for the fuse.

$$1500W/230V = 6.5A$$



A 6A circuit breaker was selected.

### 3.2.11 Cable and pulley

For connecting the actuators to the ankle model, some sort of cable was needed. The group decided that for the purpose of testing it would be beneficial to use a cable with a specified break strength. This would help ensure that in case of failure, the cable would be the weak link, and not damage the rest of the system. It was decided that the natural choice would be to use braided fishing line. This type of line is soft and easy to work with, yet is strong and has virtually no stretch. The line selected had a diameter of 0.40mm and a break strength of 26.4KG.

To be able to run the cable from the actuator to the ankle, some pulleys were needed. As the cable would change directions two times between the connected points, two pulleys were needed per actuator, for a total of eight pulleys. The group chose the Petzl P05W Aluminium Pulley because of its affordable price and high strength of 5KN.



Figure 3.10: Petzl Aluminium Pulley .

Source: [www.rs-online.com](http://www.rs-online.com)

### 3.2.12 End switches

Since the selected linear actuator did not have built-in end switches, external switches were needed. It was initially desirable to use optical sensors since they do not have any wear. On the other hand the need for extra wiring made them less attractive for use with a moving actuator. The other alternative were mechanical end stop switches. These are easy to install, only requiring two wires. This and the compact size made these the easy choice for detecting the outermost retracted position of the actuator.

### 3.2.13 Software and libraries

The following software and libraries has been used throughout this project

#### Software

- **Arduino IDE** - Free environment for programming and interacting with the arduino micro controllers. Released and made by the arduino corporation. All arduinos has been programmed using this IDE
- **Fusion 360** - 3D designing software from Autodesk.
- **Overleaf** - A free web-based tool for writing in LaTeX. Some of included features are spell checking and cloud-based storing. The cloud-based storage makes real-time editing possible and therefore it is possible to cooperate in the same files.
- **MATLAB** - A proprietary programming language and numeric computing environment developed by MathWorks.
- **Basic Micro Motion Studio** - Software used to configure and monitor Roboclaw-units.
- **Onenote** - Digital note-taking app made by Microsoft.
- **www.circuit-diagram.org** - Online based software for making circuit diagrams.

## **Libraries**

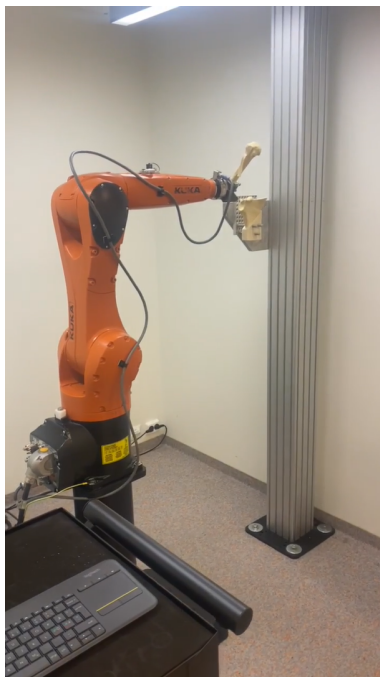
- **HX711** - Arduino library for HX711 24 bit ADC used for load cells and scales.
- **RoboClaw** - Arduino library provided by BasicMicro for controlling RoboClaw.
- **SoftwareSerial** - Arduino library to allow serial communication on digital pins of the Arduino.

# Chapter 4

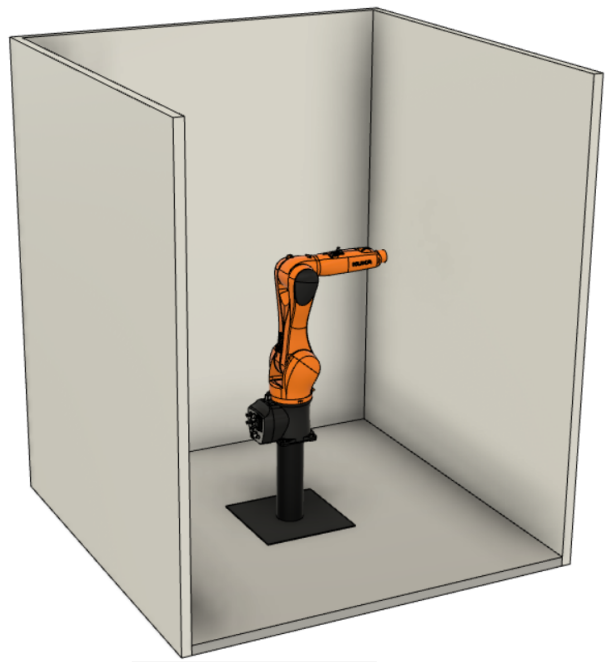
## Result

### 4.1 Design

This section explains how the rig is designed. All 3D-models are made using Autodesk Fusion360.



(a) Image.



(b) CAD model.

Figure 4.1: Current lab.

By analyzing the environment where the final product will operate, the group made a rig

design to fit within these parameters. In the room where the rig would be localized, it is 110 centimeters of clearance between the current testing rig and the wall. The group therefore made a CAD model of the room, to make further design choices an easier task.

#### 4.1.1 Main frame

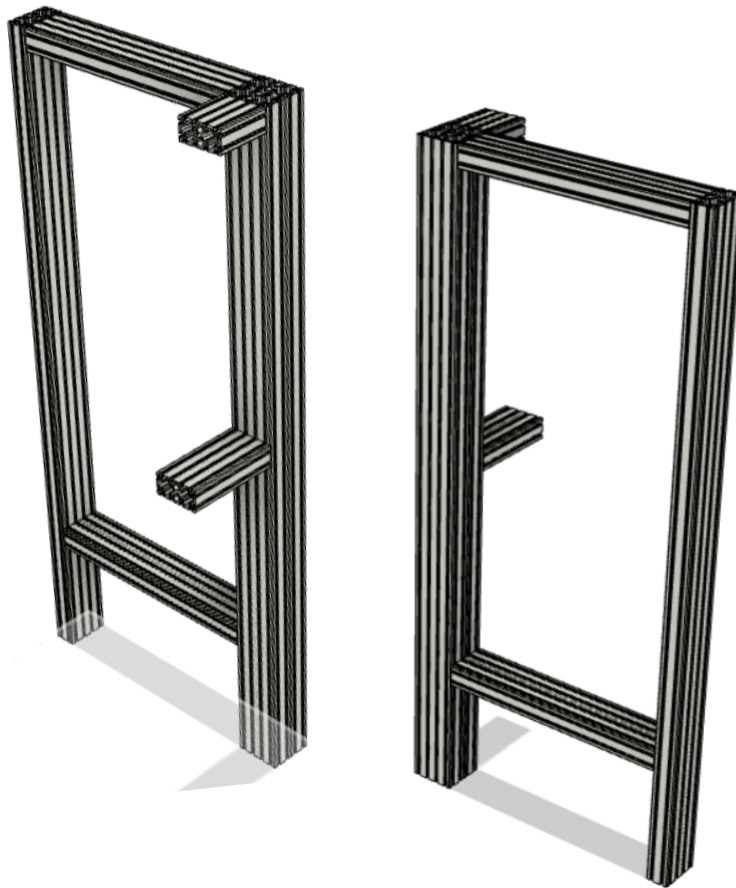


Figure 4.2: Main frame CAD model.

The main frame is the bearing construction and the shell of the rig. The material chosen for the main frame is aluminum profiles, as decided in chapter 3.2.3. The frame is constructed by 80x160 millimeter profiles using aluminum L-connectors for the purpose of connecting the profiles at a 90 degree angle. When designing the main frame, it is desirable to use as much vertical space as possible, so that it minimizes floor area used. It is also desirable that all moving parts are kept within the circumference of the frame. As a result of these conditions the main

frame is 1070mm wide, 170mm deep and 2400mm high.

### 4.1.2 Linear actuators

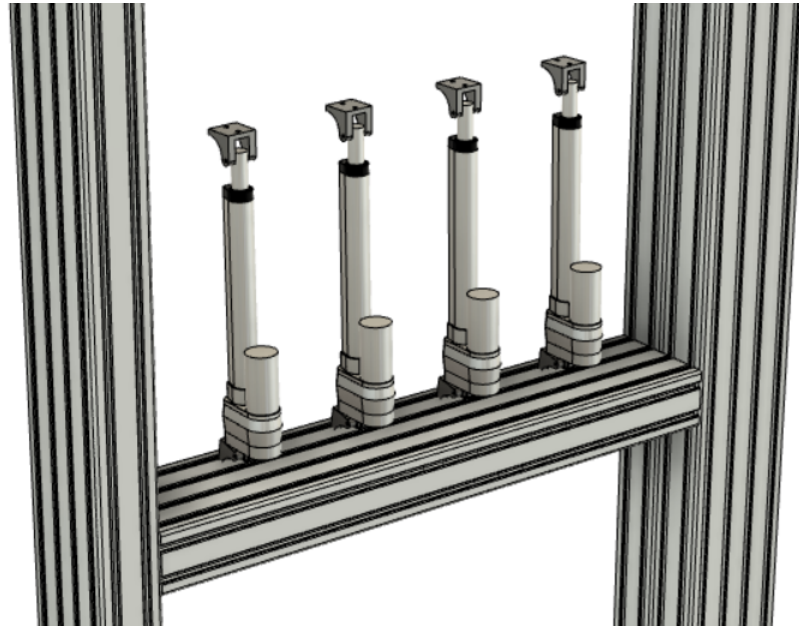


Figure 4.3: Actuators CAD model.

The actuators are mounted vertically to make the best use of the available space. In this configuration the actuators have a clearance of 1385mm from the top of the actuator to the top of the frame when retracted, and a clearance of 1080mm when fully extended. This large clearance allows for greater flexibility when configuring the load cells and pulleys.

This configuration also ensures that the actuators are mounted well within the perimeter of the frame, and do not extend out in the room. They are mounted to the rig using two 90 degree angle brackets per actuator. With each of the actuators being 77mm wide and the clearance between them being 102mm, there is adequate room for at least 5 more actuators in case of future expansion.

### 4.1.3 Load cells



Figure 4.4: Load cells CAD model.

The load cells are mounted directly on top of the actuators using a 90 degree bracket and 12mm bolts. This makes for a solid and stable mount, ensuring good stability and repeatability when measuring loads.

#### 4.1.4 Cable and pulley

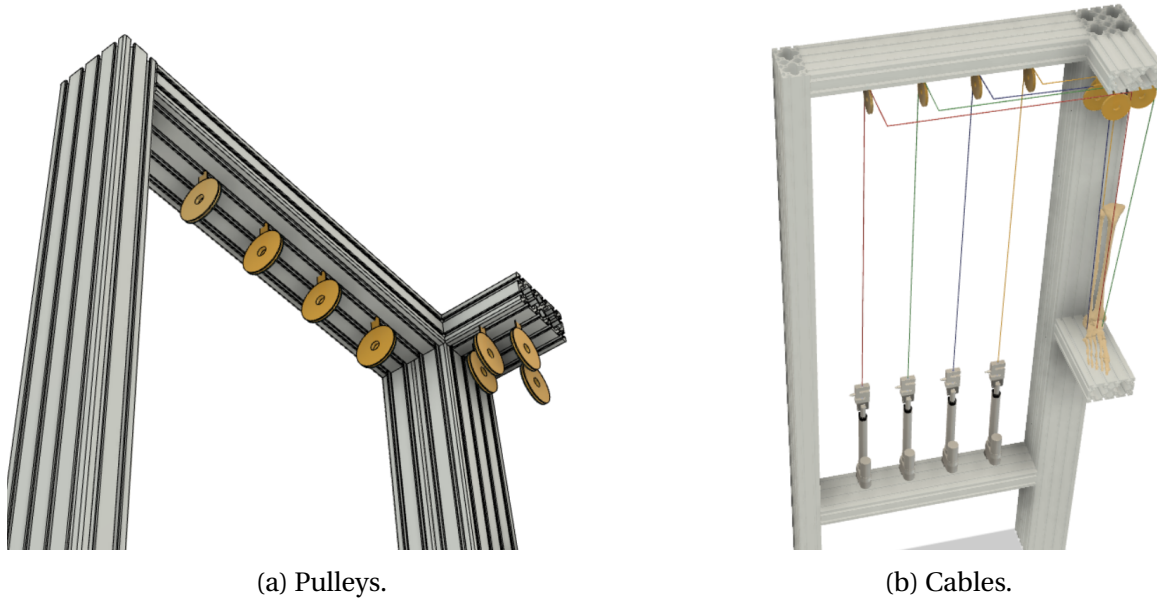
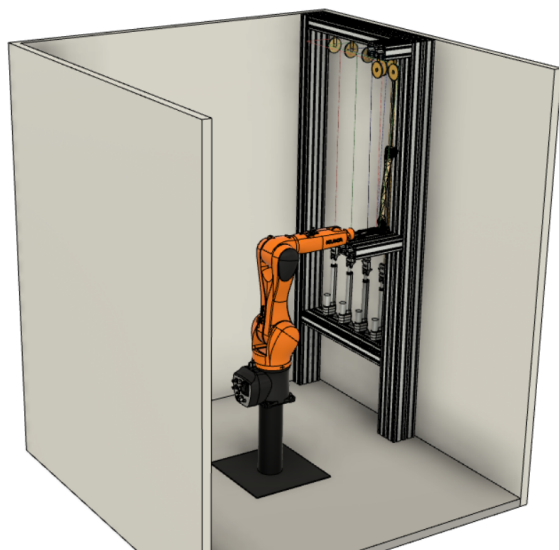


Figure 4.5: Cable and pulley CAD.

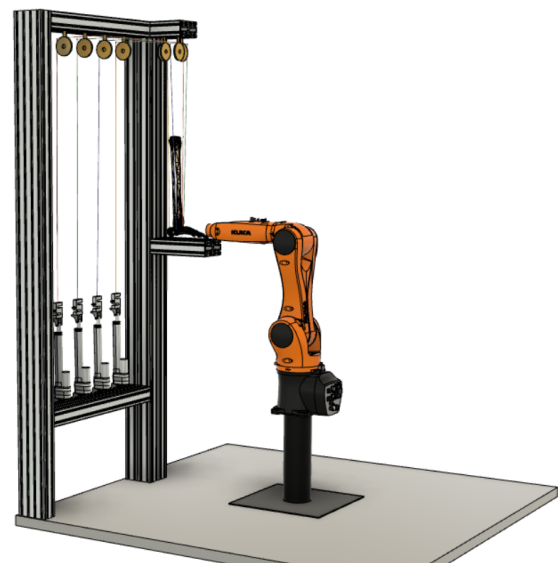
To ensure a linear and accurate connection between the linear actuators and the ankle, the pulleys are positioned directly over the connections on each part. This helps prevent any of the cables from pulling at a skewed angle. The distancing ensures that no cables or pulleys touch or overlap.



### 4.1.5 Complete Design



(a) Room.



(b) Rig.

Figure 4.6: Complete design CAD.

Figure 4.6 shows the completed rig design placed in the current laboratory.

### 4.1.6 Electrical layout

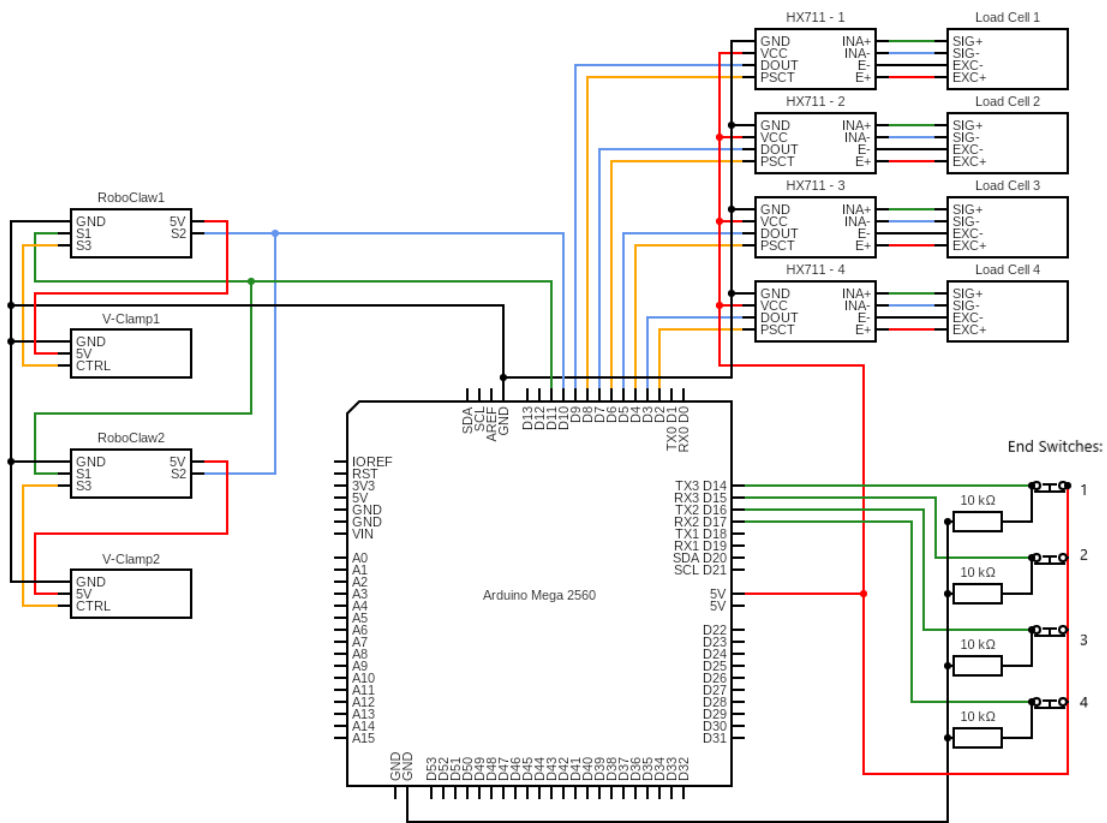


Figure 4.7: Wiring diagram control circuit.

The control circuit schematic as part of the electrical layout is as shown in figure 4.7.

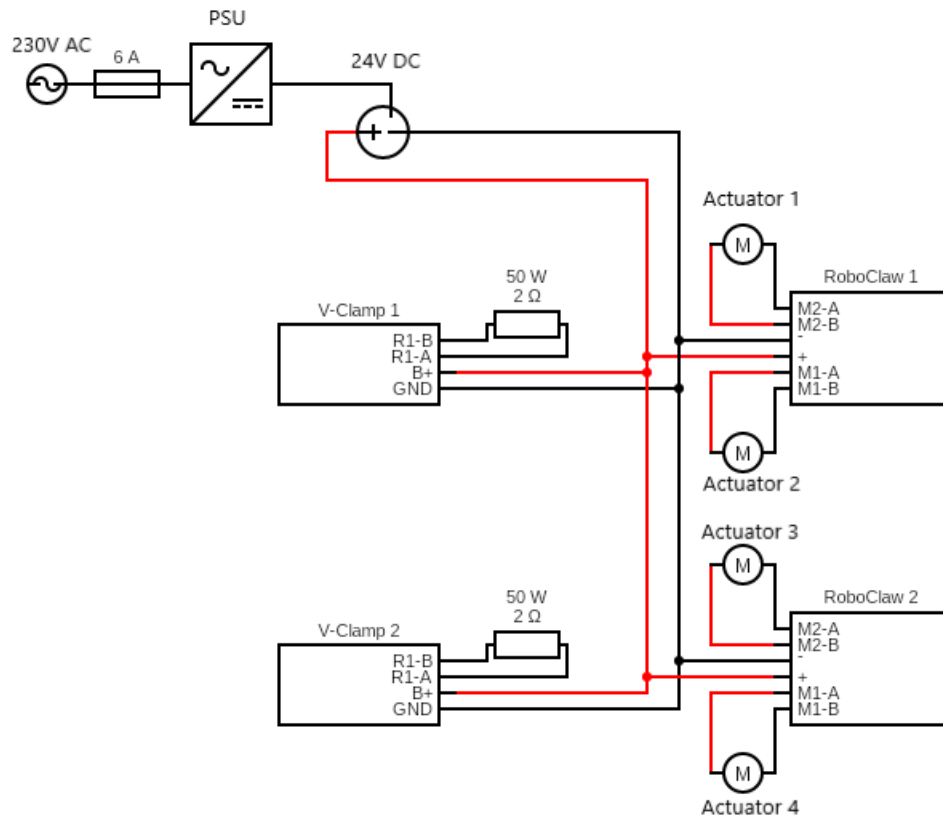


Figure 4.8: Wiring diagram main circuit.

The control circuit schematic as part of the electrical layout is as shown in figure 4.8.

## 4.2 Build

This section explains how the rig was built.

Because of limited access to the lab at Ålesund Biomechanical Laboratory, the group decided together with the supervisors to build the frame out of wood as a proof of concept, instead of the aluminium design. The build was therefore performed by the group members in the FabLab at NTNU Ålesund.

### 4.2.1 Main frame demo

The main frame demonstration was constructed using 48x48mm timber with a 24x144mm plank as a base. The corners were strengthened using 90 degree steel brackets. This frame was made to replicate the aluminium frame design, and was meant for demonstration and testing purposes. The dimensions are therefore true to the original CAD design, only without the lower feet.

### 4.2.2 Linear actuators

The linear actuator were mounted to the base of the frame using two modified 90 degree steel brackets per actuator. The brackets have a dimension of 60x60mm and originally had a 10mm mounting hole. The 10mm hole was drilled out to 12mm to fit the mounting bolt for the actuator. The brackets were fastened to the wood base using 20mm long wood screws, and to the actuators using a 12mm diameter bolt and a locking nut.

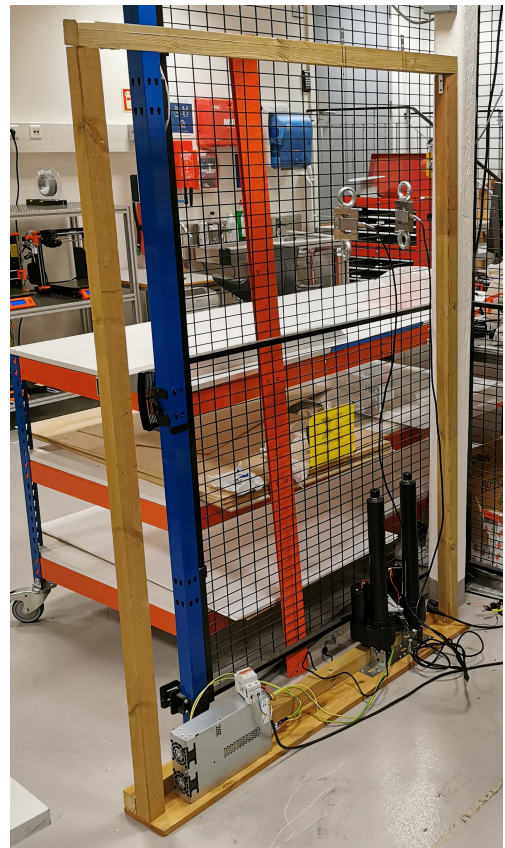
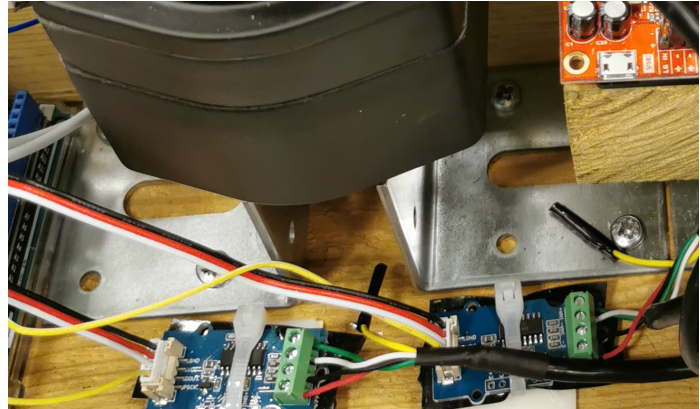


Figure 4.9: Main frame demo.



(a) Actuators.



(b) Brackets.

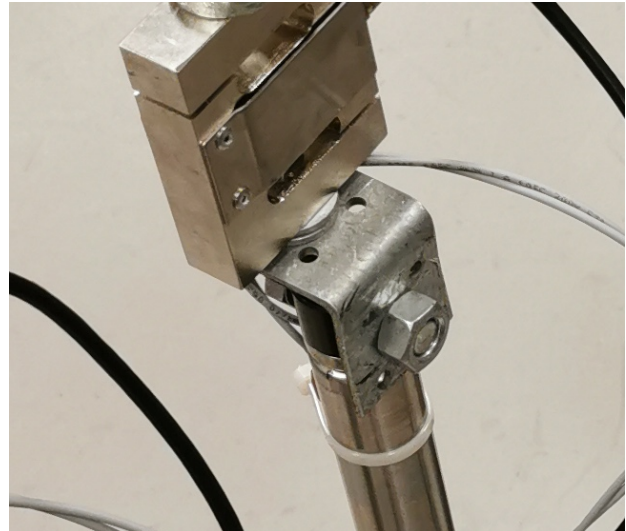
Figure 4.10: Actuators build.

### 4.2.3 Load cells

The load cells were mounted on the arm of the linear actuators using a modified 90 degree steel bracket. The brackets were originally 40x80mm but was shortened to 40x40mm. The brackets originally had 8mm mounting holes, these were drilled out to 12mm. The bracket were mounted to the actuator using a 12mm diameter bolt and a locking nut. The load cell was mounted to the bracket using a 12mm bolt. This ensures a tight and stable mount.



(a) Load cells.



(b) Brackets.

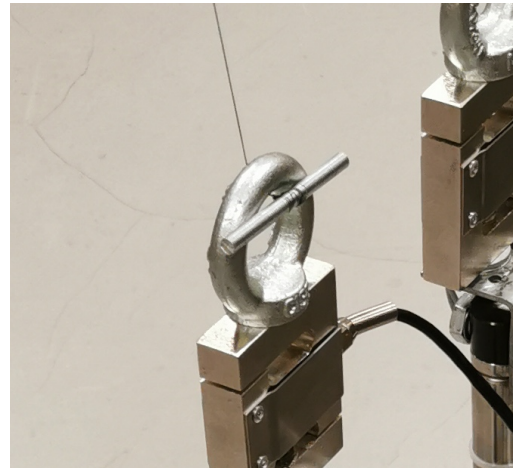
Figure 4.11: Load cells build.

#### 4.2.4 Cable and pulley

The pulleys were mounted to the top of the frame using braided fishing line. The cables running to the ankle were connected to an eye bolt on the top of the load cells using a pin, making for easy adjustment and unhooking of the cable.



(a) Pulley.



(b) Eye bolt.

Figure 4.12: Cable and pulley build.



### 4.2.5 Ankle model

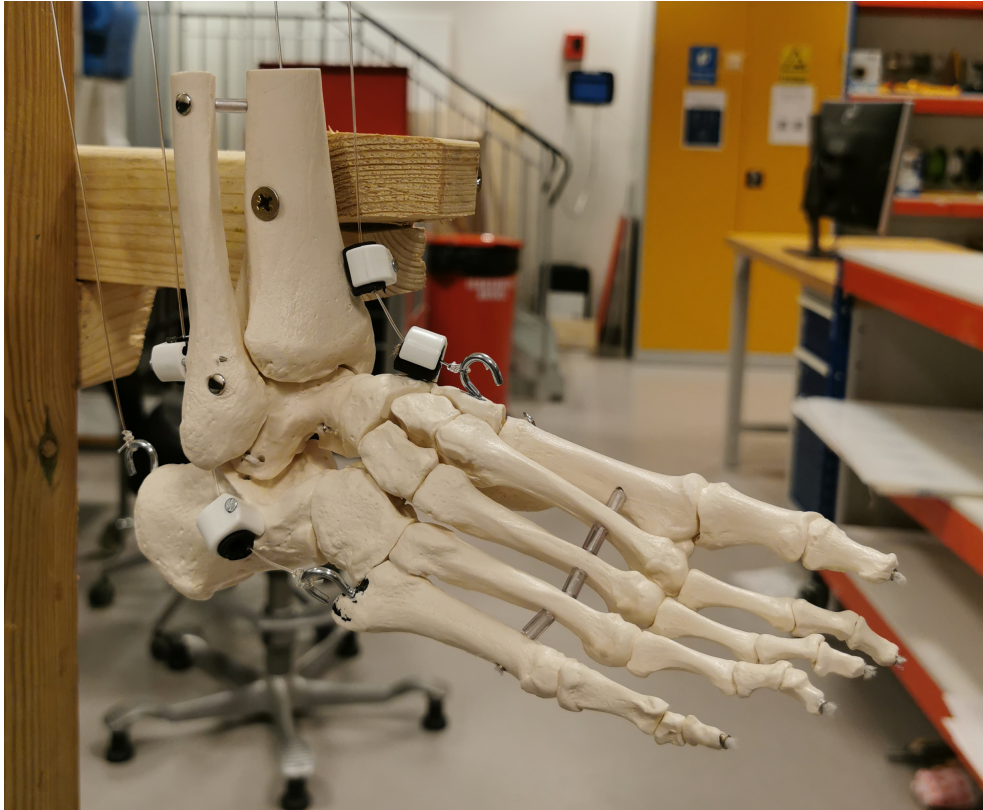
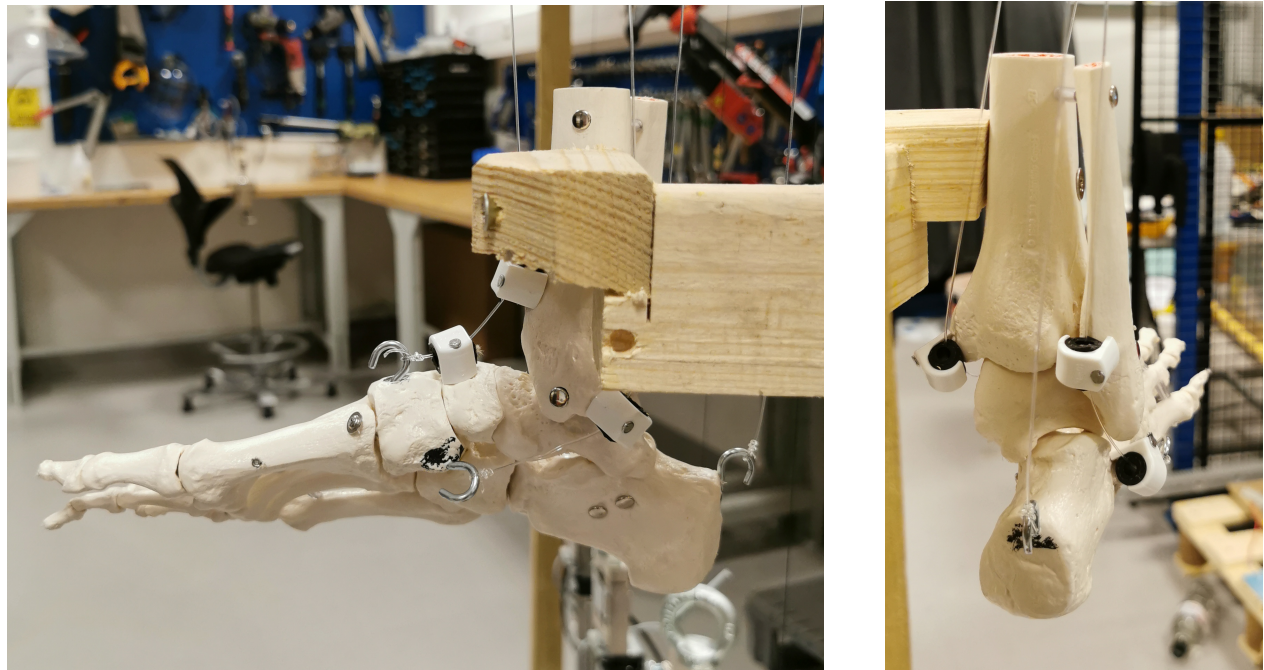


Figure 4.13: Modified ankle model.

The plastic ankle model needed modifications to be able to be used for testing purposes. Firstly holes were drilled in the location where the four tendons: Achilles, Peroneus brevis, Posterior and Anterior tibial attach to the bone. A 2.5mm screw hook was fitted in each of these holes. To simulate the retinaculum that holds the tendons in place, a 6mm inner diameter fuel hose was cut in 10mm lengths and held in place by cable clamps. To act as the tendons, clear fishing line was tied to the hooks and guided through the simulated retinaculum. The model is mounted to the frame using a single screw. This makes it possible to orient the ankle at a specified angle. As seen in figure 4.12, the tendons are clear of the frame and have full range of motion.



(a) Inner.

(b) Rear.

Figure 4.14: Modified ankle model

The finished ankle model has a negligible internal resistance and a full range of motion. The only restrictions of movement is the physical shape of the ankle model. When pulling on the simulated tendons, the end point in the range of motion makes for a notable increase in resistance, as seen in table 4.1 This is helpful for defining end points in the control software.

<b>Tendons Resistance - in grams</b>			
<b>Tendon</b>	<b>Low</b>	<b>Neutral</b>	<b>High</b>
Achilles	50	50	400
Posterior tibial	50	150	230
Anterior tibial	50	150	280
Peroneus brevis	50	400	1200

Table 4.1: Tendons resistance

#### 4.2.6 Electrical

The electrical system was connected as shown in the electrical schematics in 4.1.6. As this was a temporary build, the control circuit was mainly connected via a breadboard.



All the electrical components functioned as intended. The Serial communication between the Arduino and the RoboClaws worked without any issues. The voltage clamps kicked in as intended when the actuators had any sudden stops or retardation, therefore the power supply never had any issues. The load cells were accurate and had a fluctuation of only  $\pm 6$  grams at a 1% load, i.e. a 3 kg load.

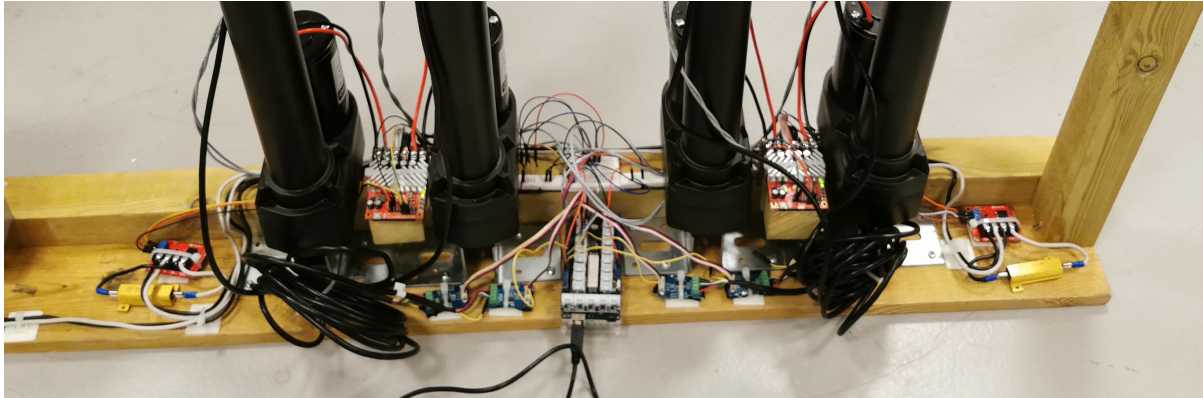
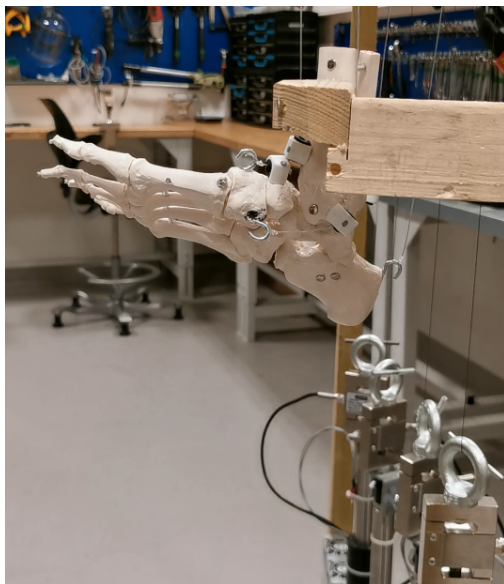
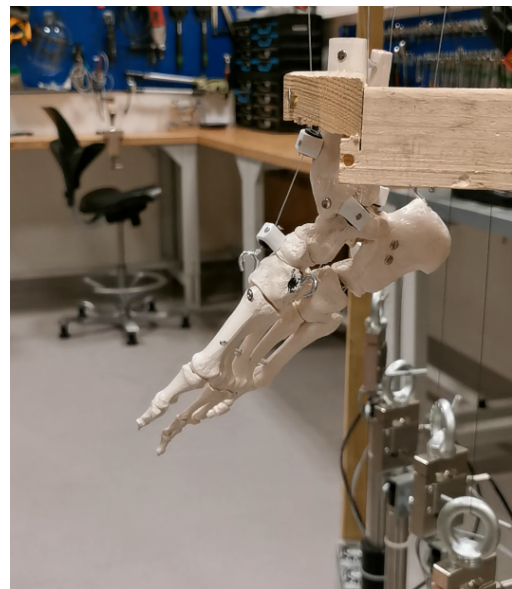


Figure 4.15: Electrical connections.

### 4.2.7 Completed build



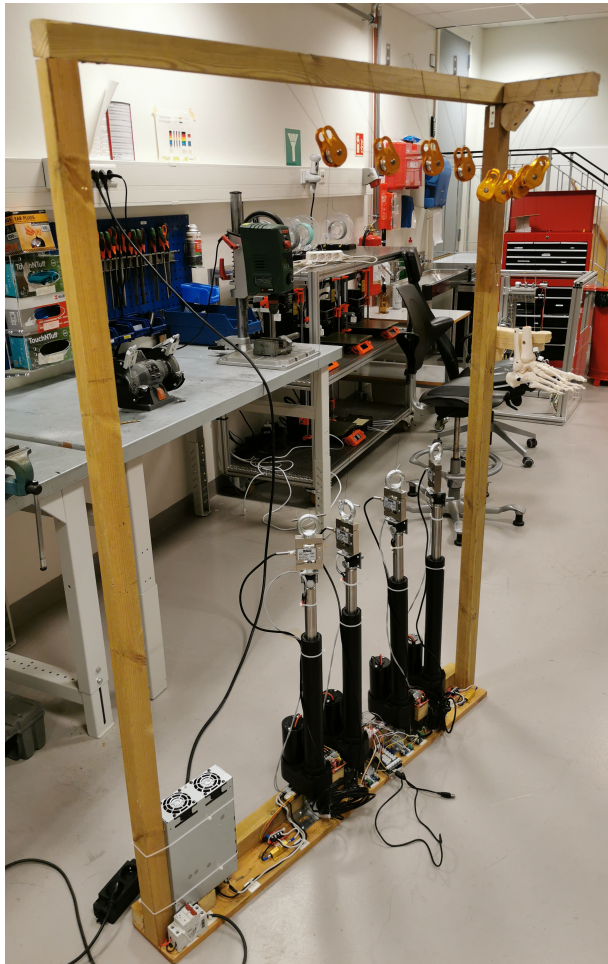
(a) Foot pointing up.



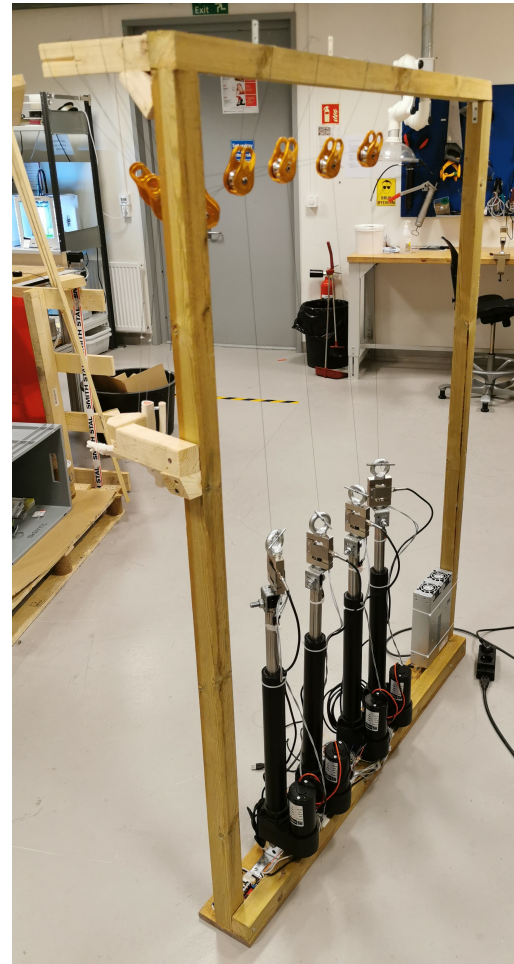
(b) Foot pointing down

Figure 4.16: Ankle model during testing

Figure 4.16 shows the ankle in the 2 end positions during the initial testing. The ankle moves steadily to each position before moving to the next position in a controlled manner. The ankle model has little to no distortion or stretch while the values are within those measured in table 4.1. However when the values exceed those values, the elastic bands in the ankle begin to stretch and the model starts to deform.



(a) Front.



(b) Rear.

Figure 4.17: Completed build

The completed proof of concept build under load as showed in figure 4.17.

## 4.3 Implementation

### 4.3.1 Arduino code

The arduino code is the code used in the prototype, it controls the actuators, reads the load-cell values and uses them for determining the direction to drive the actuators to reach the correct ankle positions. There is also implemented a text-based user interface (UI) that gives control over each actuator manually, and the option to use several automatic features like automatic walking and automatic neutral foot etc. These automatic features base their movement behaviour on a PID control loop where the output sign from the PID determines the direction of movement, and in contrast to typical PID control it does not change the speed of the actuators. This is largely due to the fact that small speeds wouldn't actually move the actuators, and so set speeds were used instead.

#### **setup()**

In the setup function as in all Arduino's the communication protocols are initiated. Then the load-cells are calibrated with calibration factors that was gathered experimentally, making calibration upon every startup automatic. The desired load-cell readings for the ankle model's different positions are loaded in the setup; these load-cell readings were also collected experimentally.

#### **loop()**

The Arduino loop consisted of a large switch statement where, if the loop is available, the person operating the rig can input different characters to change the states of the system.

There are fourteen different inputs that are implemented currently. Those can be categorized into three main categories, the automatic, manual and "other" commands. The "other" commands are the commands that control whether the system is in automatic or manual mode, and the fail-safe command that stops all motors completely disregarding any other system states or variables.

The controls that aren't mode specific are listed below:

- **x** : Stop all
- **m** : Manual mode
- **a** : Automatic mode

And inputs that are exclusively functional for manual mode:

- **1** : Select motor 1
- **2** : Select motor 2
- **3** : Select motor 3
- **4** : Select motor 4
- **s** : Stand-still
- **f** : Forward / Actuator up / Loosen
- **r** : Reverse / Actuator down / Pull

The inputs that decide actions like forward or reverse, will be applied to the selected motor only, which by default is motor one, but with the use of the "Select motor X" commands 1-4 all motor can be manually controlled.

And finally the inputs exclusively functional for automatic mode:

- **u** : Automatic upward (Drives the ankle to it's highest angle)
- **d** : Automatic downward (Drives the ankle to it's lowest angle)
- **n** : Automatic neutral (Puts the ankle in a neutral angle)
- **w** : Automatic walking

Automatic upward, downward and neutral are all simply using PID() with static desired load cell reading values that the system continuously tries to achieve. The walking mode will simply cycle through the other modes. The specifics will be described in more detail later this section.

**PID()**

The code for this function is shown in figure 4.18. The input is which motor it will read information from and the time the load cell related to that motor was gathered, then set the directional value for. First the function needs to establish a "sense" of time, to do both the integration and derivation for the integral and derivative terms respectively. This is done by comparing the previous time of reading the load cells values and the current time of reading the load cells value. Then it gets the error value for the load cell of the given motor, by comparing the reading with the desired value that relates to the current motor and the current "position". The positions are for example having the ankle in a neutral position. The error is then used in a typical PID calculation and the output is returned.

```
float PID(int motor, unsigned long currentTime){
    elapsedTime[motor] = currentTime - previousTime[motor];
    error[motor] = desiredVal[currentAutoMotion][motor] -
        Load_Cell_Value[motor];
    cumError[motor] += error[motor] * elapsedTime[motor];
    rateError[motor] = (error[motor] - lastError[motor])/elapsedTime[motor];
    previousTime[motor] = currentTime;
    lastError[motor] = error[motor];
    float pid_out = Kp * error[motor] + Ki * cumError[motor] + Kd *
        rateError[motor];
    return pid_out;
}
```

Figure 4.18: Function from Arduino code in prototype without comments and compressed, function: PID()

**roboclawMovement()**

The function roboclawMovement() was used to be able to give instant instructions to the actuators while in a iteration. The reason why this is not so simple otherwise, is because the Robo-



Claw function that gives the instructions to the actuators uses two separate function names for the two different motors in each address. The address is what identifies which RoboClaw motor controller to communicate with. The alternative would be to use several if-else statements to check which motor's PID output is currently being calculated to use the correct RoboClaw function. Instead this function uses two inputs. First, the motor is identified with the first input, and then the second input decides the actuator speed. This is also useful for improvements in the future when potentially a varying actuator speed is implemented. In that case the speed can be a variable that is simply put into this function.

The cases uses numbers starting from 0, because of the nature of arrays starting at index= 0.

```
void roboclawMovement(int motor, float velocity){
  switch(motor)
  { case 0: roboclaw.ForwardBackwardM1(address1,velocity);break;
    case 1: roboclaw.ForwardBackwardM2(address1,velocity);break;
    case 2: roboclaw.ForwardBackwardM1(address2,velocity);break;
    case 3: roboclaw.ForwardBackwardM2(address2,velocity);break;}}
```

Figure 4.19: Function from Arduino code in prototype without comments and compressed, function: roboclawMovement()

### **actuatorControl()**

In the function "actuatorControl()" the functionality of the other functions and the current states determined by the operator through the serial in the loop() is brought together. Firstly the function is divided into two parts, the manual and the automatic modes.

The manual part of actuatorControl() iterates four times, where the same sequence is repeated, but with a different motor each iteration. First it checks which state the current motor is in, which would be stand-still, forward and reverse. For stand-still and forward the motor is simply set to the speed that represents that state. In reverse, it first checks whether the current motor's end switch is pressed. That sets the speed to stand-still.

The automatic part of `actuatorControl()` does several separate tasks. It checks current status with load cell readings. It uses those readings with PID control to decide which way to move the actuators, if at all. Then checks if the current position of the ankle is sufficiently close to the desired position to be considered a correct position. If this is the case then in walking mode it will switch where it's desired position is to the next one in the motion of walking. If in any of the other automatic modes, the ankle will try to remain stable in that position. If the position is not correct then it will keep trying to achieve the correct values, within a  $\pm 30$  gram load cell reading.

The order of all these tasks are in the following order. First it resets the array that through iterations holds the correct values information. This is so that a correct value from previous runs won't interfere in future ones. Then it reads the values of all load cells. The reason why this is done every iteration, is since then the time between the reading of a load cell and the time of the iteration it is used is reduced by at most 4 times. This is then used in the PID to calculate the direction to move to get closer to the desired position. For the upwards and downwards positions of the angle of the ankle, the primary pull is in the front and back tendons, therefore these primary tendons are the only ones being checked for being correct in order to confirm correct position. Meanwhile the sides will also be pulled to try and achieve stability at their desired positions, but these will not be necessary to confirm the position of the ankle. Then the exact same method to apply the desired speeds on the motors are used as in the manual mode. After the speeds are applied, the automatic mode will check one last state, which is whether the automatic mode is set to "Walking" or any of the other automatic motions. If it is specifically set to "Walking", then it will cycle through the other motions by doing either upwards, or downwards position then neutral, and then the one not directly previous to the current neutral.

## 4.4 Simulation

### 4.4.1 Model

To simplify the ankle, first the tension of flesh, nerves and muscle were disregarded. The skeletal structure was also simplified, from the initial three boned ankle with a Tibia, Fibula and Talus into a typical two boned joint as shown in figure 4.20, connecting the leg and the foot.

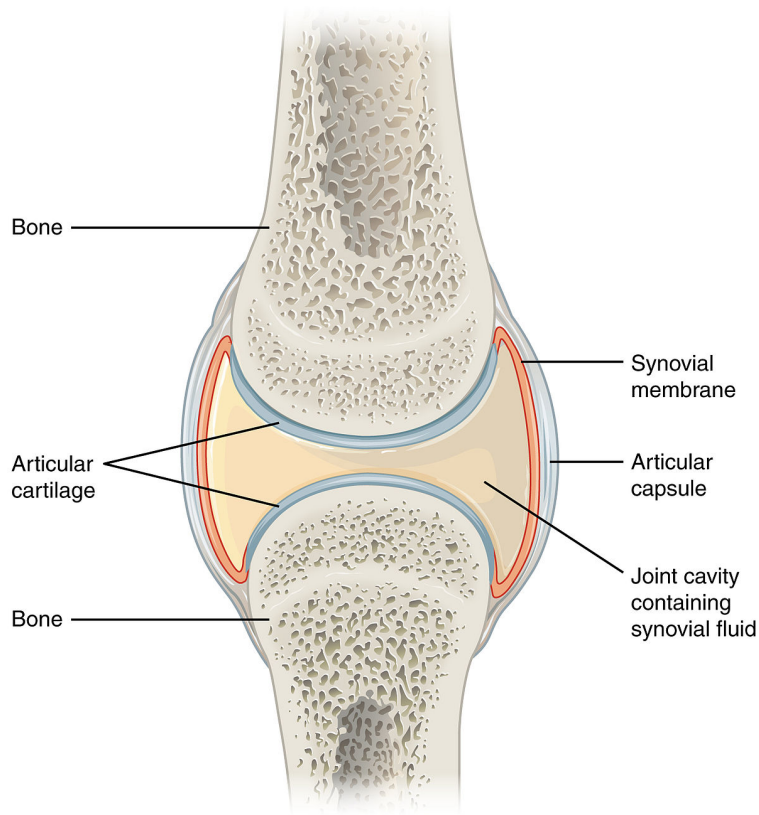


Figure 4.20: A typical two boned joint

Source: [https://en.wikipedia.org/wiki/Synovial\\_joint](https://en.wikipedia.org/wiki/Synovial_joint)

The new complexity introduced by replacing the ankle joint was reduced to only being a point of rotation. This would make the joint a hinge joint similar to a elbow joint.





Figure 4.21: A lateral view of a elbow

Source: <https://radiopaedia.org/cases/elbow-effusion-the-fat-pads>

From the simplified ankle with only two bones, the kinematic modelling starts with defining the ankle's degree of freedom. This ankle has two degrees of freedom as shown in the figure 4.22.

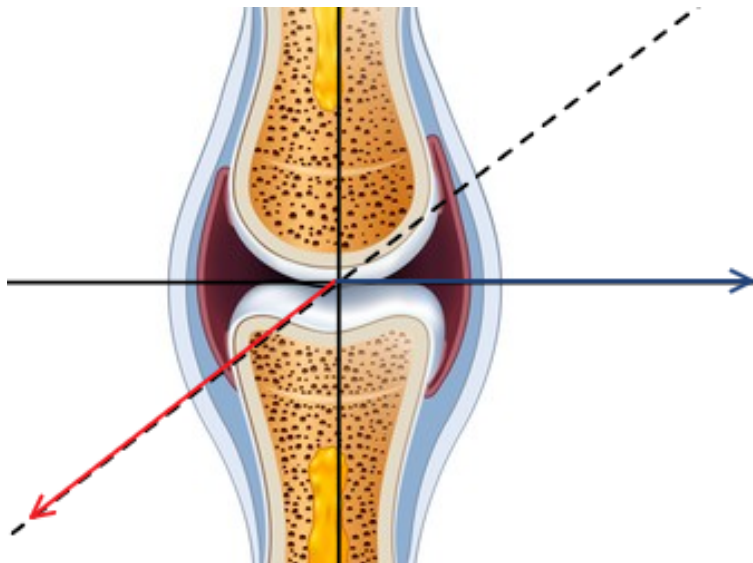


Figure 4.22: Kinematic model of ankle with two degrees of freedom

Source: <https://www.shutterstock.com/nb/image-vector/normal-synovial-joint-anatomy-healthy-detailed-217315702>

To simplify the kinematic model further, the rotation to the left or right was locked, and only forward and backwards rotation remains. This means that the new kinematic model has only one degree of freedom, very similar to an elbow joint. Therefore an elbow is used to show the kinematics of the ankle when it is limited to one degree of freedom in figure 4.23.

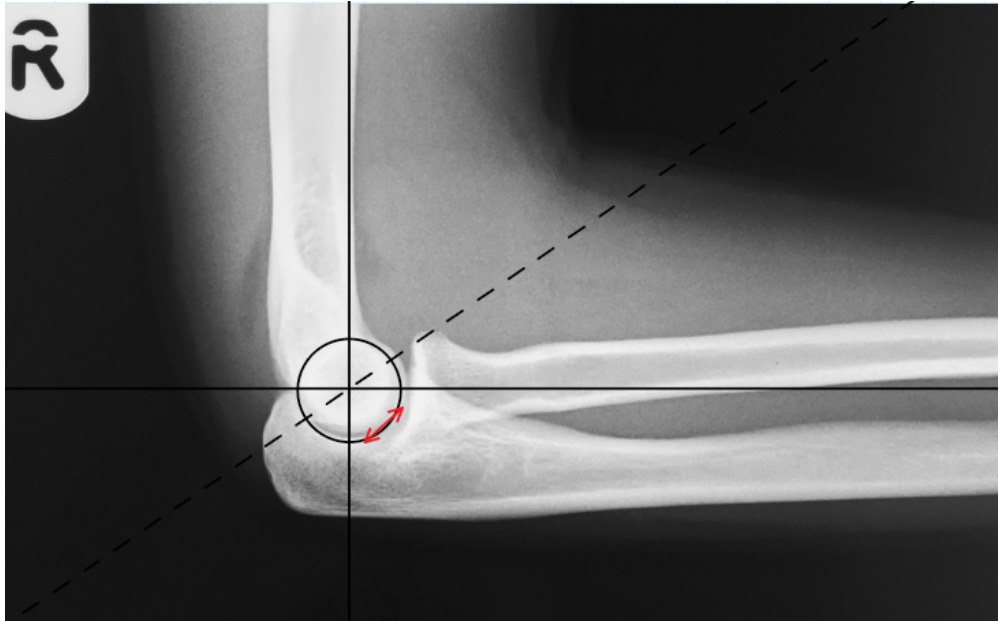


Figure 4.23: Kinematic model of ankle limited to one degree of freedom

Source: <https://radiopaedia.org/cases/elbow-effusion-the-fat-pads>

In this case the transformation from a kinematic model to an analyzable physical one was resolved through re-representing the properties of the system with a mass-spring-damper system.

Since the system to be transformed had only one degree of freedom, the mass-spring-damper system's degree of freedom also had to be one. The mass-spring-damper system configuration used is shown in figure 4.24. The effect of having the spring and damper connected to the mass in parallel is that both are acting specifically in response to the mass's change in position.

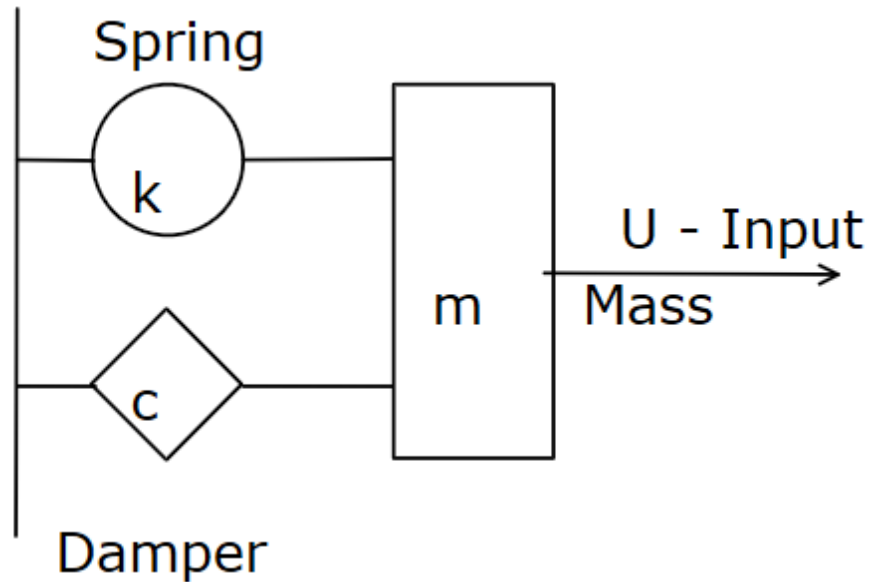


Figure 4.24: Mass-spring-damper representation of the simplified ankle

The connection between this model and the initial ankle is that the angle of the ankle is the position of the mass, and the spring force represents tension, and the damping represents all the friction inside the ankle complex. The mass simply represents the foot's mass. This model therefore includes the tension, friction, and mass of the ankle, when the skeletal structure of it is assumed to be replaced by a hinge joint similar to an elbow joint.

From this the equation(s) to describe the system can be found analytically. The following equations uses:

- $x$  = Position of the mass in regard to  $x_0$  which is the position defined to have no spring force
- $k$  = Spring coefficient
- $c$  = Damping coefficient
- $m$  = Mass
- $u$  = Input force / Pulling force

Finding the sum of forces acting upon the system and :

$$\sum F = u - \text{Damping force} - \text{Spring force} \quad (4.1)$$

$$\sum F = m\ddot{x} \quad (4.2)$$

$$\text{Damping force} = c\dot{x} \quad (4.3)$$

$$\text{Spring force} = kx \quad (4.4)$$

$$m\ddot{x} + c\dot{x} + kx = u \quad (4.5)$$

This equation is of second-order and must be split into first-order equations to be used in state-space, therefore some new definitions are made:

- $x_1 = x$ , which is the position
- $x_2 = \dot{x} = \dot{x}_1$ , which is the rate of change of position

The first-order equations then become:

From definition:

$$\dot{x}_1 = x_2 \quad (4.6)$$

Inserting the new definitions into the sum of forces equations gives:

$$m\dot{x}_2 + cx_2 + kx_1 = u \quad (4.7)$$

$$\dot{x}_2 = \frac{u}{m} - \frac{cx_2}{m} - \frac{kx_1}{m} \quad (4.8)$$

Which is the final system equations.

Finally the equations need to be translated into a state-space representation. The output of the system is the position of the system, therefore:

$$y = x_1 \quad (4.9)$$

And with this and the found first order equations the state-space representation ends up looking

as shown in equations 4.10 and 4.11.

$$\dot{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \quad (4.10)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (4.11)$$

Which is the final state-space representation of the model.

#### 4.4.2 Code

The code used in the simulation consists of five actions. The mathematical model of the ankle, which now is a mathematical model of a mass-spring-damper system, is loaded into Matlab's workspace. The input over time is generated, the ankle's height which represents how far into the air the foot is lifted is generated, the simulation is executed, the results are stored and the results are transformed into an angle.

The input and ankle height over time are created by creating a linear decrease, linear increase or constant value throughout several stages. These stages translate to specific actions or as described in the code for the prototype, motions. The motion going from upwards angle and max height in the air, to then a neutral angle and min height which would be on the ground, is a stage. These are completely separately made, but are equally linear, and was manually created to fit each other. This is shown in figure 4.25.

```
u = [0:stepsize:max max*ones(1,max/stepsize) flip(0:stepsize:max)
    - flip(-max:stepsize:0) -max*ones(1,max/stepsize) -max:stepsize:0];
ankle_height = [flip(max/2:stepsize/2:max) flip(0:stepsize/2:max/2)
    - zeros(1, max/stepsize) 0:stepsize/2:max/2 max/2:stepsize/2:max
    - max*ones(1,max/stepsize)];
```

Figure 4.25: The input and ankle height generation code from the simulation, compressed and without comments

The state-space representation is loaded into Matlab's workspace by inserting the state-space matrices, A, B, C and D and using the built-in Matlab function `ss()` to generate the state-space instance. This is shown in the code in figure 4.26.

```
A = [0 1; -k/m -c/m];
B = [0 1/m]';
C = [1 0];
D = 0;
sys = ss(A,B,C,D, 'StateName', {'Position' 'Velocity'}, 'InputName', 'Force');
```

Figure 4.26: The state-space model generation in Matlab, compressed and without comments

The arguments after the matrices in the function `ss()`, are to give names to inputs and outputs, so that the clarity of the system is improved. Otherwise the names of the inputs would be  $x_1, x_2...$  etc.

The actual simulation is then done by using the built-in Matlab function `lsim()`. This function takes the model, inputs and the time the simulation is "taking place", and outputs the model's output over time. This is shown in figure 4.27.

```
Y = lsim(sys,u,t);
```

Figure 4.27: The code the simulation is run, "**Y**" stores the simulation results, "**sys**" is the state-space model, "**u**" is the input and "**t**" is the simulation time references, without comments

And finally the results are translated into a angle, since it up to this point representing a point on a 1-dimensional line corresponding to the position of the mass from the mass-spring-damper system. This is done with a function consisting of a single line of code, `posToRad()`, which simply divides the input by  $2\pi$  and returns it.

## 4.5 Visualization

The visualization is as stated in section 3.1.5 primarily created to show the results from the simulation in a meaningful way. Therefore the goal in the visualization design process is to make the end-result as intuitive and clear as possible.

### 4.5.1 Code

The primary ways to visualize anything in Matlab is through plots and figures. Images can be used as well, but only one image at a time since when a second image is put into a figure, the first one is removed. Matlab has several other solutions for visualizing data, and the one used in the visualization of the simulation in this project is the `draw()` function. This function draws lines between points put into arrays that correspond to each "bundle" of lines. These bundles are generated as shown in figure 4.28.

```
a1 = animatedline('Color', [0,1,0]);
a2 = animatedline('Color', [102/255,51/255,0]);
a3 = animatedline('Color', [1,0,0]);
a4 = animatedline('Color', [0,0,1]);
```

Figure 4.28: The line bundles being generated, each line generates a line bundle with its own color and points array, without comments

These bundles are filled with points in a order to "draw" the lines from one point to the next, then ending up with a shape. How "a1's point array is filled is shown in figure 4.29.

```
addpoints(a1,A(1,i),A(2,i))
addpoints(a1,B(1,i),B(2,i))
addpoints(a1,C(1,i),C(2,i))
addpoints(a1,D(1,i),D(2,i))
addpoints(a1,A(1,i),A(2,i))
```

Figure 4.29: The way the a1 point array is filled, without comments

The points put into the `al` point array are A, B, C, D and then again A, and in that specific order. This draws a line from A to B, then B to C and so on, until it ends back at A and the shape is completed.

The way the points were decided was through basic geometric equations and using the simulation results for the angle of the ankle, and the ankle height. From these two variables, the A point is decided to be at the x-coordinate= 0, while having a y-coordinate=The ankle height. This provides the A point for all points in time. The next step is to find B, meaning the tip of the foot, which is decided by the length of the foot and the angle of the ankle. Then when this point is found, the displacement from this point and to C is simply the ankle's current angle  $+\frac{\pi}{2}$  multiplied with the foot's height. Then the point and displacement to next point is added together to find the next point. This pattern is then repeated and the same pattern is applied to the leg, except the A point is shifted to make the leg and foot overlap, and that the leg doesn't depend on the ankle's angle. The geometric logic is shown in figure 4.30 and the code implementing this geometric logic on figure 4.31.

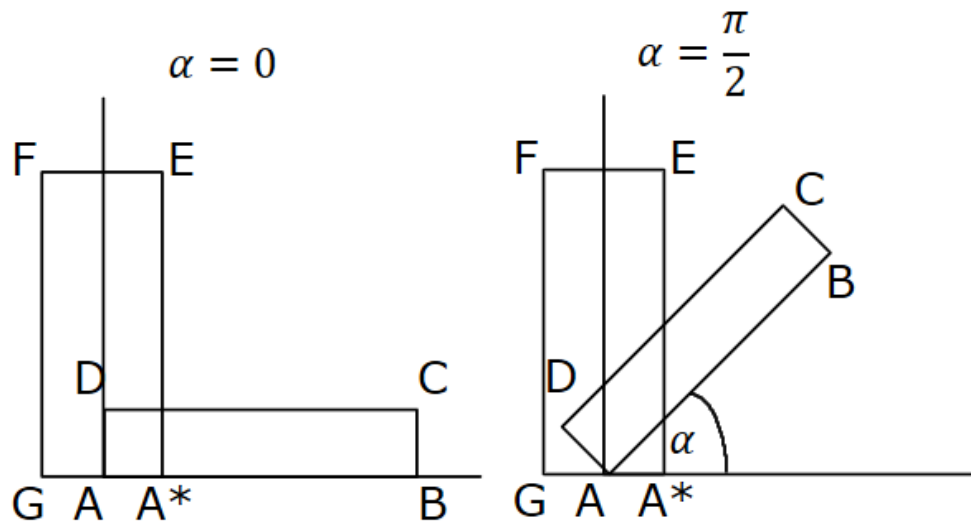


Figure 4.30: The geometric logic used to find all points for visualization, where every corner of the two rectangle's angles are  $\frac{\pi}{2}$  rad



```
AB_displacement = [foot_length*cos(ankle_angle');
  - foot_length*sin(ankle_angle')];
B = A + AB_displacement;
BC_displacement = [foot_height*cos(pi/2 + ankle_angle');
  - foot_height*sin(pi/2 + ankle_angle')];
C = B + BC_displacement;
CD_displacement = [foot_length*cos(pi + ankle_angle'); foot_length*sin(pi +
  - ankle_angle')];
D = C + CD_displacement;
```

Figure 4.31: The code used to calculate the geometric points to "draw" for the visualization, without comments

# Chapter 5

## Discussion

In this chapter we will discuss the process of this project as well as all choices we have made. We will discuss what we think of the choices now and if it might have been done differently and in so what could have been changed. We will also discuss how the process went and what we learned from it.

### 5.1 Rig

#### 5.1.1 Design and materials

Although the design of the main frame is a good solution, it still has room for improvement. No dedicated cable management solution was implemented during the design phase and therefore had to be done during the building phase, this was far from ideal and should have been planned better. The design had more than adequate room for expansion. The cable and pulley design worked as intended with no obstruction. As the frame was designed to be located at Ålesund Biomechanical Laboratory, the design was very specific and tailored to the dimensions of that room. If we were to make a more universal design it would probably be quite different.

Because of the Covid-19 pandemic, we were not allowed to work at the lab, and were therefore not able to implement the aluminium frame solution. We would of course have preferred to be able to build the originally thought-out design. However, the wooden frame worked nicely

as a proof of concept with the low loads during the initial testing, but the natural flex that comes with using wood as material would make it unusable for heavy loads or for accurate testing.

### **5.1.2 Hardware**

The actuators and RoboClaws were strong and accurate, however in retrospect for this task the actuator did not need to be as fast, and a slower and more affordable actuator could have been used instead. Another feature we should have prioritized was to choose a actuator with a built in positioning sensor, that would help with the initial positioning when testing and would remove the need for end switches. Considering the loads used in testing, the 300kg capacity of the load cells was excessive, and a more accurate result could be achieved with a lower weight rated load cell, but at the cost of versatility. The fishing line used for the testing proved capable and had no stretch, but for a more permanent solution something like a steel wire rope would be more appropriate.

### **5.1.3 Software**

When coding the software for the Arduino controller, we decided on using a a system of states, so that we could have complete control over the actuators, and at any time "reset" the system to stand-still if ever necessary. This also complimented the switching between manual control and using automatic features.

The choice to use set speeds though, were not our ideal solution, but since there was technical issues with running too low speeds on the actuators, and system control instability issues with too high speeds, we decided on choosing set speeds to make the system stable and functional even with disturbance.

### **5.1.4 Testing**

The rig was able to move the ankle in a realistic movement pattern. As revealed in section 4 Result, the load cells were accurate and only had a fluctuation  $\pm 6$  grams at a 3 kg load. These

are good results, especially considering that is at only 1% of the load cells rated capabilities. The modified ankle model had a low internal resistance and a very realistic range of motion. The model held its shape while under the low loads during testing, but would start to stretch when the load became too high, stronger elastic bands could help to prevent this.

## 5.2 Simulation

### 5.2.1 Different stages throughout the project

The first simulations simply created values that would be nice outputs to have from a model, but didn't actually simulate a system other than if the system is considered to be a completely unrealistic and arbitrarily moving one. In this simulation version, the simulation results were simply generated, and these generated results were linear, either increasing, decreasing or constant. Therefore every movement had a perfectly linear motion or exact speed. This was still quite resembling of an ankle movement, just more robotic.

Then it became a rotating with a certain speed simulation, where there was close to a 1:1 ratio from input to angular speed in the ankle. Still not really a model of a realistic system at all, but at least it was responsive to inputs instead of arbitrarily generated. In this version the model was simplified to a one-dimensional circular motion, with no regards to physical attributes or laws of physics. At this point, the simulation could be run with different inputs, showing different ankle motions.

Lastly and finally, the simulation uses a state-space model with inputs and outputs to simulate an ankle, where the model is derived from a simplified ankle. In this version an actual model is used, which clearly differentiates this version from the rest. Because of this, different inputs can be used to get different results just like in the second version, but in contrast to the last version it is restrained to the model's specifications.

### 5.2.2 Deciding on level of complexity in the model

In creating our model we did many simplifications, which had us end up with a one-dimensional mass-spring-damper system that was mathematically translated to state-space. The primary simplifications were anatomical and were because of our lack of medical and/or anatomical knowledge and experience.

Were we to use a more realistic model starting-point the resulting model would if correctly implemented, be a more accurate model. The only two reasons why we didn't decide to this was because of the amount of time we were given to learn and implement such a complex model, and the potentially overly complex model that could make the simulations slow or impossible to compute.

Specifically the skeletal simplification, the choice to make the simulation model one-dimensional was made because the goal of this simulation primarily was to simulate walking, where the front and back tendons were the dominant pullers, and the pulls on the sides were small and equal, simply keeping the foot balanced. This ended up being a quite accurate assumption since we ended up having a very similar situation in the actual prototype, where the side-pulls are quite balanced and weak in comparison to the front and back tendons which are clearly dominant.

After deciding on having a one-dimensional ankle model, the choice to use a mass-spring-damper representation of the system was both for clarity in the mathematical model, since a circular motion is trickier to track than a linear one, but also because of more experience with mass-spring-damper systems and their translations into state-space.

Instead of changing into a mass-spring-damper system, we could have implemented the same attributes to the system, calling them friction, torque etc. It would simply be a different type of physics applied to analyze the system.

## 5.3 Visualization

### 5.3.1 Different available solutions for visualizing the data

There were several options that could've been implemented. As previously mentioned, the visualization tools in Matlab are primarily focused on using figures. These figures can contain several types of input. Using the `draw()` function is a very simple and logical choice for creating geometric shapes, because usually when dealing with geometric shapes, it needs to be confined or defined as being a series of points with lines connecting them. The other primary choices were to use `plot()` or images.

The method we could've used where we use `plot()` instead requires finding the slopes, starting heights and x-limits for every line used to create the geometric figures. These would then be plotted with "hold on" which would make all the plots stack in the same figure window, and be created with color of the shape they belong to. The main issues with this method is both the added complexity from having to calculate a slope which could possibly slow down the visualization making it less usable with future implementation in live-view. Also the plotting would depending on the point-style chosen need many or few points, but either way still need at least two points per line which is almost double what is used with `draw()`. Lastly `plot()`, requires to create more variables, like the x-limits, which again, adds to complexity and slows the visualization down.

Using images of course was also a very interesting option, since it could end up having a much more intuitive image. The problem was implementing it through Matlab. Matlab did not allow two images to be used at the same time, and also each image used in the figures got its own axis that was visually unpleasing and distracting. This method was quickly scrapped because of the fact to rotate a ankle with images, needs two images, which couldn't be implemented in Matlab. The reason why two images would be necessary is that the leg and the foot would have to be able to move independently. Otherwise the ankle would just be at the same angle during the whole simulation. The only other option would be to simply have the leg or the foot in conjunction with one of the other solutions doing the other part. The primary reason why this

was scrapped as well as that the combination would be only as good or less good than simply having a clear version using only one of the other methods.

The second part of the visualization was the actual methodology of using trigonometry and displacement between points. Initially this was thought of as a solution using vectors, where these lines between points were vectors, and that through vector rotation, we could rotate the visualization of the foot. The primary difference between these solutions are that using vector rotation is more complex and require more specialized code to work in comparison to using geometric formulas directly calculating the points and displacements, at least for our use since we specifically wanted to draw simple geometric shapes.

## **5.4 Personal Experiences**

### **5.4.1 Planning**

During the start of the project, a lot of time was used for planning the development of the prototype. Especially the construction design and placement was 3D-modelled before the building and implementation started. This would have payed off if we would have been able to construct the planned rig.

### **5.4.2 Division of labor**

The two group members have quite different previous job and field experience. The different interests and backgrounds have been taken into consideration when when distributing the work to take advantage of each members expertise. This ended up in dividing the project into two main parts, one was the design and building of the rig, the other the simulation and visualization. Each member got the main responsibility for one of the parts. This ended up being a good solution.

### **5.4.3 Data collection**

As both the group members had no previous experience withing the field of medicine, the learning curve was steep. We interviewed the orthopedic surgeon at the lab when we needed guidance with anatomy. The use of this type of technology for medical research is very limited, therefore the available literature on the related topics are also limited. The medical aspects of the project is therefore mostly based on the interviews of the surgeon.

### **5.4.4 Covid-19**

Because of the Covid-19 pandemic, many of the initial plans for the project changed, most notably the fact that we were not allowed in to the lab towards the end of the project. This forced us to think outside of the box, and create a proof of concept instead. This proof of concept was less stable and somewhat harder to control, but it still worked accurately enough to satisfy our goal of making realistic walking motion.



# Chapter 6

## Conclusions

The project aimed to develop a prototype rig for performing ankle stability tests. By using new automated technologies to perform testing on human joints, new types of orthopedic surgical methods can be tested and validated. Requirements for the prototype was to be able to control a ankle to move in a realistic movement pattern, and to measure the load on the tendons during this process. The project also aims to make a digital simulation and visualisation of a ankle joint. This needed a model that was a mathematical approximation of the actual ankle complex, within reason in regards to the goal of being able to simulate walking.

The prototype accomplished the required task of controlling the ankle and measuring the load. Using linear actuators and load cells to move and measure the tendons, the results are a realistic movement and accurate load readings. The prototype that was built during this project is not a finished product. However, the solutions and ideas that was used, proved to be effective for this type of project.

Our simulation was modelled with a lot of simplifications, still it ended up having a good resemblance to the behaviour of a real foot. The system and the inputs could also have been optimized to make the model an even more realistic approximation. The visualization also worked as intended, however it could have been more complex to improve the visual clarity.

Even though the final solution did not work flawlessly and has a lot of possible improve-

ments, the group considers the project to be a success. We started from scratch when developing and building a rig for these kinds of tests. As none of the group members have any previous experience within the field of medicine, the learning curve was steep. Still, at the end of the project we are left with a lot of valuable experience regarding the technical aspects, but also regarding project management.

## 6.1 Further work

The following points are suggestions for further development of the ankle testing rig.

- Build and assemble a more permanent aluminium main frame.
- Implement positioning sensors for the actuators. Either internal or external encoders. This would also add tendon movement length data along with the load data.
- Optimize the Arduino code.
- Add the integral part of the PID controller to improve steady-state errors.
- Implement better automatic sideways motion, and in general add more automatic control to the system.
- Improve or replace the simulation model with a more accurate and complex one.
- Improve the visualization by replacing the rectangular shapes with more realistic or intuitive shapes.

# Chapter 7

## Appendices

### A Code

#### Main(), for simulation and visualization

```
clear all;
foot_length = 5;
foot_height = 1;
stepsize = 0.04;
run("Simulation_Final.m");
run("Visualization_Final.m");
```

#### Simulation

```
% Simulation
% Mass spring damper
k = 1.5; % spring constant
m = 2; % mass constant
c = 0.5; % damper constant
max = 3; % Max heigth lift and force pull
% u is the input during the simulation, and the array elements are force
- applied at individual points in time throughout the simulation
```

```

u = [0:stepsize:max max*ones(1,max/stepsize) flip(0:stepsize:max)
    - flip(-max:stepsize:0) -max*ones(1,max/stepsize) -max:stepsize:0]; %
    - input (force (F))
frames = numel(u); % Number of frames of the simulation
% ankle_height is the ankle's height during the simulation, and the array
    - elements are the height at individual points in time throughout the
    - simulation
ankle_height = [flip(max/2:stepsize/2:max) flip(0:stepsize/2:max/2)
    - zeros(1, max/stepsize) 0:stepsize/2:max/2 max/2:stepsize/2:max
    - max*ones(1,max/stepsize)];
% System
A = [0 1;
    -k/m -c/m];
B = [0 1/m]';
C = [1 0];
D = 0;
sys = ss(A,B,C,D,'StateName',{'Position' 'Velocity'}, 'InputName','Force');
    - % defining the state-space system
% Simulation
t = 0:60/numel(u):60-(60/numel(u)); % Points in time
Y = lsim(sys,u,t); % The actual simulation being run with the model, inputs
    - with regards to time, and time
ankle_angle = PosToRad(Y); % Translating the position result from the
    - mass-spring-damper simulation into a rotary format

```

## Visualization

```

% Coordinate center
A = [zeros(1,frames);
    ankle_height];

```

```

% Foot geometry
AB_displacement = [foot_length*cos(ankle_angle');
                  foot_length*sin(ankle_angle')];
B = A + AB_displacement;
BC_displacement = [foot_height*cos(pi/2 + ankle_angle');
                  foot_height*sin(pi/2 + ankle_angle')];
C = B + BC_displacement;
CD_displacement = [foot_length*cos(pi + ankle_angle');
                  foot_length*sin(pi + ankle_angle')];
D = C + CD_displacement;
% DA_displacement = [foot_length*cos(3*pi/2 + ankle_angle');
%                   foot_length*sin(3*pi/2 + ankle_angle')];

% Leg geometry
leg_height = 1.2*foot_length;
leg_width = 0.6*foot_height;
A_shifted = A + 0.7*leg_width*[ones(1,frames); zeros(1,frames)];
AE_displacement = [leg_height*cos(pi/2);
                  leg_height*sin(pi/2)];
E = A_shifted + AE_displacement;
EF_displacement = [leg_width*cos(pi);
                  leg_width*sin(pi)];
F = E + EF_displacement;
FG_displacement = [leg_height*cos(3*pi/2);
                  leg_height*sin(3*pi/2)];
G = F + FG_displacement;
% GA_displacement = [leg_width*cos(0);
%                   leg_width*sin(0)];

```

```

% Forces
j_vector = [zeros(1,frames);
            ones(1,frames)];

% Lines and figure specs
a1 = animatedline('Color', [0,1,0]); % Green line
a2 = animatedline('Color', [102/255,51/255,0]); % Brown line
a3 = animatedline('Color', [1,0,0]); % Red line
a4 = animatedline('Color', [0,0,1]); % Blue line
figure(1)
grid on
hold on
axis([-1 7 -4 11])
legend('The foot','The leg','Pull on top of foot','Pull on back of foot')
title('Simulation of walking')
for i=1:frames
    % Clears the points from the last iteration / frame
    clearpoints(a1)
    clearpoints(a2)
    clearpoints(a3)
    clearpoints(a4)

    % Foot
    addpoints(a1,A(1,i),A(2,i)) % A
    addpoints(a1,B(1,i),B(2,i)) % B
    addpoints(a1,C(1,i),C(2,i)) % C
    addpoints(a1,D(1,i),D(2,i)) % D
    addpoints(a1,A(1,i),A(2,i)) % A

    % Leg

```

```

    addpoints(a2,A_shifted(1,i),A_shifted(2,i)) % Shifted A
    addpoints(a2,E(1,i),E(2,i)) % E
    addpoints(a2,F(1,i),F(2,i)) % F
    addpoints(a2,G(1,i),G(2,i)) % G
    addpoints(a2,A_shifted(1,i),A_shifted(2,i)) % Shifted A

    % Forces
    if(u(i)<0)
        addpoints(a3, [-0.5 -0.5], -3.5+[0 abs(u(i))]) % Force pulled from
            - behind leg
    else
        addpoints(a4, [0.5 0.5], -3.5+[0 abs(u(i))]) % Force pulled from in
            - front of leg
    end
    drawnow
end

```

### PosToRad()

```

function [angle] = PosToRad(positions)
%PosToRad Translates a position in relation to a spring starting point to a
- angle
% Takes points of position and translates them to a angle using the
- formula: angle=x/(2*pi)
angle = positions/(2*pi);
end

```

### Arduino, main()

```

// Libraries
#include <HX711.h>
#include <RoboClaw.h>

```

```
#include <SoftwareSerial.h>

// Pins to the load cell amp
#define CLK1 2 // clock pin to the first load cell amp
#define CLK2 4 // clock pin to the second load cell amp
#define CLK3 6 // clock pin to the third load cell amp
#define CLK4 8 // clock pin to the fourth load cell amp
#define DOUT1 3 // data pin to the first lca
#define DOUT2 5 // data pin to the second lca
#define DOUT3 7 // data pin to the third lca
#define DOUT4 9 // data pin to the fourth lca

// Roboclaw
SoftwareSerial serial(10,11); // Communication with RoboClaw controllers
RoboClaw roboclaw(&serial,10000); // Communication with RoboClaw controllers
#define address1 0x80 // Address to RoboClaw controller
#define address2 0x81 // Address to RoboClaw controller

// Load cell readings
HX711 scale1, scale2, scale3, scale4;
int Load_Cell_Value[4];
int offset[4]; // Offsets instead of tare()

// PID Control
// The PID control gains
float Kp = 0.4;
float Kd = 0.2;
float Ki = 0;

// Defining necessary values for PID calculations
```



```
float elapsedTime[4], error[4], lastError[4], cumError[4], rateError[4],
  - out[4], desiredVal[3][4];
unsigned long Load_Cell_Read_Time[4], previousTime[4];

// For autoswitching between different motions / up, down, etc.
bool correctValues[4] = {false, false, false, false};
int currentAutoMotion = 0; // Automatic motion towards max angle = 0 and
  - towards min angle = 1 and towards neutral angle = 2
int previousMotion = 2; // Previous direction of movement in order to vary
  - correctly between

// Motion controls
bool stop_all = false; // Enable or disable actuators - false=motion enabled
  - and true=motion disabled
bool walking = false; // Automatic mode - true=simulated walking and
  - false=not simulated walking
int currentMotor; // Current motor, for manual motor control
bool manual = true; // Manual control override
int manual_control[4]={0, 0, 0, 0}; // Motion statuses for each motor during
  - manual mode
const int END_SWITCH[4]={14, 15, 16, 17}; // End switches

void setup() {
  Serial.begin(115200); // Start serial communication at 115200 baudrate

  //Open roboclaw serial ports
  roboclaw.begin(38400);
```

```
// End-switches for each actuator
pinMode(END_SWITCH[0], INPUT); // Actuator 1
pinMode(END_SWITCH[1], INPUT); // Actuator 2
pinMode(END_SWITCH[2], INPUT); // Actuator 3
pinMode(END_SWITCH[3], INPUT); // Actuator 4

scale1.begin(DOUT1, CLK1); // Front
scale2.begin(DOUT2, CLK2); // Outside-side
scale3.begin(DOUT3, CLK3); // Inside-side
scale4.begin(DOUT4, CLK4); // Back

// Setting calibration factors
scale1.set_scale(14); // Front
scale2.set_scale(14); // Outside-side
scale3.set_scale(14); // Inside-side
scale4.set_scale(14); // Back

// DESIRED VALUES for [0][x] moving up to max and [1][x] moving down to
- min
// Max angle
desiredVal[0][0]=1200; // Front
desiredVal[0][1]=150; // Outside-side
desiredVal[0][2]=150; // Inside-side
desiredVal[0][3]=50; // Back
// Min angle
desiredVal[1][0]=50; // Front
desiredVal[1][1]=150; // Outside-side
desiredVal[1][2]=150; // Inside-side
desiredVal[1][3]=400; // Back
```

```
// Neutral angle
desiredVal[2][0]=400; // Front
desiredVal[2][1]=150; // Outside-side
desiredVal[2][2]=150; // Inside-side
desiredVal[2][3]=50; // Back

// Desired value offset // Instead of taring
offset[0]=1015; // Front
offset[1]=1935; // Outside-side
offset[2]=1810; // Inside-side
offset[3]=2380; // Back
}

void loop()
{
// Serial commands
if (Serial.available()) // Checks to see if the Serial is in use
{
    char msg = Serial.read(); // Reads user inputs
    switch(msg)
    {
        // SAFETY LOCK, sets all motors to standstill
        case 'x': // STOP ALL
            for(int i=0;i<=3;i++){ // Sets current motion of controllers to
                ↳ standstill
                manual_control[i]=0;
            }
            stop_all = true;
            break;
        // Choosing the motor to manually control
    }
}
```

```
case '1': // Motor 1
    currentMotor=0; // Sets current motor to 1
    stop_all = false;
    break;
case '2': // Motor 2
    currentMotor=1; // Sets current motor to 2
    stop_all = false;
    break;
case '3': // Motor 3
    currentMotor=2; // Sets current motor to 3
    stop_all = false;
    break;
case '4': // Motor 4
    currentMotor=3; // Sets current motor to 4
    stop_all = false;
    break;

    // Chaning the state of current motor (Default is motor 1)
case 's': //Standing still
    manual_control[currentMotor] = 0; // Sets current manual motion of
    - current motor to standstill
    stop_all = false;
    break;
case 'f': //Forward
    manual_control[currentMotor] = 1; // Sets current manual motion of
    - current motor to forward/up
    stop_all = false;
    break;
case 'r': //Reverse
```

```
    manual_control[currentMotor] = 2; // Sets current manual motion of
    ↪ current motor to reverse/down
    stop_all = false;
    break;

    // Manual and automatic switching
case 'm': // MANUAL
    manual = true;
    stop_all = false;
    break;
case 'a': // AUTOMATIC
    manual = false;
    stop_all = false;
    break;

    // Specific movements/Motion patterns
case 'u': // Upwards
    walking = false;
    currentAutoMotion = 0; // Sets current automatic motion to
    ↪ upwards/max angle
    stop_all = false;
    break;
case 'd': // Downwards
    walking = false;
    currentAutoMotion = 1; // Sets current automatic motion to
    ↪ downwards/min angle
    stop_all = false;
    break;
case 'n': // Neutral
    walking = false;
```

```

        currentAutoMotion = 2; // Sets current automatic motion to
        - neutral/neutral angle
        stop_all = false;
        break;
    case 'w': // Walking
        walking = true; // Enables autoswitch between motions, and starts
        - walking simulation
        stop_all = false;
        break;
    }
}

// Update the controller with current load cell values and control mode
actuatorControl(manual);
}

// Functions
// Actuator control
void actuatorControl(bool manual){
/*
 * This function takes in one parameter, which decides whether the actuator
 - uses inputs directly chosen
 * by the user through Serial or uses inputs calculated with a PID in
 - accordance to current and previous
 * load cell readings. The motors have three speeds, 39 which is
 - reverse/down/tightening,
 * 64 which is neutral/standstill and 89 which is forward/up/loosening.
 * @params bool manual
 */
    if (manual) // Manual mode

```

```
{
  for(int i=0;i<=3;i++)
  {
    if (manual_control[i] == 0)
    {
      roboclawMovement(i,64); // Motor #(current iteration) is put at 64
        ~ speed
      // Prints to serial for observation/problemshooting
      Serial.print("Motor: ");
      Serial.print(i);
      Serial.print("is set to: ");
      Serial.println("Standing still");
      Serial.println(scale1.get_units()+offset[0]); // Prints value of
        ~ load cell 1 after adjusting for offset
      Serial.println(scale2.get_units()+offset[1]); // Prints value of
        ~ load cell 2 after adjusting for offset
      Serial.println(scale3.get_units()+offset[2]); // Prints value of
        ~ load cell 3 after adjusting for offset
      Serial.println(scale4.get_units()+offset[3]); // Prints value of
        ~ load cell 4 after adjusting for offset
    }
    else if (manual_control[i] == 1)
    {
      roboclawMovement(i,89); // Motor #(current iteration) is put at 89
        ~ speed
      // Prints to serial for observation/problemshooting
      Serial.print("Motor: ");
      Serial.print(i);
      Serial.print(" is set to: ");
      Serial.println("Forward");
    }
  }
}
```

```
    }
    else if (manual_control[i] == 2)
    {
        if(digitalRead(END_SWITCH[i]) == 1)
        {
            roboclawMovement(i,39); // Motor #(current iteration) is put at 39
            - speed
            // Prints to serial for observation/problemshooting
            Serial.print("Motor: ");
            Serial.print(i);
            Serial.print(" is set to: ");
            Serial.println("Reverse");
        }
        else
        {
            roboclawMovement(i,64); // Motor #(current iteration) is put at 64
            - speed
            // Prints to serial for observation/problemshooting
            Serial.print("Motor: ");
            Serial.print(i); // Prints which motor's end switch is being
            - pressed and therefore is restrained from moving in
            - reverse/down/tighten
            Serial.print(" is set to: ");
            Serial.println("End switch");
        }
    }
}

}

else if (!manual) // Automatic mode
```



```
{  
  
    // Prints to serial for observation/problemshooting  
    Serial.println(scale1.get_units()+offset[0]); // Prints value of load  
    - cell 1 after adjusting for offset  
    Serial.println(scale2.get_units()+offset[1]); // Prints value of load  
    - cell 2 after adjusting for offset  
    Serial.println(scale3.get_units()+offset[2]); // Prints value of load  
    - cell 3 after adjusting for offset  
    Serial.println(scale4.get_units()+offset[3]); // Prints value of load  
    - cell 4 after adjusting for offset  
    // Reset correct values from last iteration  
    for(int j=0;j<=3;j++)  
    {  
        correctValues[j]=0;  
    }  
  
    // The control loop with the PID and desired value achieved check  
    for(int i=0;i<=3;i++)  
    {  
        // Update load cell values  
        Load_Cell_Value[0] = scale1.get_units()+offset[0]; // Value of  
        - load cell 1 after adjusting for offset  
        Load_Cell_Read_Time[0] = millis(); // Time of reading  
        Load_Cell_Value[1] = scale2.get_units()+offset[1]; // Value of  
        - load cell 2 after adjusting for offset  
        Load_Cell_Read_Time[1] = millis(); // Time of reading  
        Load_Cell_Value[2] = scale3.get_units()+offset[2]; // Value of  
        - load cell 3 after adjusting for offset  
        Load_Cell_Read_Time[2] = millis(); // Time of reading
```

```

Load_Cell_Value[3] = scale4.get_units()+offset[3]; // Value of
- load cell 4 after adjusting for offset
Load_Cell_Read_Time[3] = millis(); // Time of reading
out[i] = PID(i, Load_Cell_Read_Time[i]);
// Checks with tresholds at -30 grams and 30 grams
if((currentAutoMotion==1 && i==3 &&
- out[3]<0) || (currentAutoMotion==0 && i==0 &&
- out[0]<0) || (out[i]<=30 && out[i]>=-30))
{
    out[i]=0;
}
// Prints to serial for observation/problemshooting
Serial.print(i);
Serial.print(" has pid output ");
Serial.println(out[i]); // 64 is neutral AKA 0 speed
if(out[i]>0) // Checks if the force error is positive and tightens
- to get smaller force error
{
    if(digitalRead(END_SWITCH[i]) == 1) // Checks the end switch of
- actuator #(current iteration), if switch is pressed then
- actuators are stopped and digitalRead=0
    {
        if (stop_all) // Checks if stop all is true and put actuator
- #(current iteration) in standstill if that is the case
        {
            roboclawMovement(i,64); // Motor #(current iteration) is
- put at 64 speed
        }
    }
else
{

```

```
        roboclawMovement(i,39); // Motor #(current iteration) is put
        - at 39 speed
    }
}
else
{
    roboclawMovement(i,64); // Motor #(current iteration) is put
    - at 64 speed
    // Prints to serial for observation/problemshooting
    Serial.print("Motor: ");
    Serial.print(i); // Prints which motor's end switch is being
    - pressed and therefore is restrained from moving in
    - reverse/down/tighten
    Serial.println(" End switch");
}
}
else if(out[i]<0) // Checks if the force error is negative and
- loosens to get smaller force error
{
    if (stop_all) // Checks if stop all is true and put actuator
    - #(current iteration) in standstill if that is the case
    {
        roboclawMovement(i,64); // Motor #(current iteration) is put
        - at 64 speed
    }
    else
    {
        roboclawMovement(i,89); // Motor #(current iteration) is put
        - at 89 speed
    }
}
```

```
}  
else  
{  
  roboclawMovement(i,64); // Motor #(current iteration) is put at  
  - 64 speed  
  correctValues[i]=true;  
  if(walking)  
  {  
    if(correctValues[0] && correctValues[1] && correctValues[2] &&  
    - correctValues[3])  
    {  
      if(currentAutoMotion == 0) // Checks if currently the  
      - automatic motion is set to max angle  
      {  
        currentAutoMotion = 2; // Go from max angle to neutral  
        previousMotion = 0; // Set previous direction  
      }  
      else if (currentAutoMotion == 1) // Checks if currently the  
      - automatic motion is set to min angle  
      {  
        currentAutoMotion = 2; // Go from min angle to neutral  
        previousMotion = 1; // Set previous direction  
      }  
      else if (currentAutoMotion == 2) // Checks if currently the  
      - automatic motion is set to neutral angle  
      {  
        if (previousMotion == 0)  
        {  
          currentAutoMotion = 1; // Go from neutral to max  
        }  
      }  
    }  
  }  
}
```

```

        else if(previousMotion == 1)
        {
            currentAutoMotion = 0; // Go from neutral to min
        }
    }
}
}
}
}
}
}
}
}
else
{
    // Prints to serial for observation/problemshooting
    Serial.println("NOT IN AUTOMATIC NOR MANUAL CONTROL ERROR");
    delay(60000); // Pauses code for 1 minute
}
}

// PID Control
float PID(int motor, unsigned long currentTime){
    /*
     * This function takes in two parameters, (int motor, unsinged long
     * - currentTime)
     * which uses the current time and
     * decides which motors load-cell value to read
     * and then outputs the correction value
     * @params int motor, unsigned long currentTime
     */
    // Time values
    elapsedTime[motor] = currentTime - previousTime[motor];

```

```
// ERROR VALUE
error[motor] = desiredVal[currentAutoMotion][motor] -
    - Load_Cell_Value[motor];
// Cumulative error AKA Integral control
cumError[motor] += error[motor] * elapsedTime[motor];

// Rate of change AKA Derivative control
rateError[motor] = (error[motor] - lastError[motor])/elapsedTime[motor];

// Retaining information for next iteration
previousTime[motor] = currentTime;
lastError[motor] = error[motor];

// The controller formula for output
float pid_out = Kp * error[motor] + Ki * cumError[motor] + Kd *
    - rateError[motor];
return pid_out;
}

// Roboclaw movement
void roboclawMovement(int motor, float velocity){
    /*
     * This function takes in two parameters, (int motor, float velocity) and
     - uses those to decide a speed to
     * set a motor on, and which motor. The first parameter is the input into
     - a switch case where each case
     * to use the second parameter as speed to set the motor speed of the
     - motor that is connected to that
     * case. Case 0 = Motor 1, Case 1 = Motor 2...
```

```
    * @params int motor, float velocity
    */
switch(motor)
{
    case 0: // Motor 1
        roboclaw.ForwardBackwardM1(address1,velocity);
        break;
    case 1: // Motor 2
        roboclaw.ForwardBackwardM2(address1,velocity);
        break;
    case 2: // Motor 3
        roboclaw.ForwardBackwardM1(address2,velocity);
        break;
    case 3: // Motor 4
        roboclaw.ForwardBackwardM2(address2,velocity);
        break;
}
}
```

## B Data-sheets



# Datasheet

## RS PRO

### Stock numbers:

1774499

1774504

1774505

1774509

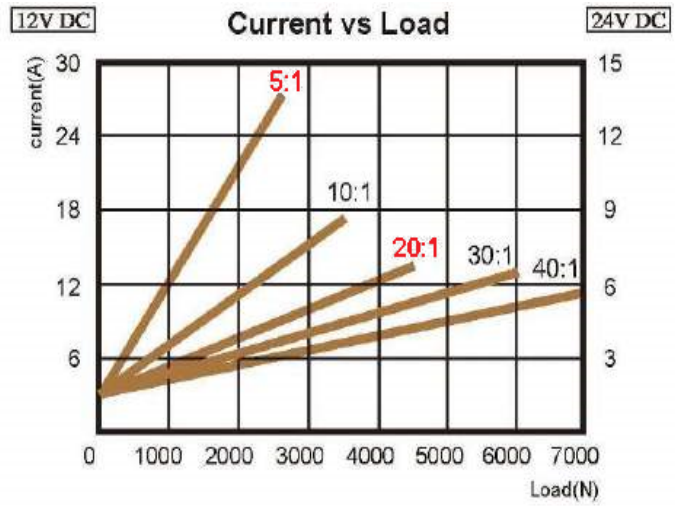
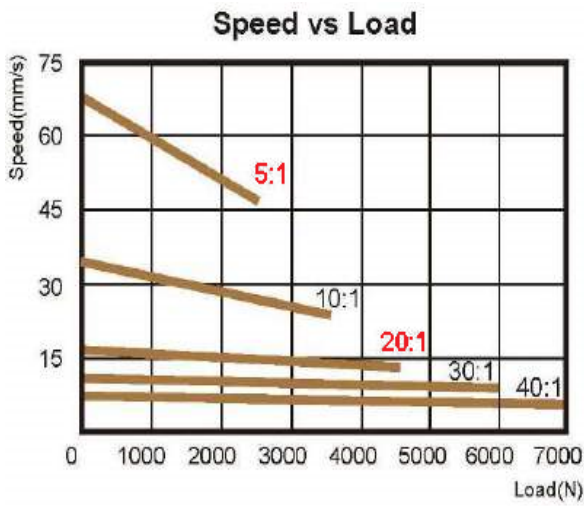
EN



- ◆ ID10 features its heavy load capability and high speed design, which is suitable for various industrial applications requiring rapid movement.
- ◆ Input voltage : 24V DC
- ◆ Max. rated load : 7,000N (Ball screw)
- ◆ Max. static load : 13,600N (Ball screw)
- ◆ Max. current : 13.2A @ 24V DC
- ◆ Max. speed : 67.1 mm/sec @ no load
- ◆ Stroke : 102~610 mm
- ◆ Duty cycle : 25%, max 2 min continuous operation in 8 min.
- ◆ IP protection level : IP54
- ◆ Colour : Black
- ◆ Certified : CE marking, EMC Directive 2014/30/EU
- ◆ Overload protection by clutch
- ◆ Power wire length : 250mm (with tinned wires)
- ◆ Operating temperature : -25°C~+65°C
- ◆ Extension tube material : Stainless steel (Ball screw)



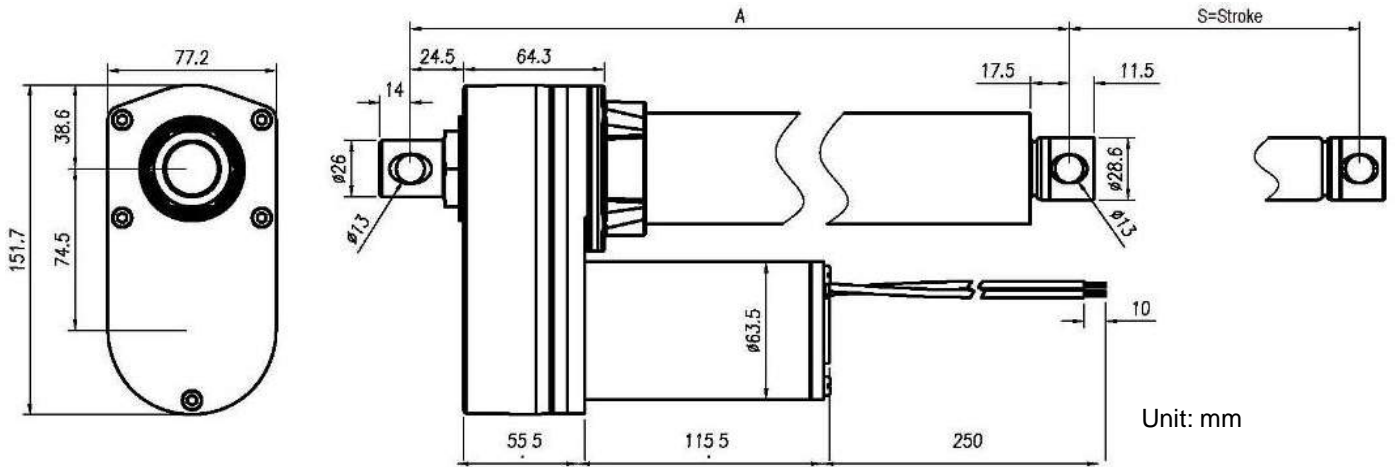
RS Number	Vendor Part Number	Push/Pull Max. (N)	Speed (mm/s)		Voltage (DC)	Stroke (mm)
			No load	Full load		
1774499	ID10-24-20-B-305	4500	16.8	14.3	24	305
1774504	ID10-24-20-B-102	4500	16.8	14.3	24	102
1774505	ID10-24-05-B-305	2500	67.1	47.2	24	305
1774509	ID10-24-05-B-102	2500	67.1	47.2	24	102



	4"	6"	8"	10"	12"	18"	24"
Stroke ( $\pm 2.5$ mm)	102	153	203	254	305	457	610
Retracted Length (A $\pm 3.8$ mm)	262	313	364	414	465	668	821

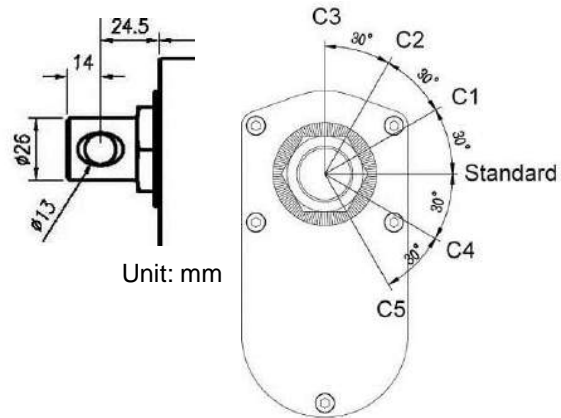
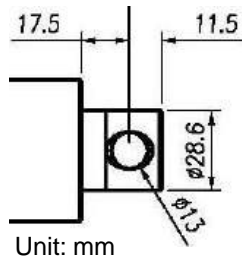


Model No.	Push/Pull Max. (N)	Speed (mm/s)		Current (A)			
		No load	Full load	No load		Full load	
				12V	24V	12V	24V
ID10-XX-05-B-XXX	2500	67.1	47.2	3.4	2.2	26.4	13.2
ID10-XX-10-B-XXX	3500	33.5	26.7	3.4	2.2	17.6	8.8
ID10-XX-20-B-XXX	4500	16.8	14.3	3.4	2.2	13.2	6.6
ID10-XX-30-B-XXX	6000	11.2	9.8	3.4	2.2	12.1	6.1
ID10-XX-40-B-XXX	7000	8.4	7.4	3.4	2.2	11.0	5.5

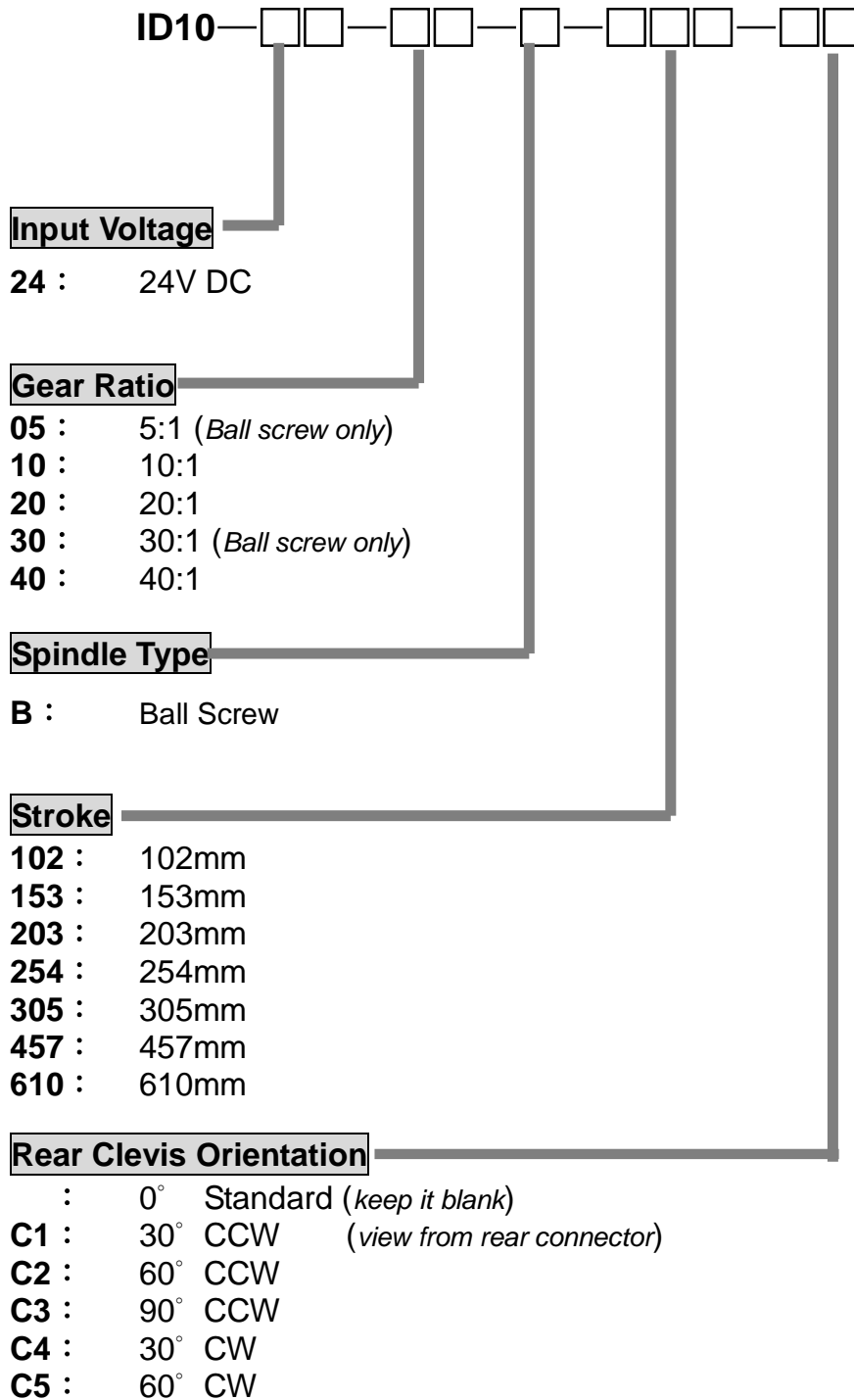


◆ Rear connector and the orientation of clevis

◆ Front connector

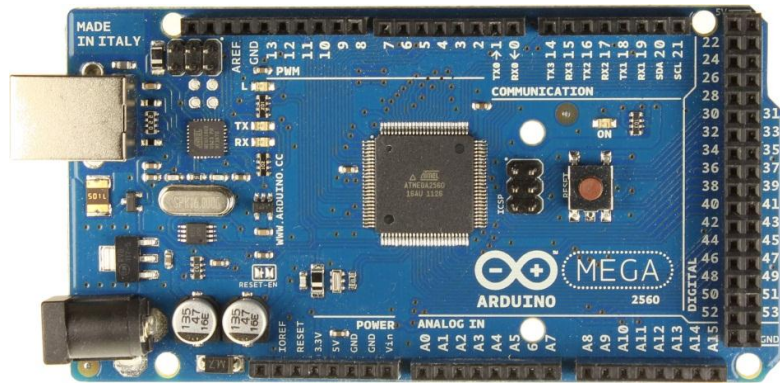


# Ordering Key



## Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 ([datasheet](#)). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.



The Mega 2560 is an update to the [Arduino Mega](#), which it replaces. The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 boards) programmed as a USB-to-serial converter.

The Mega 2560 is an update to the [Arduino Mega](#), which it replaces.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 boards) programmed as a USB-to-serial converter.

**Revision 3** of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

## Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

## Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's

power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

**VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

**5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

**3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

**GND.** Ground pins.

**IOREF.** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

## Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

## Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

**Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

**External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.

**PWM: 2 to 13 and 44 to 46.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

**SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the [SPI library](#). The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.

**LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

**TWI: 20 (SDA) and 21 (SCL).** Support TWI communication using the [Wire library](#). Note that these pins are not in the same location as the TWI pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and [analogReference\(\)](#) function.

There are a couple of other pins on the board:

**AREF.** Reference voltage for the analog inputs. Used with `analogReference()`.

**Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

## Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 (ATmega 8U2 on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](#) allows for serial communication on any of the Mega2560's digital pins.

The ATmega2560 also supports TWI and SPI communication. The Arduino software includes a Wire library to simplify use of the TWI bus; see the [documentation](#) for details. For SPI communication, use the [SPI library](#).

## Programming

The Arduino Mega can be programmed with the Arduino software ([download](#)). For details, see the [reference](#) and [tutorials](#).

The ATmega2560 on the Arduino Mega comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available [in the Arduino repository](#). The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use [Atmel's FLIP software](#) (Windows) or the [DFU programmer](#) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See [this user-contributed tutorial](#) for more information.

## Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via

USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

### **USB Overcurrent Protection**

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

### **Physical Characteristics and Shield Compatibility**

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I<sup>2</sup>C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*



	1	2	3	4	5	6	7	8
A								
B								
C								
D								
E								
	1	2	3	4	5	6	7	8

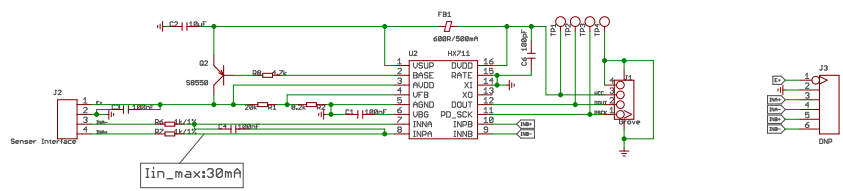
REV	Description	DATE	BY
1	Grove - ADC for load cell (HX711) v1.0	2019/04/28	Milo

# Seed Studio

TITLE: Grove - ADC for load cell (HX711)

CC-BY-SA	Design: Milo	Check:
	Date: ****	Vision:
		Sheet: 1/1





# Seed Studio

TITLE: Grove - ADC for load cell (HX711)

CC-BY-SA	Design#110	Check:
	Date: 111111	Vision:
		Sheet: 1/1

## FEATURES

- Low profile
- Alloy steel construction with nickel plated treatment
- Capacity: 300kg
- Easy to install, stable & reliable

# RS PRO Alloy Steel S Beam Tension Load Cell

RS Stock No.: 204-2768



RS Professionally Approved Products bring to you professional quality parts across all product categories. Our product range has been tested by engineers and provides a comparable quality to the leading brands without paying a premium price.

### Product Description

S Beam Load Cell 614 offers a compact design for platforms, silos, or scales. The tension Load Cell has a wide range of capacities from 50kg to 1000kg, comes in steel alloy construction.

Capacities: 300kg

Material: Alloy Steel

Dimension: 62.1\*18\*80mm

Rated Output: 2.0mV/V

Protection Class: IP66

### Applications

Hopper scales, Crane scales, Tension control and machine testing Batching systems, blending & mixing systems, belt scales, etc.

### Electrical Specifications

Rated Capacities	300kg
Accuracy Class	C3
Rated Output	2.0±0.002 mV/V
Zero Balance	±0.0200 mV/V
Non-linearity	0.03%R.O.
Hysteresis	0.03%R.O.
Repeatability	0.03%R.O.
30mins Creep	0.03%R.O.
30mins Return	0.03%R.O.
Safe Overload	120 %R.O.
Ultimate Overload	200 %R.O.
Temperature Effect on Output	0.002 %R.O./°C
Temperature Effect on Zero	0.003 %R.O./°C
Input Impedance	400±20 Ω
Output Impedance	350±3 Ω

## Load Cells



Insulation Impedance	≥5000 MΩ/(50VDC)
Recommended Excitation	4~12 VDC
Maximum Excitation	15 VDC
Operating Temperature Range	-20~60 °C
Construction	Steel Alloy
Cable	φ5×3m
Mode of Connection	Red (EXC+) , Black (EXC-) , Green (SIG+) , White (SIG-)

### Protection Category

Waterproof class	IP66
------------------	------

### Classification

eCl@ss (Version)	C3
------------------	----

### Approvals

Declarations	ROHS, OIML
--------------	------------

### Additional Information

Custom Tariff Number	84239000
----------------------	----------

# PJA1500F

PJ A 1500 F - □

① ② ③ ④ ⑤



Example recommended EMI/EMC filter  
NAC-20-472



High voltage pulse noise type : NAP series  
Low leakage current type : NAM series

- ① Series name
- ② Single output
- ③ Output wattage
- ④ Universal input
- ⑤ Output voltage

\*Make sure necessary tests will be carried out on your end equipment with the power supply installed in accordance with any required EMC/EMI regulations.

## SPECIFICATIONS

	MODEL	PJA1500F-24	PJA1500F-48	
INPUT	VOLTAGE[V]	AC85 - 264 1 φ (Output derating is required at AC85V - 115V. See 1.1 and 3.2 in Instruction Manual)		
	CURRENT[A]	ACIN 100V	18typ (Io=90%)	
		ACIN 115V	16typ (Io=100%)	
		ACIN 230V	8typ (Io=100%)	
	FREQUENCY[Hz]	50 / 60 (47 - 63)		
	EFFICIENCY[%]	ACIN 100V	84typ (Io=90%)	84typ (Io=90%)
		ACIN 115V	85typ (Io=100%)	84typ (Io=100%)
		ACIN 230V	88typ (Io=100%)	87typ (Io=100%)
	POWER FACTOR	ACIN 100V	0.98typ (Io=90%)	
		ACIN 115V	0.98typ (Io=100%)	
ACIN 230V		0.95typ (Io=100%)		
INRUSH CURRENT[A]	ACIN 100V	15/30typ (Io=90%) (Primary inrush current /Secondary inrush current) (More than 10sec to re-start)		
	ACIN 115V	15/30typ (Io=100%) (Primary inrush current /Secondary inrush current) (More than 10sec to re-start)		
	ACIN 230V	30/30typ (Io=100%) (Primary inrush current /Secondary inrush current) (More than 10sec to re-start)		
LEAKAGE CURRENT[mA]	1.5max (ACIN 240V, 60Hz, Io=100%, According to IEC62368-1 and DEN-AN)			
OUTPUT	VOLTAGE[V]	24	48	
	CURRENT[A]	ACIN 85-115V	Output derating is required at ACIN 115V or less (refer to instruction manual 3.2)	
		ACIN 115V-264V	64	32
	WATTAGE[W]	ACIN 85-115V	Output derating is required at ACIN 115V or less (refer to instruction manual 3.2)	
		ACIN 115V-264V	1536	1536
	LINE REGULATION[mV]	*2	96max	192max
	LOAD REGULATION[mV]	*2	150max	300max
	RIPPLE[mVp-p]	0 to +50°C	120max	200max
		*1 -20 to 0°C	160max	500max
	RIPPLE NOISE[mVp-p]	0 to +50°C	150max	300max
		*1 -20 to 0°C	270max	600max
	TEMPERATURE REGULATION[mV]	0 to +50°C	240max	480max
		-20 to +50°C	290max	600max
	DRIFT[mV]	*3	96max	192max
	START-UP TIME[ms]	800typ (ACIN 115V, Io=100%)		
HOLD-UP TIME[ms]	20typ (ACIN 115V, Io=100%)			
OUTPUT VOLTAGE ADJUSTMENT RANGE[V]	20.40 to 28.50	40.80 to 55.20		
OUTPUT VOLTAGE SETTING[V]	24.00 to 24.96	48.00 to 49.92		
PROTECTION CIRCUIT AND OTHERS	OVERCURRENT PROTECTION	Works over 105% of rating and recovers automatically		
	OVERVOLTAGE PROTECTION[V]	28.80 to 34.80	57.00 to 67.20	
	OPERATING INDICATION	LED (Green)		
ISOLATION	INPUT-OUTPUT	AC3,000V 1minute, Cutoff current = 25mA, DC500V 50MΩ min (At room temperature)		
	INPUT-FG	AC2,000V 1minute, Cutoff current = 25mA, DC500V 50MΩ min (At room temperature)		
	OUTPUT-FG	AC500V 1minute, Cutoff current = 100mA, DC500V 50MΩ min (At room temperature)		
ENVIRONMENT	OPERATING TEMP., HUMID. AND ALTITUDE *4	-20 to +70°C (Output derating is required), 20 - 90%RH (Non condensing), 3,000m (10,000 feet) max		
	STORAGE TEMP., HUMID. AND ALTITUDE	-20 to +75°C, 20 - 90%RH (Non condensing), 9,000m (30,000 feet) max		
	VIBRATION	10 - 55Hz, 19.6m/s <sup>2</sup> (2G), 3minutes period, 60minutes each along X, Y and Z axes		
	IMPACT	196.1m/s <sup>2</sup> (20G), 11ms, once each X, Y and Z axes		
SAFETY AND NOISE REGULATIONS	AGENCY APPROVALS	UL62368-1, C-UL (CSA62368-1), EN62368-1, Complies with DEN-AN		
	CONDUCTED NOISE	Complies with FCC-A, VCCI-A, CISPR22-A, EN55011-A, EN55022-A, additional EMI/EMC Filter required for meeting class B		
	HARMONIC ATTENUATOR *5	Complies with IEC61000-3-2 class A		

## SPECIFICATIONS

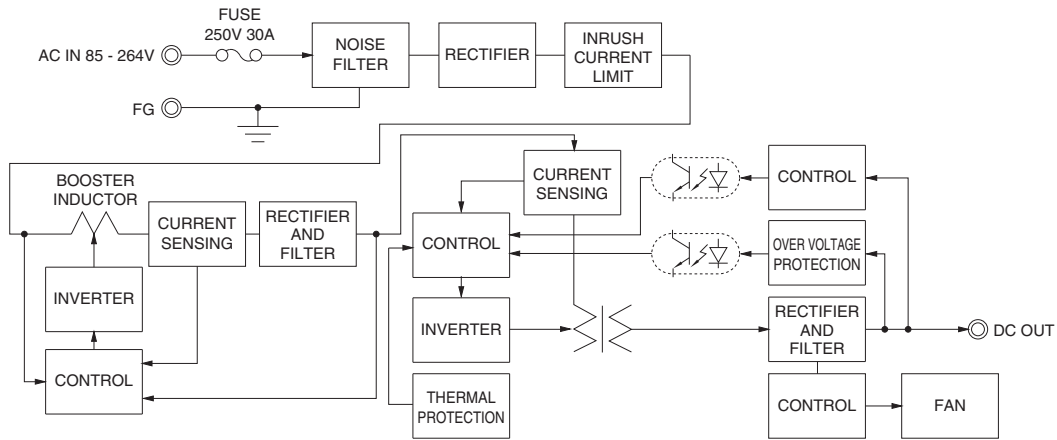
OTHERS	CASE SIZE/WEIGHT	178 × 61 × 268mm [7.01 × 2.40 × 10.55 inches] (Excluding terminal block and screw) (W × H × D) / 3.5kg max
	COOLING METHOD	*6 Forced cooling (internal fan)
WARRANTY	WARRANTY	*7 5 years (subject to the operating conditions)

- \*1 This is the result of measurement of the testing board with capacitors of 22 μF and 0.1 μF placed at 150 mm from the output terminals by a 20 MHz oscilloscope or a ripple-noise meter equivalent to Keisoku-Giken RM103.  
See 1.6 of Instruction Manual for more details.
- \*2 Consult us about dynamic load and input response.
- \*3 Drift is the change in DC output for an eight hour period after a half-hour warm-up at 25°C.
- \*4 Output power derating is required. See 3.2 in Instruction Manual.
- \*5 Consult us about other classes.
- \*6 The fan speed slows down or stops at no load.
- \*7 See 3.3 in Instruction Manual for more details.
- \* Do not use the power supply in overcurrent conditions or in unspecified input voltage ranges. Otherwise the internal components may be damaged.
- \* Parallel operation is not possible with this mode.
- \* Sound noise may be heard from the power supply when used for pulse load.

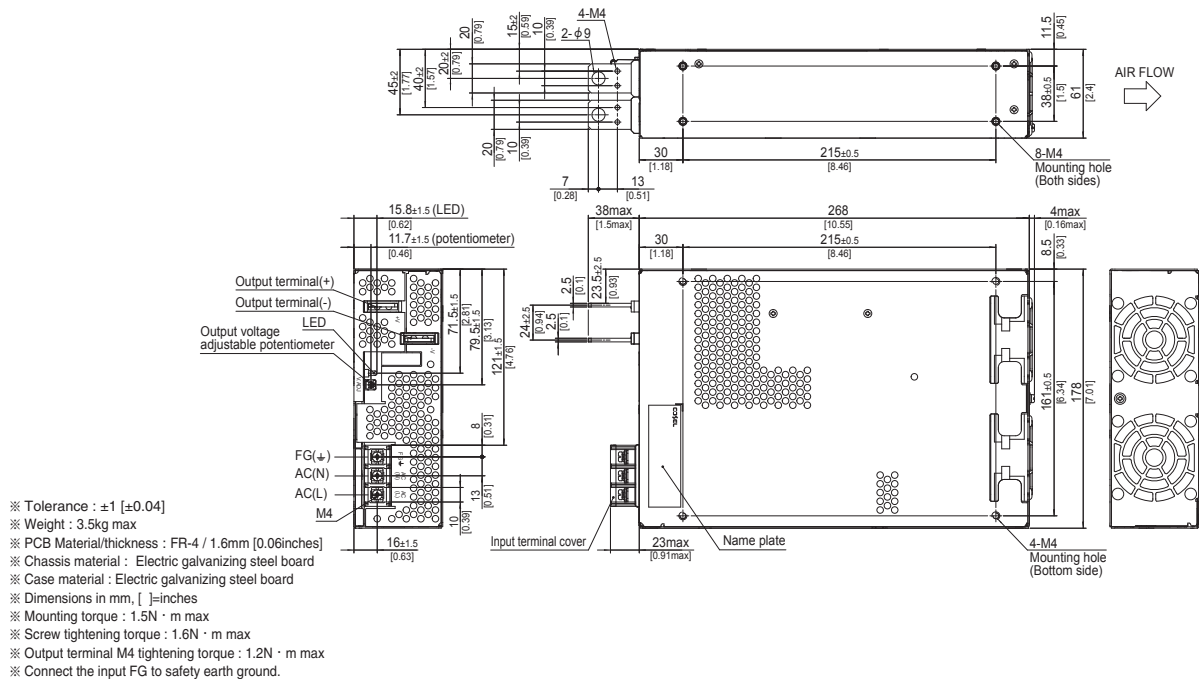
## Features

- Cost-effective
- Wide operating temperature range (-20°C to +70°C see instruction manual)
- Longer life (see Instruction Manual)
- Stop or slow fan speed at no load
- Low profile (meets 2U height = 61 mm or 2.4 inches)

## Block diagram



## External view



# HS Aluminium Housed Resistors



Manufactured in line with the requirements of MIL 18546 and IEC 115, designed for direct heatsink mounting with thermal compound to achieve maximum performance.



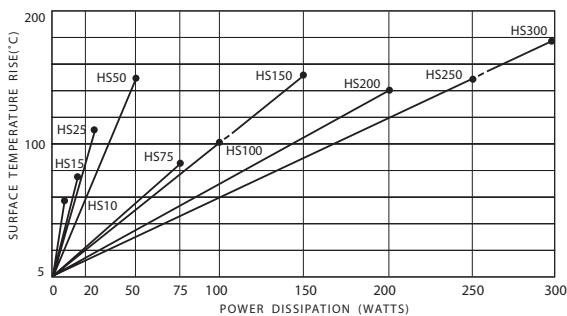
- High Power to volume
- Wound to maximise High Pulse Capability
- Values from R005 to 100K
- Custom designs welcome
- RoHS Compliant

## Characteristics

Tolerance (Code):	Standard $\pm 5\%$ (J) and $\pm 10\%$ (K). Also available $\pm 1\%$ (F), $\pm 2\%$ (G) and $\pm 3\%$ (H)
Tolerance for low $\Omega$ values:	Typically $\geq R05 \pm 5\% \leq R047 \pm 10\%$
Temperature coefficients:	Typical values $< 1K$ 100ppm Std. $> 1K$ 25ppm Std. For lower TCR's please contact Arcol
Insulation resistance (Dry):	10,000 M $\Omega$ minimum
Power dissipation:	At high ambient temperature dissipation derates linearly to zero at 200°C
Ohmic values:	From R005 to 100K depending on wattage size
Low inductive (NHS):	Specify by adding N before HS Series code, e.g. NHS50
NHS ohmic value:	Divide standard HS maximum value by 4
NHS working volts:	Divide standard HS maximum working voltage by 1.414

## Temp. Rise & Power Dissipation

Surface temperature of resistor related to power dissipation. The resistor is standard heatsink mounted using a proprietary heatsink compound.



## Heat Dissipation

Heat dissipation: Whilst the use of proprietary heat sinks with lower thermal resistances is acceptable, uprating is not recommended. For maximum heat transfer it is recommended that a heat sink compound be applied between the resistor base and heat sink chassis mounting surface. It is essential that the maximum hot spot temperature of 200°C is not exceeded, therefore, the resistor must be mounted on a heat sink of correct thermal resistance for the power being dissipated.

## Ordering Procedure

Standard Resistor. To specify standard: Series, Watts, Ohmic Value, Tolerance Code, e.g.: HS25 2R2 J

Non Inductive Resistor. To specify add N, e.g.: NHS100 10R J

ARCOL UK Limited,  
Threemilestone Ind. Estate,  
Truro, Cornwall, TR4 9LG, UK.  
T +44 (0) 1872 277431  
F +44 (0) 1872 222002  
E sales@arcolresistors.com

[www.arcolresistors.com](http://www.arcolresistors.com)

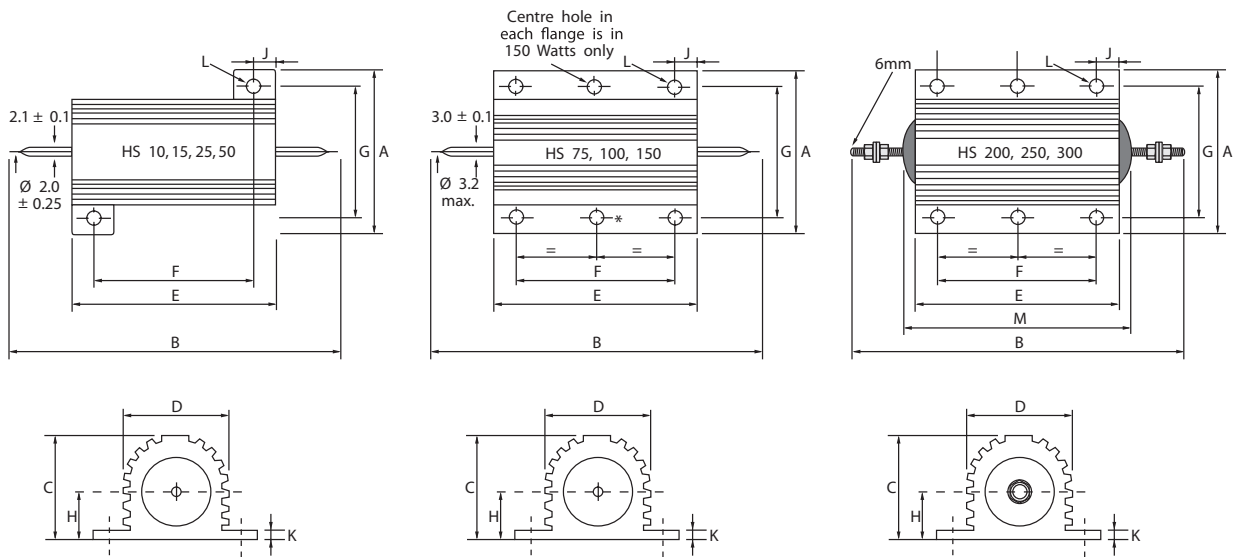
The information contained herein does not form part of a contract and is subject to change without notice. ARCOL operate a policy of continual product development, therefore, specifications may change.

It is the responsibility of the customer to ensure that the component selected from our range is suitable for the intended application. If in doubt please ask ARCOL.

## Electrical Specifications

Size	Style MILR 18546	Power rating on std. heatsink @25°C	Watts with no heatsink @25°C	Resistance range	Limiting element voltage	Voltage proof AC Peak	Voltage proof AC rms.	Approx weight gms	Typical surface rise HS mounted	Standard heatsink	
										cm <sup>2</sup>	Thickness mm
HS10	RE 60	10	5	R005-10K	160	1400	1000	4	5.8	415	1
HS15	RE 65	15	7	R005-10K	265	1400	1000	7	5.1	415	1
HS25	RE 70	25	9	R005-36K	550	3500	2500	14	4.2	535	1
HS50	RE 75	50	14	R01-86K	1250	3500	2500	32	3.0	535	1
HS75		75	24	R01-50K	1400	6363	4500	85	1.1	995	3
HS100		100	30	R01-70K	1900	6363	4500	115	1.0	995	3
HS150		150	45	R01-100K	2500	6363	4500	175	1.0	995	3
HS200		200	50	R01-50K	1900	7070	5000	475	0.7	3750	3
HS250		250	55	R01-50K	2200	7070	5000	600	0.6	4765	3
HS300		300	60	R01-68K	2500	7070	5000	700	0.6	5780	3

## HS10-HS300 Standard Resistor



## Dimensions (mm)

Size	A Max	B Max	C Max	D Max	E Max	F±0.3	G±0.3	H Max	J Max	K Max	L ±0.25*	M Max
HS10	16.5	30.0	8.8	8.5	15.9	11.3	12.4	4.5	2.4	1.8	2.4	
HS15	21.0	36.5	11.0	11.2	19.9	14.3	15.9	5.5	2.8	1.8	2.4	
HS25	28.0	51.0	14.8	14.2	27.3	18.3	19.8	7.7	5.2	2.6	3.2	
HS50	28.0	72.5	14.8	14.2	49.1	39.7	21.4	8.4	5.2	2.6	3.2	
HS75	47.5	72.0	24.1	27.3	48.7	29.0	37.0	11.8	10.4	3.7	4.4	
HS100	47.5	88.0	24.1	27.3	65.2	35.0	37.0	11.8	15.4	3.7	4.4	
HS150	47.5	121.0	24.1	27.3	97.7	58.0	37.0	11.8	20.4	3.7	4.4	
HS200	72.5	145.7	41.8	45.5	89.7	70.0	57.2	20.5	10.4	5.5	5.1	103.4
HS250	72.5	167.0	41.8	45.5	109.7	89.0	57.2	20.5	10.4	5.5	5.1	122.4
HS300	72.5	184.4	41.8	45.5	127.7	104.0	59.0	20.5	12.4	5.5	6.6	141.4

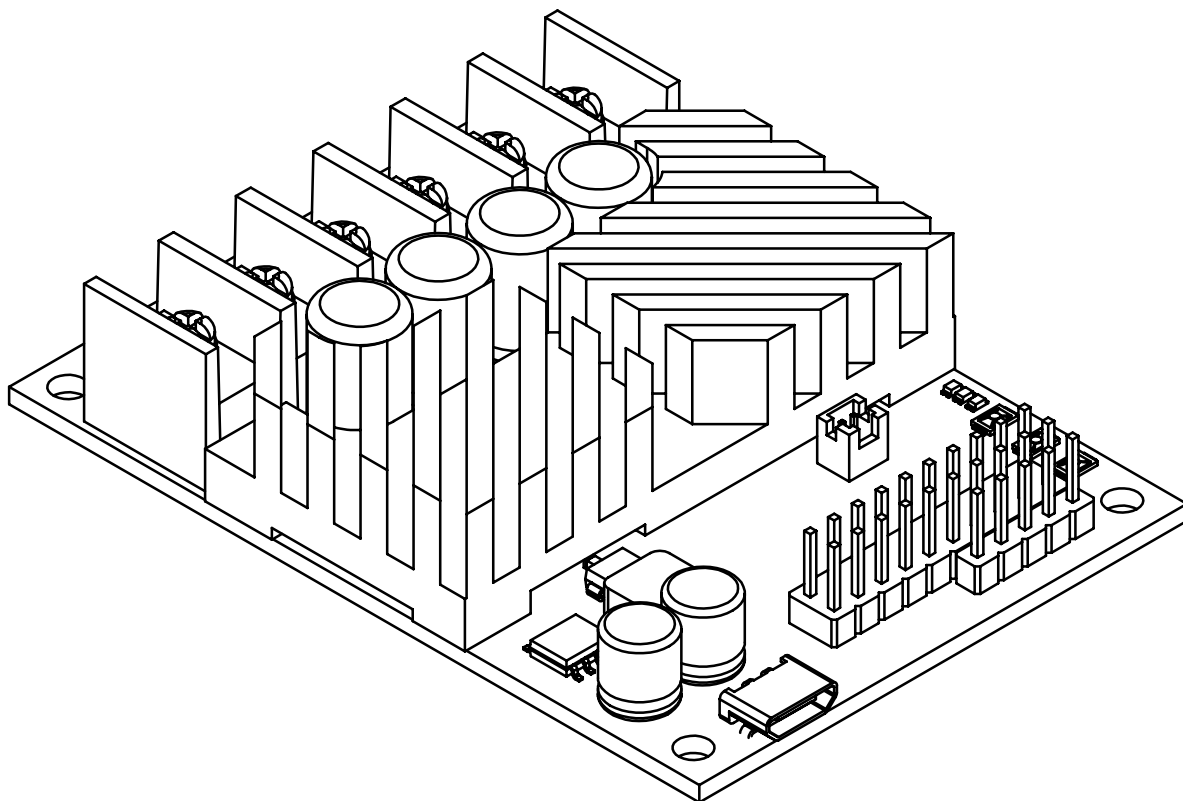
\* HS200-HS300 Watts is ± 0.45





# BASICMICRO

## MOTION CONTROL

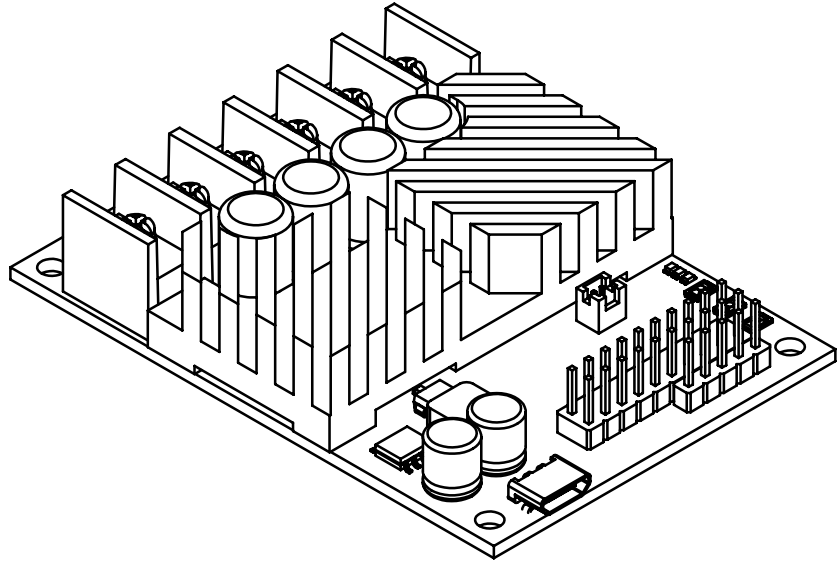


### RoboClaw 2x15A, 34VDC Dual Channel Brushed DC Motor Controller

Data Sheet Version 2.4

## Feature Overview:

- 15 Amps Continuous Per Channel
- 30 Amps Peak Per Channel
- Dual Quadrature Decoding
- 9.8 million PPS Decoding
- Multimode Interface
- TTL Serial
- USB Port
- Analog Interface
- R/C Input Control
- Limit, Home and E-Stops
- Up to 34VDC Operation
- Cooling Fan With Automatic Control
- 3.3v Compliant Control Outputs
- 5v Tolerant Control Inputs
- Programmable Current Limiting
- Programmable Voltage Clamping
- Closed and Open Loop Operation
- Auto Tuning PID Feature
- Mixed Control Modes
- Data Logging
- Diagnostic LEDs
- Field Firmware Updates
- Regulated 5VDC, 3A User Available Output
- Over Voltage and Under Voltage Protection
- Easy Tuning, Monitor and Setup with PC utility



## Device Overview

The RoboClaw is an intelligent, high performance motor controller designed to control dual brushed DC motors. It can be controlled from USB, RC radio, PWM, TTL serial, analog and microcontrollers such as an Arduino or Raspberry Pi.

RoboClaw automatically supports 3.3V or 5V logic levels, travel limit switches, home switches, emergency stop switches, power supplies, braking systems and contactors. A built-in switching mode BEC supplies 5VDC at up to 3 Amps for powering user devices. In addition power supplies can be utilized by enabling the built in voltage clamping control feature.

A wide variety of feedback sensors are supported. This includes quadrature encoders, potentiometers and absolute encoders which can be easily configured using the available auto tune function. With sensors, two brushed DC motors can be controlled in closed loop mode allowing precise control over position and speed. With the ability to use potentiometers, servo systems can be created and controlled from any of RoboClaw's interface modes.

For greater control, built-in commands are available for controlling acceleration, deceleration, distance, speed, current sense, voltage limits and more. In addition, RC and analog modes can be configured by user defined settings to control acceleration and deceleration rates.

RoboClaw incorporates multiple protection features including temperature, current, over voltage and under voltage limits. The protection features are self monitoring and protect RoboClaw from damage in any operating condition. User definable settings such as maximum current limit, maximum and minimum battery voltages are provided for more refined control.

RoboClaw's regenerative capabilities will charge a supply battery during slow down or braking. It's advance circuitry can change direction during full throttle without damage! RoboClaw also incorporates a LiPo cutoff mode to prevent battery damage.

## Multimode Interface

RoboClaw's I/O are voltage protected and can handle up to 5VDC. The I/O only output a high of 3.3V. This allows RoboClaw to be interfaced to 5V or 3V logic easily with no translation circuits required. RoboClaw can be connected directly to a Raspberry Pi or Arduino. All of RoboClaw's inputs are internally pulled-up to prevent false triggers. Inputs can also be configured using the Motion Studio application.

### **User Regulated Power Output**

RoboClaw provides regulated power (BEC) for user devices. A high efficiency switching regulator supplies 5VDC at up to 3 Amps. This voltage can be used to power external sensors, encoders, MCUs and other electronics. The regulated user power is automatically current limited and thermally protected.

### **Main Battery**

The peak operational input voltage depending on the model can be up to 34VDC, 60VDC or 80VDC. The models maximum input voltage can not be exceeded. If the maximum voltage is exceeded the motors will be disabled. Fully charged batteries maximum voltage must be taken into account when in use. RoboClaw is a regenerative motor controller. During regeneration, voltages can peak over the maximum rated voltage in which RoboClaw is designed to handle these over voltage spikes by braking the motors.

### **Logic Battery**

RoboClaw accepts a logic battery. The logic battery is also known as a backup battery. The user regulated power output (BEC) is by default powered from the main battery, unless a logic battery is detected. The logic battery source is coupled to the main battery through an on board automatic switch. If the main battery voltage drops below the logic battery input level, the logic circuit and user regulated power output will be drawn from the logic battery.

### **Software**

RoboClaw can be easily configured using the Motion Studio software tool. The Windows based application enables users to quickly configure RoboClaw. The software can be used during run time to monitor and control several operational parameters. Motion studio is available from the Basicmicro.com website. It can also be found in the Downloads section of the Basicmicro website or listed under the Download tabs on the production page.

### **User Manual**

This data sheet only covers model specific information and basic wiring. To properly setup and use RoboClaw refer to the RoboClaw User Manual available for download from <http://www.basicmicro.com>.

### **Cooling**

RoboClaw will generate heat. The maximum current ratings can only be achieved and maintained with adequate heat dissipation. The motor controller should be mounted so that sufficient airflow is provided. Which will dissipate the heat away from the motor controller during operation. Some models of RoboClaw include a built-in automatic cooling fan controller, which can be used to help maintain continuous currents under extreme conditions.

### **Emergency Stop**

The motor controller should be wired using an external contactor, relay or high amperage mechanical switch to control the main power input. A second power source should be used to power the logic section in situations where the main power will be under heavy load. Voltage drops can occur from constant full load or high speed direction changes. Voltage drop can cause logic brown outs if only a main battery is used without a logic battery.

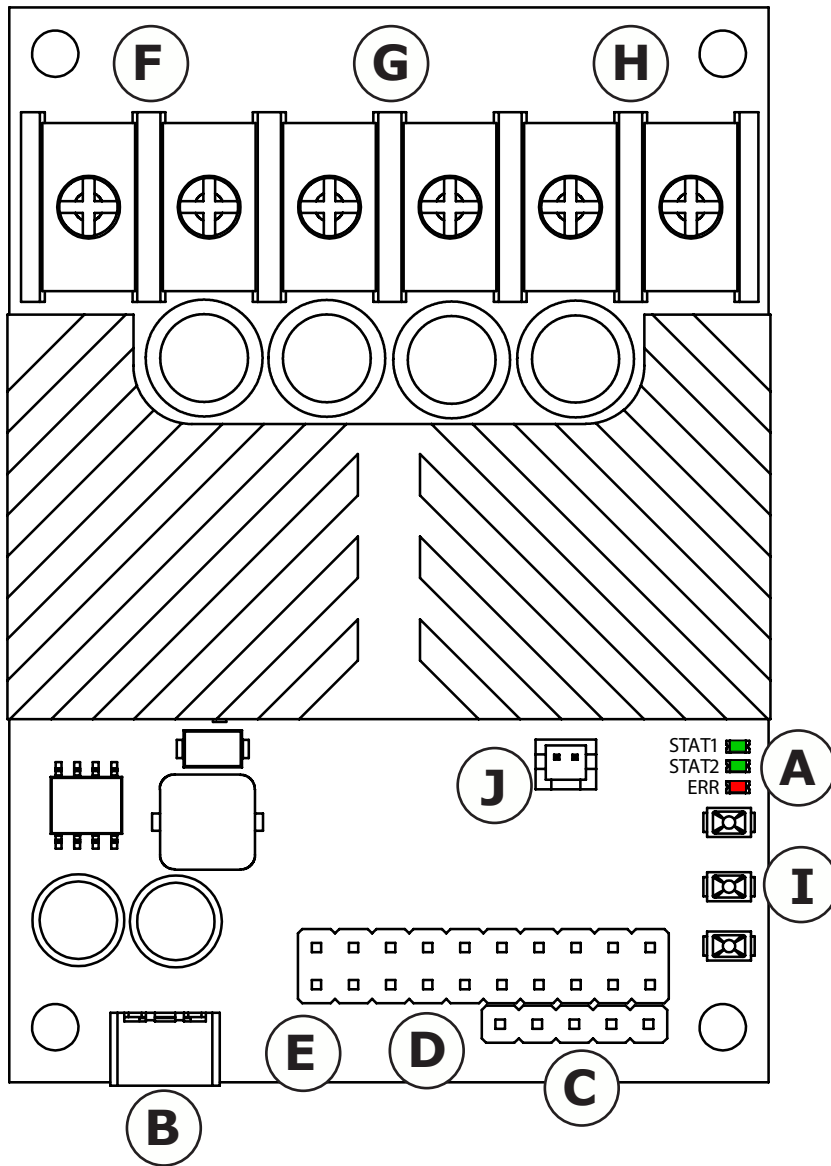
### **USB**

The motor controllers USB port should be used for configuration and debugging. The USB protocol is not designed for electrically noisy environments. The USB port will likely disconnect and not automatically recover during operation in electrically noisy environments. To recover from a dropped USB port, the motor controllers USB cable may require being unplugged and re-plugged in. The TTL serial control should be the preferred method of control in electrically noisy environments.

### **Firmware Updates**

Firmware updates will be made available to add new features or resolve any technical issue. Before using RoboClaw for the first time it is recommended to update to the latest firmware. Download and install Motion Studio. Refer to the RoboClaw User Manual or Application Notes for additional information on updating the RoboClaw firmware.

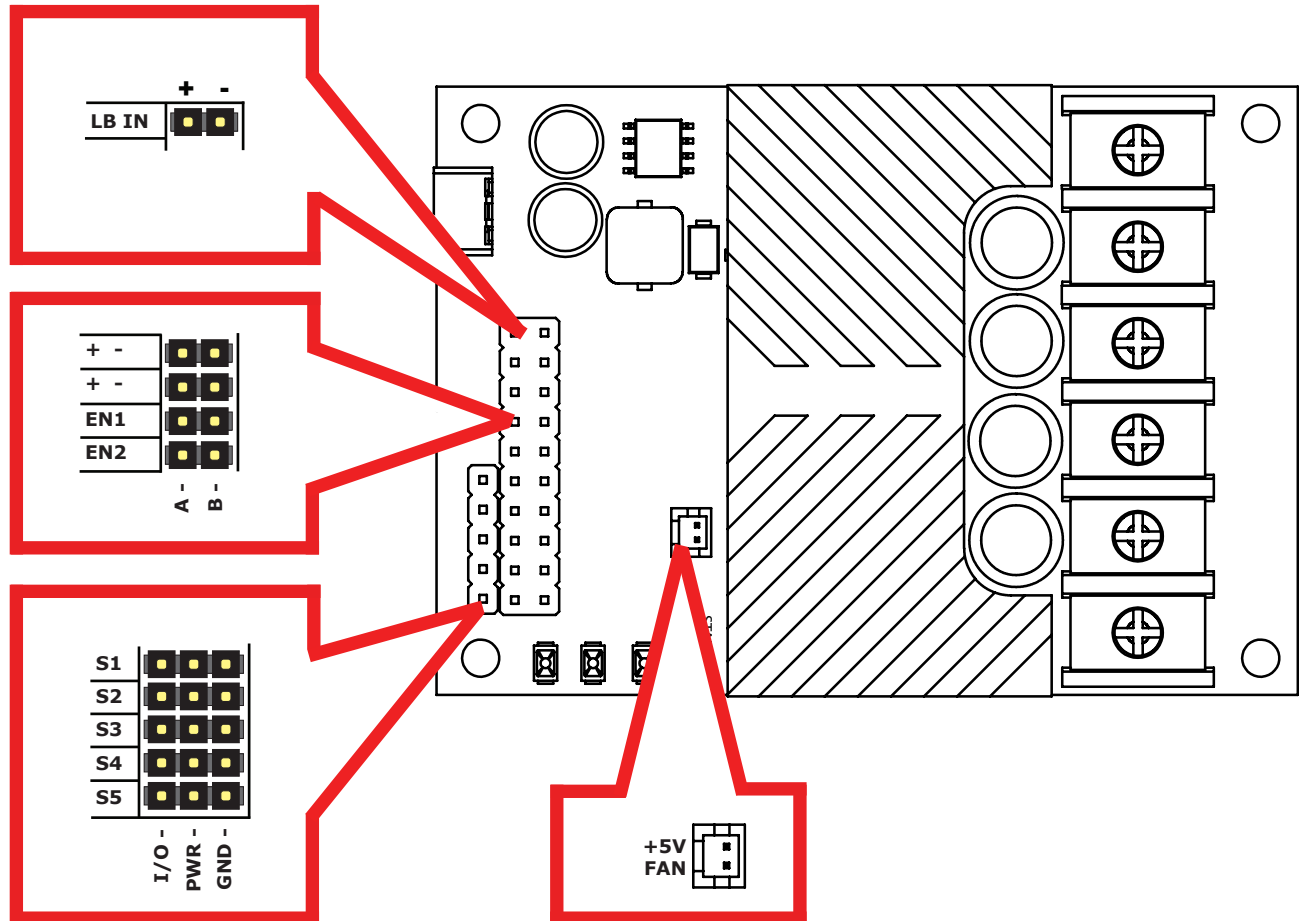
Hardware Overview:



ID	Function	DESCRIPTION
A	Status LEDs	Provides RoboClaw status information.
B	USB Port	Communicate with RoboClaw via USB.
C	Control Inputs	S1,S2,S3,S4 and S5 control inputs.
D	Encoder Inputs	Dual encoder input and power pins.
E	Logic Battery	Logic battery jumper setup and logic battery power input.
F	Motor Channel 1	Motor driver output screw terminals for channel 1.
G	Main Battery	Main battery screw terminal input.
H	Motor Channel 2	Motor driver output screw terminals for channel 2.
I	Setup Buttons	Configure RoboClaw. Can bypass and use IonMotion PC setup utility.
J	Fan Control	Automatic fan control. 5VDC Fan. On at 45°C and off at 35°C

**Control Interface**

The RoboClaw uses standard male pin headers with 0.100" (2.54mm) spacing. The pin headers are ideal for use with standard servo cables and other popular interface connectors. The table below list the pins and their respective functions. All pins are 5V tolerant and output 3.3V for compatibility with processor such as Raspberry Pi and Arduino. R/C pulse input, Analog and TTL can be generated from any microcontroller such as a Arduino or Raspberry Pi. The R/C Pulse input pins can also be driven by any standard R/C radio receiver. There are several user configurable options available. To configure RoboClaw, install Motion Studio and connect it to an available USB port.



NAME	UART TTL	ANALOG	R/C PULSE	FLIP SWITCH	E-STOP	HOME	LIMIT	V-CLAMP	Encoder
S1	RX	Motor 1	Motor 1						
S2	TX	Motor 2	Motor 2						
S3				X	X			X	
S4					X	Motor 1	Motor 1	X	
S5					X	Motor 2	Motor 2	X	
EN1									Motor 1
EN2									Motor 2
+5V									
FAN									

**Logic Battery (LB IN)**

The logic circuit of RoboClaw can be powered from a secondary battery wired to LB IN. A logic battery will prevent brownouts when the main battery is low or under heavy load. The positive (+) terminal is located at the board edge and ground (-) is the inside pin closest to the heatsink.

**Encoder Power (+ / -)**

The pins labeled + and - are the source power pins for encoders. The positive (+) is located at the board edge and supplies +5VDC. The ground (-) pin is near the heatsink. On RoboClaws with screw terminals, power for the encoders can be supplied by the 5VDC and GND on the main screw terminal.

**Encoder Inputs (1A / 1B / 2A / 2B)**

The encoders inputs are labeled EN1 and EN2. EN1 is for encoder 1 and EN2 is for encoder 2 which also correspond to motor channel 1 and motor channel 2. Quadrature encoder inputs are typically labeled 1A, 1B, 2A and 2B. Channel A of both EN1 and EN2 are located at the board edge on the pin header. Channel B pins are located near the heatsink on the pin header. Quadrature encoders are directional. When connecting encoders make sure the leading channel for the direction of rotation is connected to A. If one encoder is backwards to the other you will have one internal counter counting up and the other counting down. Use Motion Studio to determine the encoders direction relative to the motors rotation. Encoder channels A and B can be swapped in software using Motion Studio to avoid re-wiring the encoder or motor.

**Control Inputs (S1 / S2 / S3 / S4 / S5)**

S1, S2, S3, S4 and S5 are configured for standard servo style headers I/O (except on ST models), +5V and GND. S1 and S2 are the control inputs for serial, analog and RC modes. S3 can be used as a flip switch input, when in RC or Analog modes. In serial mode S3, S4 and S5 can be used as emergency stops inputs or as voltage clamping control outputs. When configured as E-Stop inputs, they are active when pulled low. All I/O have internal pull-ups to prevent accidental triggers when left floating. S4 and S5 can be configured as home switch and limit switch inputs. The pins closest to the board edge are the I/Os, center pin is the +5V and the inside pins are ground. Some RC receivers have their own supply and will conflict with the RoboClaw's 5v logic supply. It may be necessary to remove the +5V pin from the RC receivers cable in those situations.

**Cooling Fan Control**

The cooling fan control will automatically turn on and off a fan based on RoboClaws temperature. The fan will turn on when the board temperature reaches 45°C and will automatically turn off when the board temperature falls below 35°C. The fan control circuit can power a 5VDC fan at up to 230mA. A wide range of fans can be used. The CFM rating of the fan will determine how effective the fan is at cooling. A tested fan is available from DigiKey under part number: 259-1577-ND. However any fan can be used provided it meets the electrical specifications outlined above.

## Main Battery Screw Terminals

The main power input can be from 6VDC to 34VDC on a standard RoboClaw and 10.5VDC to 60VDC or 80VDC on an HV (High Voltage) RoboClaw. The connections are marked + and - on the main screw terminal. The plus (+) symbol marks the positive terminal and the negative (-) marks the negative terminal. The main battery wires should be as short as possible.



***Do not reverse main battery wires or damage will occur.***

## Disconnect

The main battery should include a quick disconnect in case of a run away situation and power needs to be cut. The switch must be rated to handle the maximum current and voltage from the battery. Total current will vary depending on the type of motors used. A common solution would be an inexpensive contactor which can be sourced from sites like Ebay. A power diode rated for approximately 2 to 10 Amps should be placed across the switch/contacter to provide a return to the battery when power is disconnected. The diode will provide the regenerative power a place to go even if the switch is open.

## Motor Screw Terminals

The motor screw terminals are marked with M1A / M1B for channel 1 and M2A / M2B for channel 2. For a typical differential drive robot the wiring of one motor should be reversed from the other. The motor and battery wires should be as short as possible. Long wires can increase the inductance and therefore increase potentially harmful voltage spikes.

## Control Modes

RoboClaw has 4 main functional control modes explained below. Each mode has several configuration options. The modes can be configured using Motion Studio or the built-in buttons. Refer to the RoboClaw User Manual for installation and setup instructions.

## RC

Using RC mode RoboClaw can be controlled from any hobby RC radio system. RC input mode also allows low powered microcontrollers such as a Basic Stamp to control RoboClaw. Servo pulse inputs are used to control the direction and speed. Very similar to how a regular servo is controlled. Encoders are supported in RC mode, refer to the RoboClaw user manual for setup instructions.

## Analog

Analog mode uses an analog signal from 0V to 2V to control the speed and direction of each motor. RoboClaw can be controlled using a potentiometer or filtered PWM from a microcontroller. Analog mode is ideal for interfacing RoboClaw with joystick positioning systems or other non microcontroller interfacing hardware. Encoders are supported in Analog mode, refer to the RoboClaw user manual for setup instructions.

## Simple Serial

In simple serial mode RoboClaw expects TTL level RS-232 serial data to control direction and speed of each motor. Simple serial is typically used to control RoboClaw from a microcontroller or PC. If using a PC, a MAX232 or an equivalent level converter circuit must be used since RoboClaw only works with TTL level inputs. Simple serial includes a slave select mode which allows multiple RoboClaws to be controlled from a signal RS-232 port (PC or microcontroller). Simple serial is a one way format, RoboClaw can only receive data. Encoders are not supported in Simple Serial mode.

## Packet Serial

In packet serial mode RoboClaw expects TTL level RS-232 serial data to control direction and speed of each motor. Packet serial is typically used to control RoboClaw from a microcontroller or PC. If using a PC a MAX232 or an equivalent level converter circuit must be used since RoboClaw only works with TTL level input. In packet serial mode each RoboClaw is assigned a unique address. There are 8 addresses available. This means up to 8 RoboClaws can be on the same serial port. Encoders are supported in Packet Serial mode, refer to the RoboClaw user manual for setup instructions.

## USB Control

USB can be used in any mode. When RoboClaw is in packet serial mode and another device, such as an Arduino, is connected commands from the USB and Arduino will be executed and can potentially override one another. However if RoboClaw is not in packet serial mode, motor movement commands will be overridden by Analog or RC pulse input. USB packet serial commands can then only be used to read status information and set configuration settings.



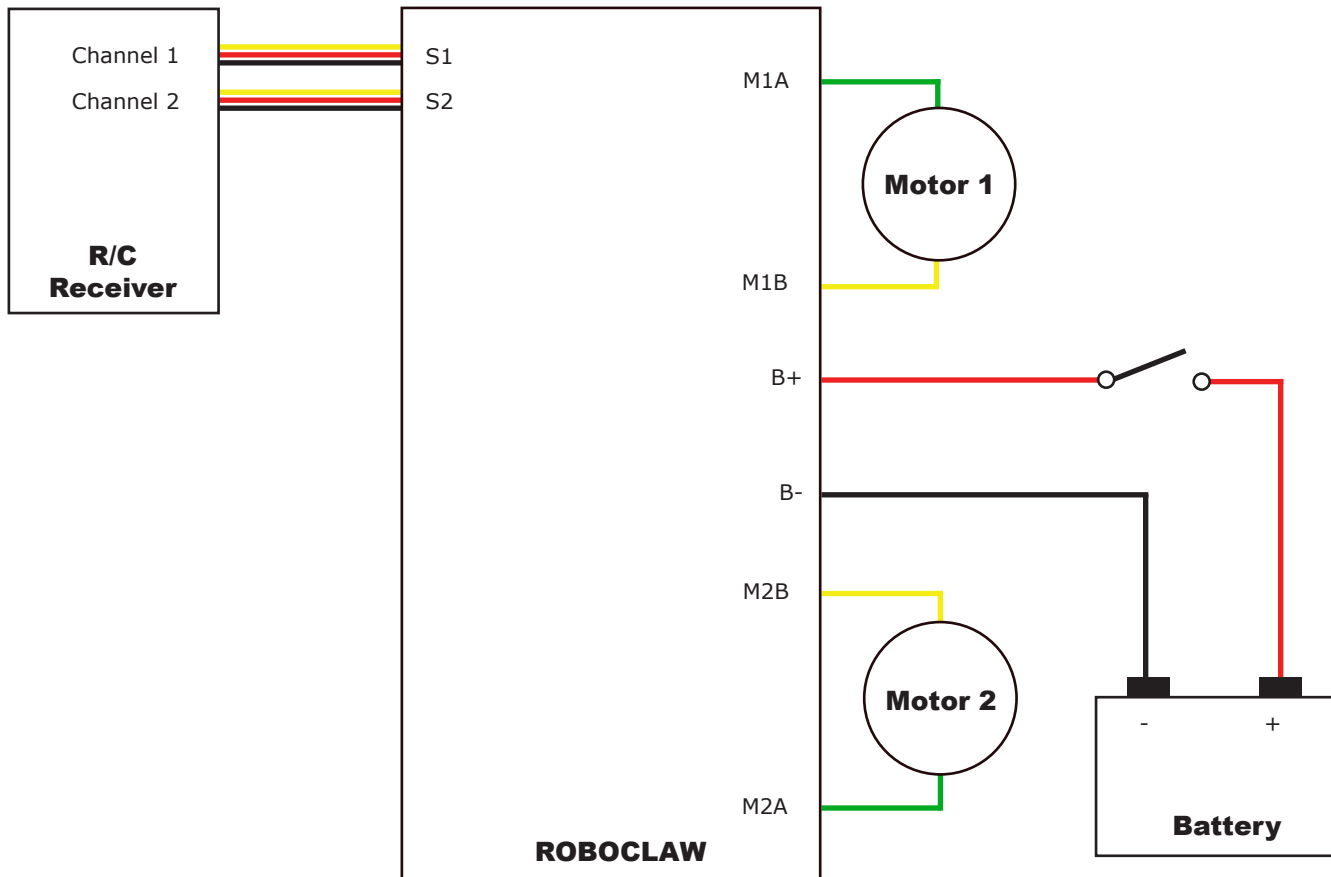
## Wiring Basics

There are several wiring configurations for RoboClaw. Each configuration will have unique wiring requirements to ensure safe and reliable operation. The diagram below illustrates a very basic wiring configuration used in a small motor system where safety concerns are minimal. This is the most basic wiring configuration possible. All uses of RoboClaw should include some kind of main battery shut off switch, even when safety concerns are minimal. Never underestimate a system with movement when an uncontrolled situation arises.

In addition, RoboClaw is a regenerative motor controller. If the motors are moved when the system is off, it could cause potential erratic behavior due to the regenerative voltages powering the system. The regenerative voltages can cause problems if a power supply is used for main power. A voltage clamping circuit is recommended to dump the excessive voltages. See the RoboClaw user manual or Application Notes for voltage clamping setup and wiring diagrams.

## R/C Mode

The below wiring diagram is very basic and for use with R/C mode. R/C mode can be used when pairing RoboClaw with a standard R/C receiver. R/C mode can also be used with a microcontroller and using servo pulses to control RoboClaw. The RoboClaw supplies power to the R/C system. If the R/C receiver used, has its own power the 5V pin on the 3 pin header must be remove otherwise it will interfere with RoboClaw's BEC.



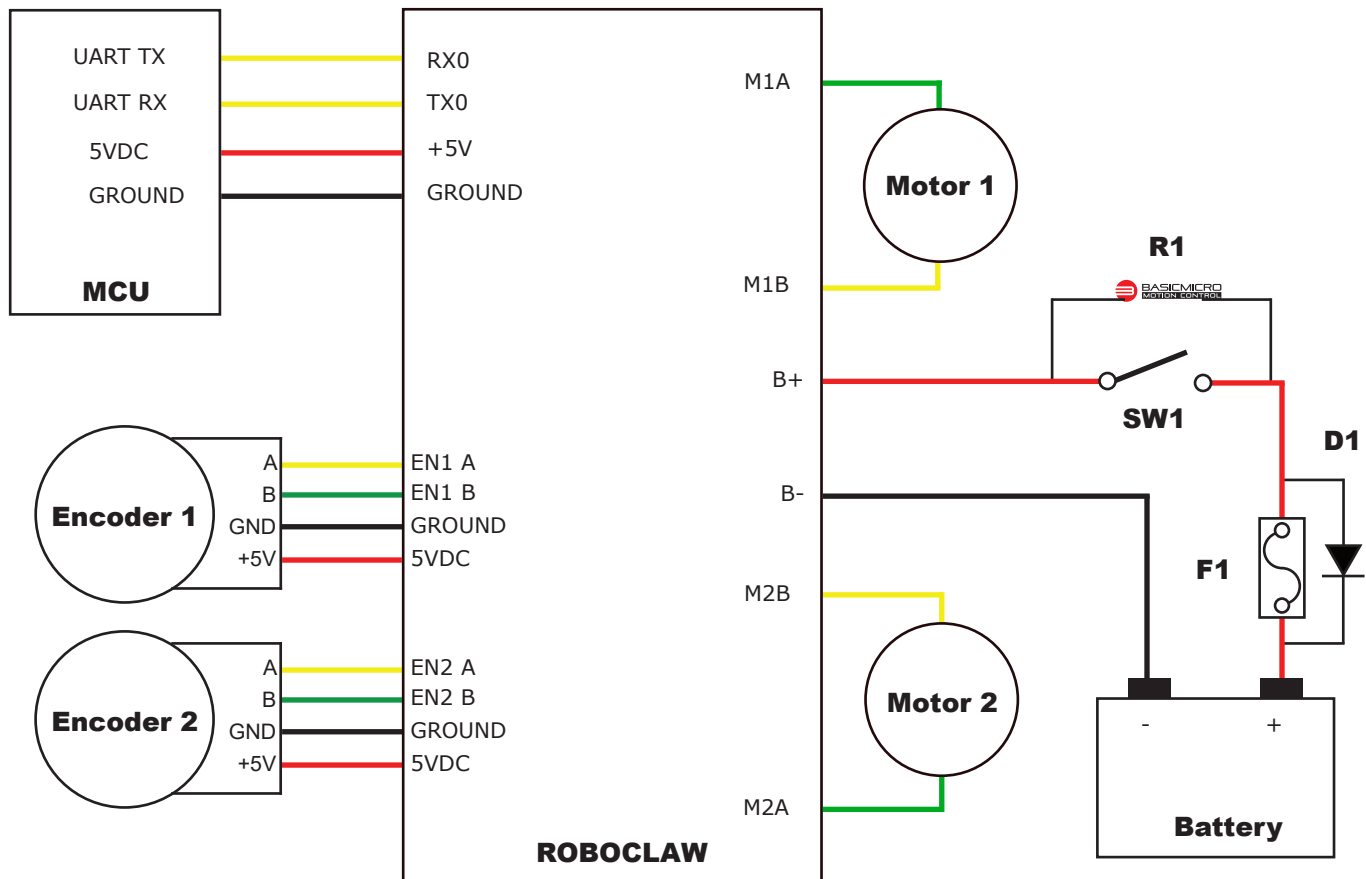
**Wiring Safety**

In all system with movement, safety is a concern. This concern is amplified when dealing with higher voltages. The wiring diagram below illustrates a properly wired system. An external main power cut off is required (SW1). The external cut off can consist of a high amperage mechanical switch or a contactor.

When the RoboClaw is switched off or a fuse is blown, a high current diode (D1) is required to create a return path to the battery for potential regenerative voltages. In addition a pre-charge resistor (R1) is required to reduce the high inrush currents to charge the on board capacitors. A pre-charge resistor (R1) should be around 1K, 1/2Watt for a 60VDC motor controller which will give a pre-charge time of about 15 seconds. A lower resistances can be used with lower voltages to decrease the pre-charge time.

**Closed Loop Mode**

A wide range of sensors are supported for closed loop operation. RoboClaw supports dual quadrature encoders (up to 9.8 million PPS), absolute encoders, potentiometers and hall effect sensors. The wiring diagram below is an example of closed loop mode using quadrature encoders. Quadrature encoders are directional. RoboClaw’s internal counters will increment for clockwise rotation (CW) and decrement for counter clockwise rotation (CCW). When wiring encoders A and B channels it is important they are wired to match the direction of the motor. If the encoder is wired in reverse it can cause a run away condition. All motor and encoder combinations will need to be tuned (see the RoboClaw user manual).

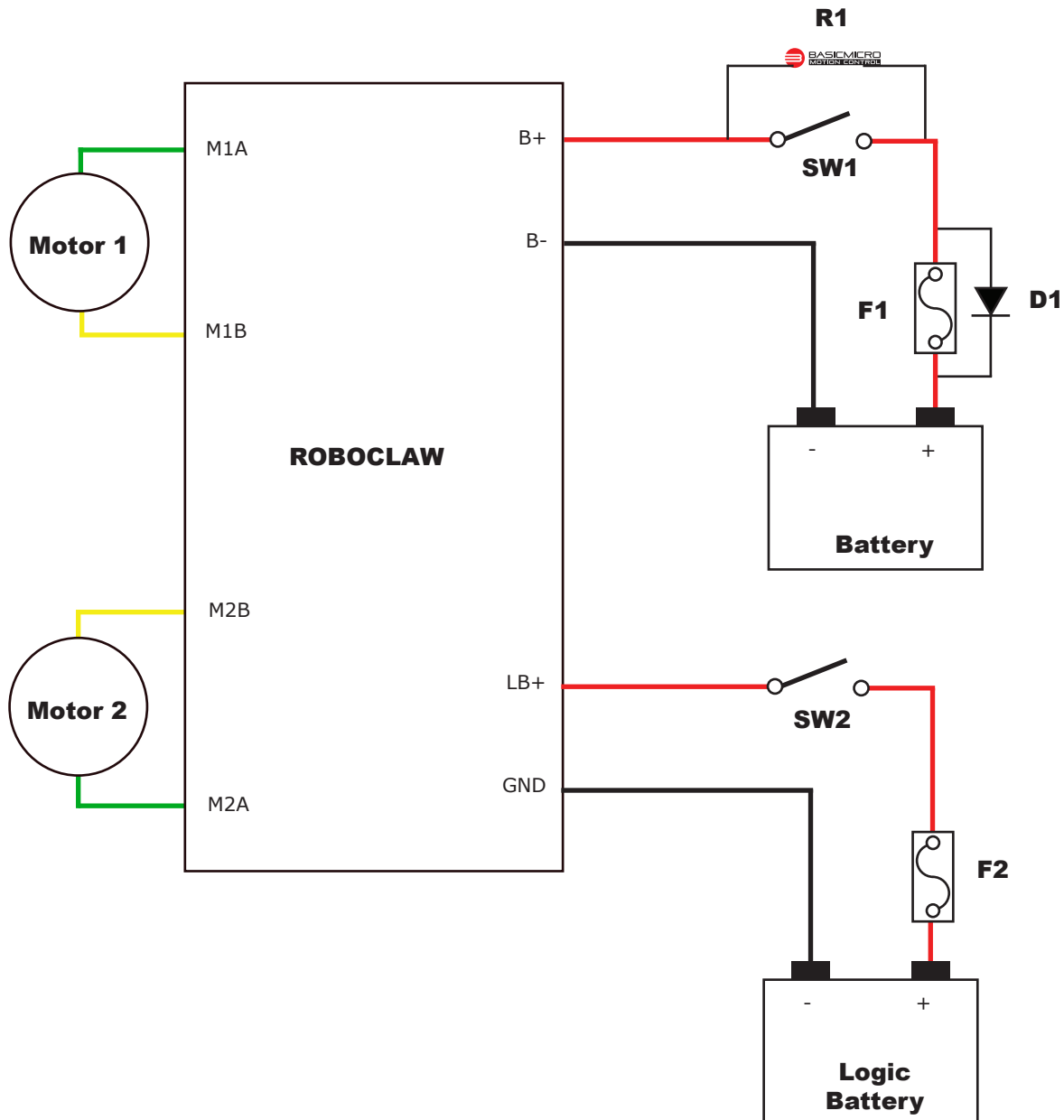


**Logic Battery**

An optional logic battery is supported. Under heavy loads the main power can suffer voltage drops, causing potential logic brown outs which may result in uncontrolled behavior. A separate power source for the motor controllers logic circuit, will remedy potential problems from main power voltage drops. The logic battery maximum input voltage is 34VDC with a minimum input voltage of 6VDC. The 5V regulated user output is supplied by the secondary logic battery if supplied. The mAh of the logic battery should be determined based on the load of attached devices powered by the regulated 5V user output.

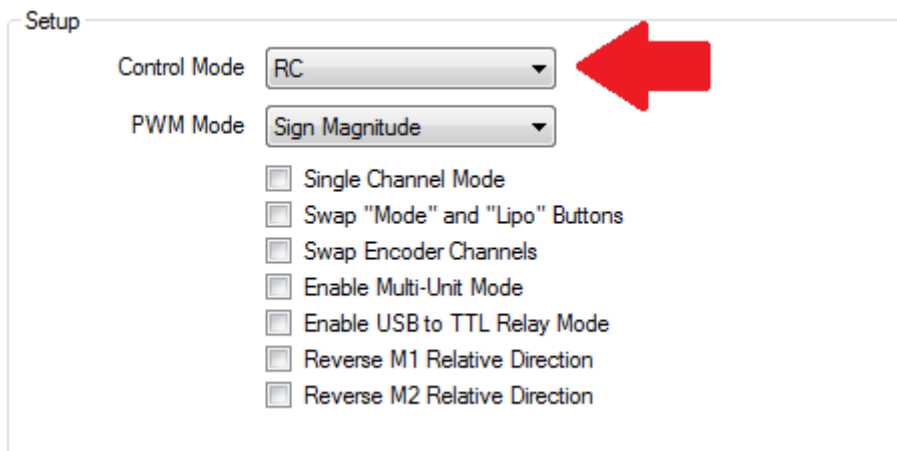
**Logic Battery Jumper**

The configuration below utilizes a logic battery. Older models of RoboClaw have a logic battery jumper. On models where the LB-MB header is present the jumper must be removed when using a separate logic battery. If the header for LB-MB is not present, then the RoboClaw will automatically set the logic battery power source.

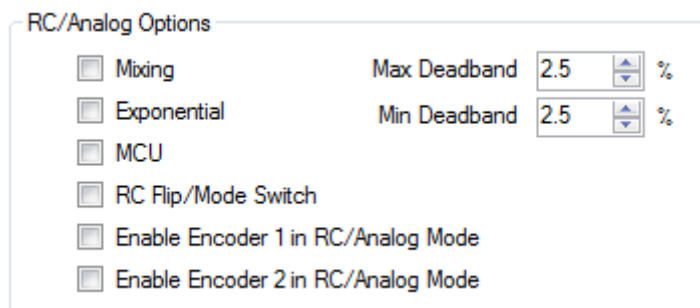


**RC Control Setup**

1. Wire the RoboClaw Solo, power supply, motor and RC hardware as show in the RC wiring diagram in this datasheet.
2. Additionally, it is recommened to wire safety hardware as show in the Safety Wiring section of this datasheet.
2. Connect a micro USB cable between the RoboClaw Solo and a computer with Motion Studio installed. The USB connection does not power the RoboClaw, a power supply must be connected. Open Motion Studio and click "Connect Selected Unit" to connect to the motor controller.
3. Click on PWM Settings on the left-hand side of the application. Use the slider for motor 1 in the Control section of the application. The motor should move in the forward direction when the slider is moved up and in the reverse direction when moved down. If the motor does not move the proper direction power down the controller and reverse the motor wiring.
3. Next, click on General Setting on the left-hand side of the application. This is where the RC settings are set.
4. In the panel labeled Setup set the Control Mode to RC.



5. Next, RC options can be set in the panel labeled RC/Analog Options.



6. If the ability to reverse controls for the motor channel with an additional RC channel is desired check the RC Flip/Mode Switch checkbox. The RC channel that will be used as a flip switch should be wired to the S3 header on the Solo.
7. If dampening for the RC input is desired check the checkbox labeled Exponential. This setting makes control of the motor channel less sensitive.
8. If a custom amount of deadband should be applied to the RC input set the boxes labeled Max and Min Deadband to the desired amount. Deadband allows the stick of the RC transmitter to travel a small amount without a response from the motor channel.
9. If the motor being used has an encoder and has been wired to the Solo check the checkbox labeled Enable Encoder 1 in RC/Analog Mode. This setting will provide velocity control for the motor which helps it maintain a constant speed regardless of load.

10. To complete the setup for RC mode click Device in the menu at the top of the application and then click on Write Settings. The settings are now saved to the Solo.

11. Disconnect the Solo from Motion Studio by clicking Disconnect Selected Unit in the upper left-hand side of the application. The USB connection can then be removed from the Solo.

12. The robot or other system can now be operated remotely with the RC transmitter.

## Additional Control Modes and Documentation

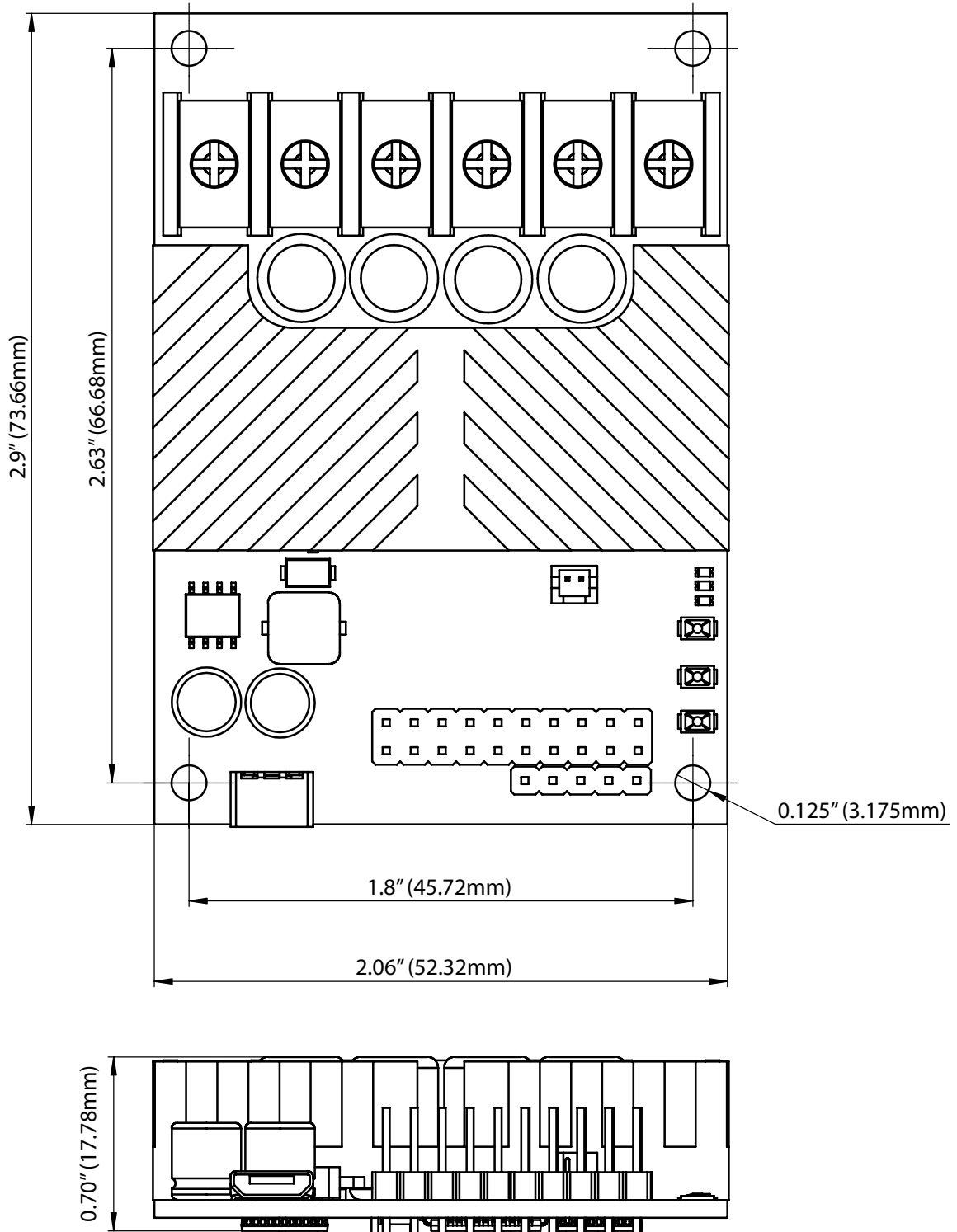
The RoboClaw Solo also features other control modes such as packet serial, simple serial and analog. Documentation for these mode's wiring and software configuration can be found in the User Manual as well as our Application Notes.

The User Manual can be found here: <https://www.basicmicro.com/downloads>  
Application Notes can be found on our website: <https://resources.basicmicro.com/>

Mechanical Specifications

Characteristic	Model	Min	Typ	Max	Rating
Weight	2X15A		2.2 (62)		Oz (g)

Dimensions



## Electrical Specifications

Characteristic	Min	Typ	Max	Rating
Main Battery	6		34	VDC
Logic Battery	6	12	34	VDC
Maximum External Current Draw (BEC)			3	A
Motor Current Per Channel		15 <sup>(2)</sup>	30 <sup>(1,2)</sup>	A
Motor Current Bridged		30 <sup>(2)</sup>	60 <sup>(1,2)</sup>	
On Resistance		4.3		mOhm
Logic Circuit Current Draw		30mA		mA
Input Impedance		100		Ω
Input	0		5	VDC
Input Low	-0.3		0.8	VDC
Input High	2		5	VDC
I/O Output Voltage	0		3.3	VDC
Digital and Analog Input Voltage			5	VDC
Analog Useful Range	0		2	VDC
Analog Resolution		1		mV
Pulse Width	1		2	mS
Encoder Counters		32		Bits
Encoder Frequency			9,800,000	PPS
RS232 Baud Rate (Note 3)			460,800	Bits/s
RS232 Time Out (Note 3)	10			ms
Temperature Range	-40	40	100	°C
Temperature Protection Range	85		100	°C
Humidity Range			100 (4)	%

**Notes:**

1. Peak current is automatically reduced to the typical current limit as temperature approaches 85°C.
2. Current is limited by maximum temperature. Starting at 85°C, the current limit is reduced on a slope with a maximum temperature of 100°C, which will reduce the current to 0 amps. Current ratings are based on ambient temperature of 25°C.
3. RS232 format is 8Bit, No Parity and 1 Stop bit.
4. Condensing humidity will damage the motor controller.

## Warranty

Basicmicro warranties its products against defects in material and workmanship for a period of 1 year. If a defect is discovered, Basicmicro will, at our sole discretion, repair, replace, or refund the purchase price of the product in question. Contact us at [sales@basicmicro.com](mailto:sales@basicmicro.com). No returns will be accepted without the proper authorization.

## Copyrights and Trademarks

Copyright© 2020 by Basicmicro, Inc. All rights reserved. All referenced trademarks mentioned are registered trademarks of their respective holders.

## Liability Statement

By purchasing and using Basicmicro's products you acknowledge and agree to the following: Basicmicro has no liabilities (and, to the Basicmicro's knowledge, there is no basis for any present or future action against the company giving rise to any liability) arising out of any injury to individuals or property as a result of ownership, possession or use of any product designed or sold by Basicmicro. No product from Basicmicro should be used in any medical devices and/or medical situations. No product should be used in any life support situations.

## Contacts

Email: [sales@basicmicro.com](mailto:sales@basicmicro.com)  
Tech support: [support@basicmicro.com](mailto:support@basicmicro.com)  
Web: <http://www.basicmicro.com>

## Discussion List

A web based discussion board is maintained at <http://www.basicmicro.com>

## Technical Support

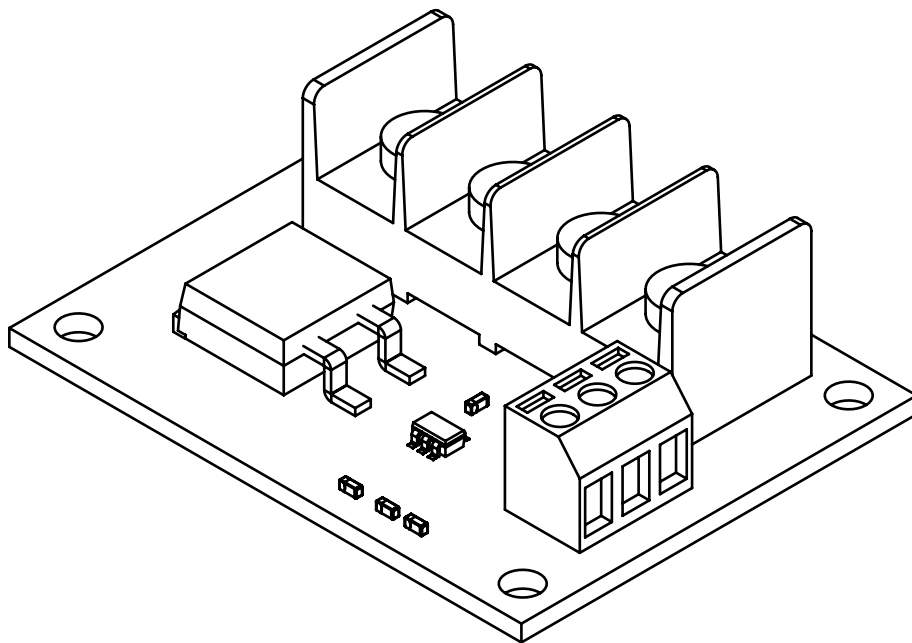
Technical support is available by sending an email to [support@basicmicro.com](mailto:support@basicmicro.com), by opening a support ticket on the Basicmicro website or by calling 800-535-9161 during normal operating hours. All email will be answered within 48 hours.





# BASICMICRO

## MOTION CONTROL

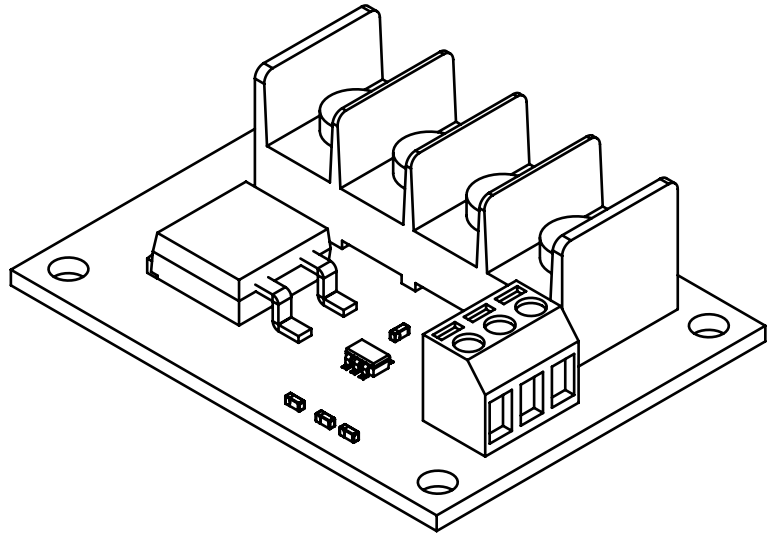


**VClamp - Voltage Clamp**

**Data Sheet Version 1.1**

### Feature Overview:

- 60 Amps Peak Clamping
- 60VDC Peak Operation
- TTL Control
- Air and Conduction Cooling
- 3.3v Compliant Control Outputs
- 5v Tolerant Control Inputs
- Status LED



### Device Overview

The VClamp adapter will allow a switching power supply to be used with either RoboClaw or MCP motor controllers. During coasting or slow down, excessive voltages can be generated at the power input. The excess voltages can cause certain power supplies to shut down due to over voltage. The VClamp can dissipate the excess energy through a braking resistor.

The VClamp can be controlled by RoboClaw or MCP and is activated when the maximum set voltage of the RoboClaw or MCP is exceeded. VClamp is also ideal in battery systems where regenerative voltages need to be controlled to prevent over charging.

### Control Interface

VClamp is controlled by a simple ON/OFF interface. A low signal on its control input will activate it, turning on the MOSFET. When the VClamp is active, the excess voltage will be sent through a connected power resistor, which will then be bled off as heat.

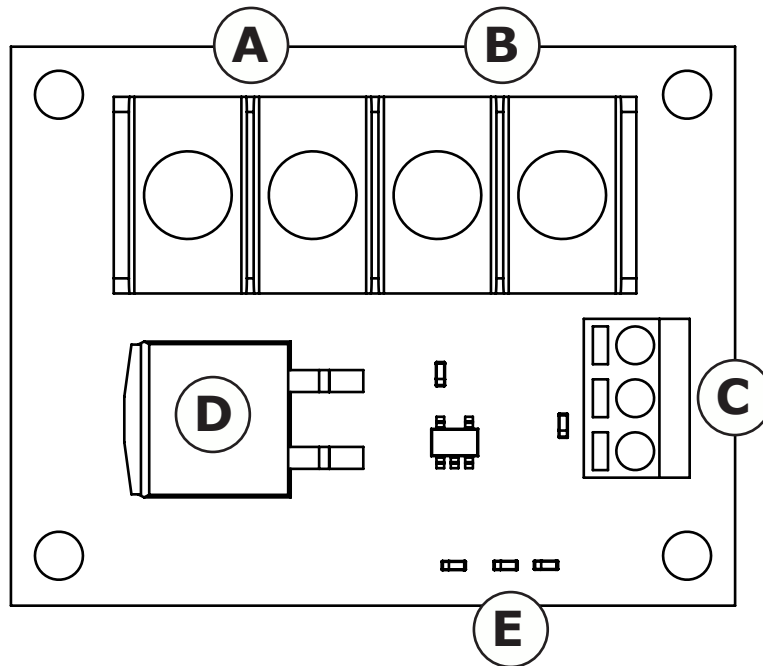
### Limits

VClamp can dissipate up to 60 amps at 60VDC. This is limited by the type of braking resistor used and if the resistor has a heat sink.

### Function

During a regeneration period, when the maximum set voltage is exceeded by 1V, RoboClaw or MCP will automatically turn on the VClamp. It will stay active until the voltage settles to the maximum voltage that was previously set.

Hardware Overview:

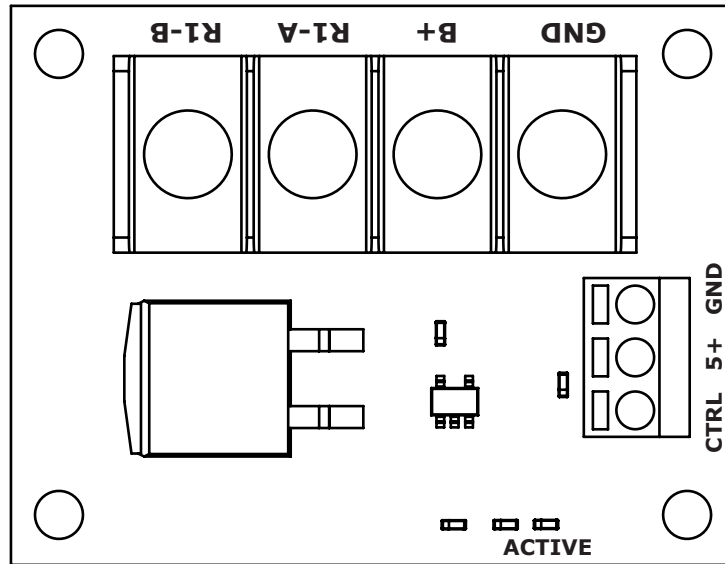


ID	Function	Description
A	Resistor Terminals	Braking resistor used for power dissipation.
B	Main Power	Main power connection. Positive and GND connections.
C	Control Inputs	Control input.
D	MOSFET	Part Number: BUK965R8-100E,118
E	Status LED	Status LED indicates when clamping MOSFET is active.

## Functional Overview

The R1-A and R1-B screws terminals are used to connect the braking resistor. The B+ and GND connections on the main screw terminal are tied to the main power source. The control interface screw terminals provide a connection for control signal (CTRL), signal ground and 5VDC to power VClamp. CTRL is an active LOW and is internally pulled-up with a 10K resistor to keep it off when no signal is applied. A low input on CTRL will turn the MOSFET (BUK965R8-100E,118) on.

When VClamp is active and the MOSFET is on the led marked ACTIVE will be lit. Any excess voltage will be bled off as heat via the attached braking resistor to ground.



Name	Function
GND	Main supply voltage ground terminal.
B+	Main supply voltage positive terminal.
R1-A	Dissipation braking resistor terminal.
R1-B	Dissipation braking resistor terminal.
CTRL	Active LOW to turn on VClamp.
5+	5VDC required to operate VClamp.
GND	Signal ground.
ACTIVE	Status LED. On when VClamp is active.

**Resistor Selection**

The resistor needs to be sized based on the running voltage and potential current that will need to be dissipated. Below is an example chart of potential resistor values and wattages. The resistors can be sourced from suppliers such as DigiKey. The excess energy will be turned into heat. The recommended resistors below have an aluminum shell for mounting to help dissipate the excess heat.

Designator	Part	Resistance	12VDC	24VDC	36VDC	48VDC	60VDC
R1	KAL50FB1R00	1 Ohm	12A	24A	36A	48A	60A
R1	KAL50FB2R00	2 Ohm	6A	12A	18A	24A	30A
R1	KAL50FB3R00	3 Ohm	X	6A	9A	12A	15A
R1	KAL50FB5R00	5 Ohm	X	X	7.2A	9.6A	12A

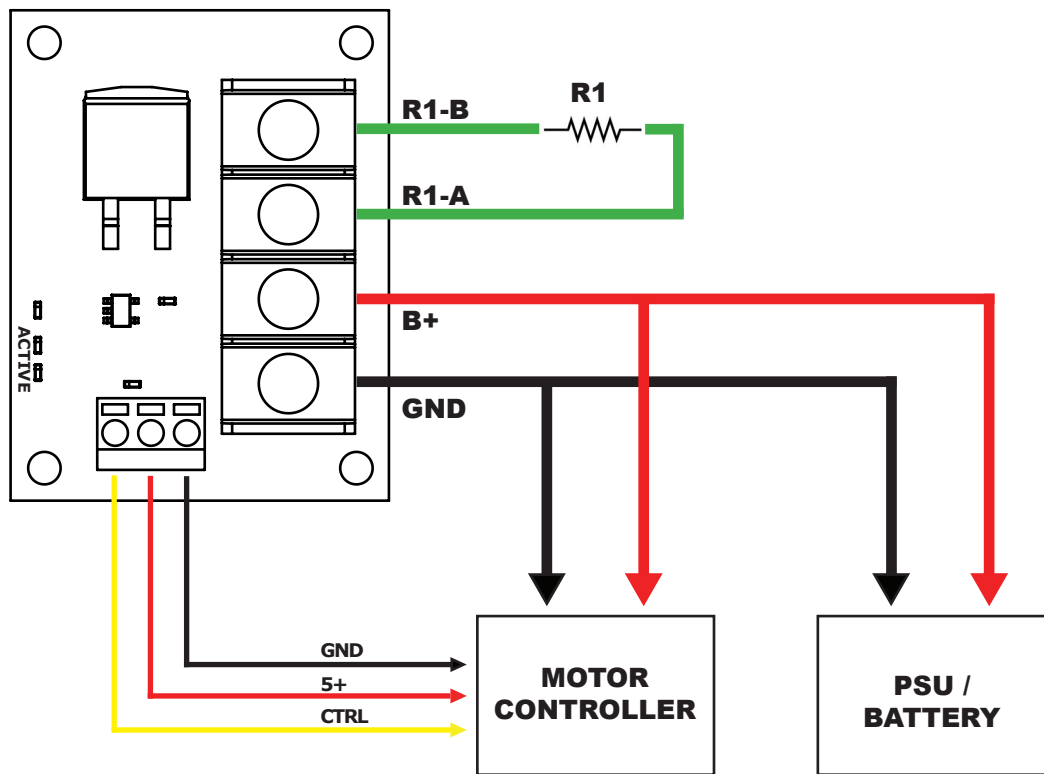
**RoboClaw and MCP Software Setup**

Each of the auxiliary pins on RoboClaw or MCP have a drop down option in Motion Studio. One of these options will be VClamp. Set the desired auxiliary pin to VClamp and select the invert option for the pin. Then set the battery maximum voltage to at least 1 to 2 volts above. Refer to the VClamp application note for more information.

**Wiring**

The wiring schematic below illustrates how to properly wire the VClamp. R1 is the braking resistor. R1 will turn the excess energy into heat. The VClamp can be controlled directly by the motor controller or separate control system such as an external microcontroller. B+ and GND from the main screw terminal connect directly to the motor controller power source.

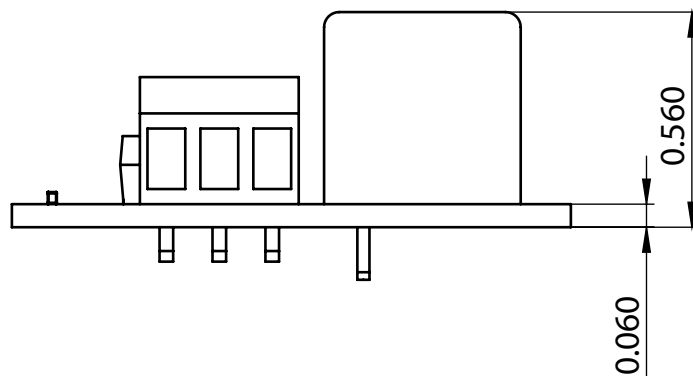
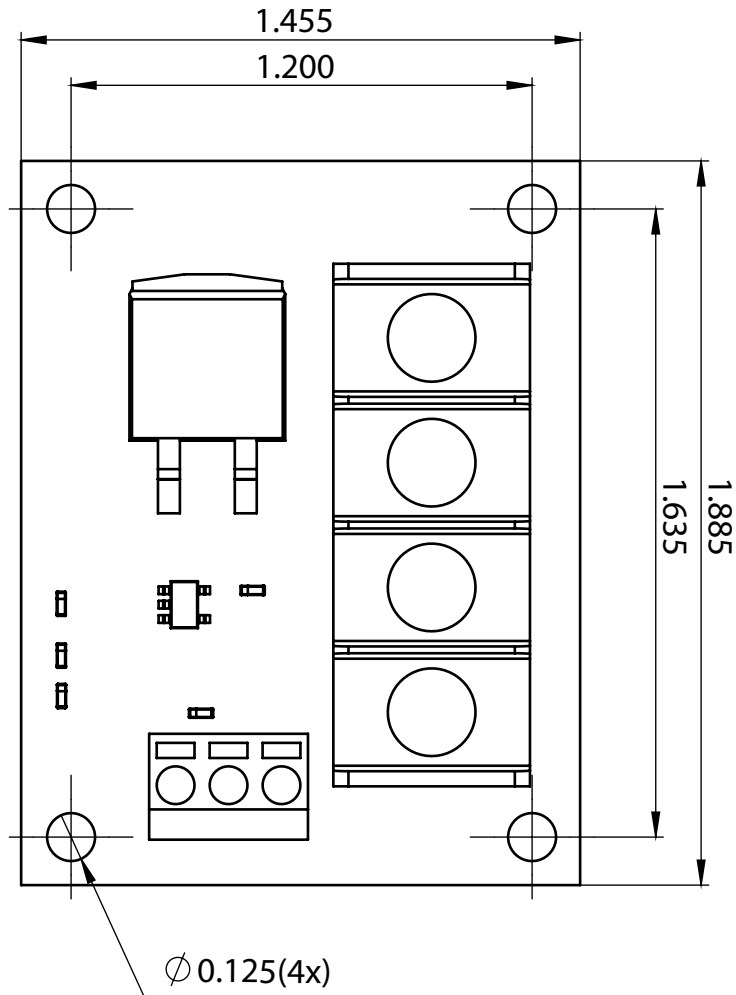
The below schematic outlines the basic setup for VClamp. It does not take into account proper safety such as a fuse or contactor in the system. Refer to the motor controller’s data sheet, manual and application notes.



Mechanical Specifications

Characteristic	Model	Min	Typ	Max	Rating
Weight	VClamp		0.8 (22)		Oz (g)

Dimensions



**Electrical Specifications**

Characteristic	Min	Typ	Max	Rating
Battery	6		60	VDC
Current			60	A
On Resistance		5.8		mOhm
Logic Circuit Current Draw		10mA		mA
Input Low	-0.3		0.8	VDC
Input High	2		5	VDC
Temperature Range	-55		150	°C
Humidity Range			100 (1)	%

**Notes:**

1. Condensing humidity will damage the device.

**Warranty**

Basicmicro warrants its products against defects in material and workmanship for a period of 1 year. If a defect is discovered, Basicmicro will, at our sole discretion, repair, replace, or refund the purchase price of the product in question. Contact us at [sales@basicmicro.com](mailto:sales@basicmicro.com). No returns will be accepted without the proper authorization.

**Copyrights and Trademarks**

Copyright© 2020 by Basicmicro, Inc. All rights reserved. All referenced trademarks mentioned are registered trademarks of their respective holders.

**Liability Statement**

By purchasing and using Basicmicro's products you acknowledge and agree to the following: Basicmicro has no liabilities (and, to the Basicmicro's knowledge, there is no basis for any present or future action against the company giving rise to any liability) arising out of any injury to individuals or property as a result of ownership, possession or use of any product designed or sold by Basicmicro. No product from Basicmicro should be used in any medical devices and/or medical situations. No product should be used in any life support situations.

**Contacts**

Email: [sales@basicmicro.com](mailto:sales@basicmicro.com)

Tech support: [support@basicmicro.com](mailto:support@basicmicro.com)

Web: <http://www.basicmicro.com>

**Discussion List**

A web based discussion board is maintained at <http://www.basicmicro.com>

**Technical Support**

Technical support is available by sending an email to [support@basicmicro.com](mailto:support@basicmicro.com), by opening a support ticket on the Basicmicro website or by calling 800-535-9161 during normal operating hours. All email will be answered within 48 hours.



**C Preproject report**

# FORPROSJEKT - RAPPORT

## FOR BACHELOROPPGAVE

TITTEL:

**CHAOS: Aktiv muskelstyring til biomekanisk testing**

KANDIDATNUMMER(E):

**509362 & 487537**

DATO:	EMNEKODE:	EMNE:	DOKUMENT TILGANG:
<b>20.01.2020</b>	<b>IE303612</b>	<b>Bacheloroppgave</b>	- Åpen
STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:	
<b>AUTOMATISERINGSTEKNIKK</b>	10/1	- Ikke i bruk -	

OPPDRAKSGIVER(E)/VEILEDER(E):

Andreas Dalen  
Øystein Bjelland  
Aleksander Larsen Skrede

OPPGAVE/SAMMENDRAG:

Ålesund Biomekaniske Lab er forskningslab der ortopedi og ingeniørkunst møtes. Den medisinske forskningen går ut på å teste nye ortopediske operasjonsmetoder på kadaverprøver. For å teste resultat av nye operasjonsmetoder, f.eks. leddstabilitet, benyttes en Kuka KR-6 industriell robot med 6DOF lastscelle. Det neste steget er å også inkludere aktiv muskelstyring i stabilitetstester. Dette kan f.eks. gjøres med elektriske lineær-aktuatorer med tilhørende styringsystem. Oppgaven er å utvikle et slikt system for aktuelt ledd.

## INNHOOLD

<b>INNHOOLD</b> .....	<b>2</b>
<b>1 INNLEDNING</b> .....	<b>4</b>
<b>2 BEGREPER</b> .....	<b>4</b>
<b>3 PROSJEKTORGANISASJON</b> .....	<b>4</b>
3.1 PROSJEKTGRUPPE.....	4
3.1.1 Oppgaver for prosjektgruppen – organisering.....	4
3.1.2 Oppgaver for prosjektleder.....	5
3.1.3 Oppgaver for sekretær.....	5
3.1.4 Oppgaver for øvrige medlemmer.....	5
3.2 STYRINGSGRUPPE (VEILEDER OG KONTAKTPERSON OPPDRAGSGIVER).....	5
<b>4 AVTALER</b> .....	<b>5</b>
4.1 AVTALE MED OPPDRAGSGIVER.....	5
4.2 ARBEIDSSTED OG RESSURSER.....	5
4.3 GRUPPENORMER – SAMARBEIDSREGLER – HOLDNINGER.....	5
<b>5 PROSJEKTBESKRIVELSE</b> .....	<b>6</b>
5.1 PROBLEMSTILLING - MÅLSETTING - HENSIKT.....	6
5.2 KRAV TIL LØSNING ELLER PROSJEKTRESULTAT – SPESIFIKASJON.....	6
5.3 PLANLAGT FRAMGANGSMÅTE(R) FOR UTVIKLINGSARBEIDET – METODE(R).....	7
5.4 INFORMASJONSINNSAMLING – UTFØRT OG PLANLAGT.....	7
5.5 VURDERING – ANALYSE AV RISIKO.....	7
5.6 HOVEDAKTIVITETER I VIDERE ARBEID.....	7
5.6.1 Ansvarsfordeling og ressursestimater.....	7
5.7 FRAMDRIFTSPLAN – STYRING AV PROSJEKTET.....	8
5.7.1 Hovedplan.....	8
5.7.1.1 Styringshjelpemidler.....	8
5.7.2 Utviklingshjelpemidler.....	9
5.7.3 Intern kontroll – evaluering.....	9
5.8 BESLUTNINGER – BESLUTNINGSPROSESS.....	9
<b>6 DOKUMENTASJON</b> .....	<b>9</b>
6.1 RAPPORTER OG TEKNISKE DOKUMENTER.....	9
<b>7 PLANLAGTE MØTER OG RAPPORTER</b> .....	<b>9</b>
7.1 MØTER.....	9
7.1.1 Møter med styringsgruppen.....	9
7.1.2 Prosjektmøter.....	9
7.2 PERIODISKE RAPPORTER.....	9
7.2.1 Framdriftsrapporter (inkl. milepæl).....	9
<b>8 PLANLAGT AVVIKSBEHANDLING</b> .....	<b>10</b>
<b>9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING</b> .....	<b>10</b>
<b>10 REFERANSER</b> .....	<b>10</b>
<b>VEDLEGG</b> .....	<b>10</b>



## 1 INNLEDNING

Denne oppgaven ble valgt hovedsakelig på grunn av en interesse for kombinasjonen av elektroniske og biomekaniske systemer. Oppgaven består av en rekke varierte utfordringer og problemstillinger. Disse fremstår som svært relevante med tanke på den opparbeidede kompetansen fra tidligere fag/emner.

Det biomekaniske robotlaboratoriet ved Ålesund sjukehus er unikt i Skandinavia, og det finnes svært få liknende laboratorier i Europa. Ortopedene ved sjukehuset samarbeider med ingeniører fra NTNU for å få det til.

Oppgaven går ut på å utvikle en rigg som skal brukes til å simulere aktiv muskelstyring av et ledd. For å gjøre dette er det planlagt å bruke lineære aktuatorer til å aktivere senene i et ledd. Dette simulerer måten en muskel virker i et levende vesen. I første omgang skal riggen brukes til et ankelledd, men det er et underliggende ønske om at riggen skal bli så modulær som mulig og kunne brukes til flere ledd.

## 2 BEGREPER

- Definisjon av sentrale begreper i prosjektet.
- Aktiv muskelstyring
  - o Styring av ledd/kroppsdel gjennom aktivering av muskler
- Biomekanikk
  - o Mekanikken til biologi, altså de fysiske egenskapene til menneskekroppen
- Ortopedi
  - o Er en del av kirurgien som omhandler medfødte misdannelser, deformiteter, sykdommer og skader i bevegelsesapparatet
- Leddstabilitet
  - o Leddets evne til å håndtere forventede stress situasjoner
- Elektriske lineære aktuatorer
  - o Elektriske svitsjer som påfører strøm på hva enn som er tilkoblet på lineært vis
- Ledd simulasjon
  - o Simulering av ledd og bevegelsene til leddet
- Kontrollsystem
  - o Kontrollapparat som styrer en eller flere prosesser som sammen kalles ett system

## 3 PROSJEKTORGANISASJON

### 3.1 Prosjektgruppe

Studentnummer(e)
509362
487537

Tabell: Studentnummer(e) for alle i gruppen som leverer oppgaven for bedømmelse i faget ID 302906

#### 3.1.1 Oppgaver for prosjektgruppen – organisering

Erlend Lillevik- Prosjektleder

Andreas Nonslid Håvardsen- Sekretær

### 3.1.2 Oppgaver for prosjektleder

Lederansvar er innkalling til møte, plan for gjennomgang i møte og øvrig planlegging for gruppekontakt med veiledere eller andre ressurspersoner.

### 3.1.3 Oppgaver for sekretær

Sekretærens ansvar er notering av tanker og forsøk i løpet av møter, samt notering av fremgang. Sekretæren må passe på at det som blir gjennomgått på møtet er mulig å lese og forstå både for de som hadde mulighet til å delta, samt de som ikke deltok.

### 3.1.4 Oppgaver for øvrige medlemmer

Alle medlemmer har ansvar for å møte på møter for å holde flyt med gruppen, og å sende inn grunn for mangel av deltakelse samt sin fremgang og arbeid på prosjektet uavhengig av oppmøte. Alle medlemmer har også ansvar for å yte sitt beste for å oppnå sine delegerte oppgaver. I tilfelle der de ikke klarer å utføre en oppgave, er de også ansvarlige for å si ifra til gruppen om dette, slik at gruppen sammen kan løse problemet sammen på best mulig vis.

## 3.2 *Styringsgruppe (veileder og kontaktperson oppdragsgiver)*

Oppdragsgiver for prosjektet er **Andreas Dalen**

Veiledere er **Øystein Bjelland & Aleksander Larsen Skrede**

Kontaktperson er **Øystein Bjelland**

## 4 AVTALER

### 4.1 *Avtale med oppdragsgiver*

### 4.2 *Arbeidssted og ressurser*

- Laben ved Ålesund biomekaniske lab er til disposisjon mens medlemmer av styringsgruppen er til stede. Laben ved NTNU er til disposisjon døgnet rundt. \*
- Ålesund biomekaniske lab har det meste av ressurser man trenger for å gjennomføre oppgaven. Om det skulle være noe som mangler så har NTNU også mye ressurser tilgjengelig.
- Det er god tilgang til ressurspersoner da Ålesund biomekaniske lab vanligvis er bemannet under normale arbeidstider. Skulle det være nødvendig å ha tilgang til ressurser utenfor disse tidene blir det kommunikasjon via E-post eller Microsoft Teams.
- Det er ingen konfidensialitet tilknyttet denne oppgaven. Men skal det tas hensyn på grunn av oppgavens nære tilknytning til sykehuset.
- Det er avtalt møter med styringsgruppen annenhver uke, mandag klokken 13:00.

\* Forbeholdt at Covid-19 restriksjonene ikke forverres.

### 4.3 *Gruppenormer – samarbeidsregler – holdninger*

Når det jobbes i grupper, er man avhengig av god kommunikasjon. Dette innebærer å informere de andre i gruppen om problemer eller utfordringer som kan påvirke gruppen sin fremgang. Man skal også

respektet andre sin tid med å møte til avtalt tid og gjennomføre sine arbeidsoppgaver innen de fastsatte fristene. Om dette ikke er gjennomførbart skal man snarest informere resten av gruppen slik man kan komme frem til en løsning. Man skal alltid høre og respektere andre sin mening og eventuelle uenigheter skal løses ved med saklig argumentasjon.

Uavhengig av yrke og stilling er det visse holdninger man bør inneha. Det skal vises respekt for kolleger og samarbeidspartnere uavhengig av kjønn, livssituasjon, kultur, livssyn, legning og etnisk opprinnelse. Man skal også opprettholde sin faglige integritet, vedkjenne seg sitt faglige ansvar og utføre sitt arbeid i henhold til anerkjente kvalitetsnormer.

Som ingeniør er man i stor grad utsatt for å kunne havne i situasjoner med etiske problemstillinger. Derfor bør man reflektere over sine egne mål og etiske prinsipper. Slike etiske refleksjoner hjelper deg å tenke systematisk gjennom problemstillinger og vurdere ulike løsnings og handlingsalternativer.

Som ingeniør har man også et samfunnsansvar, derfor skal man vise respekt for samspillet mellom teknologi og menneskelige verdier, samt bidra til åpenhet om konsekvenser av teknologiske løsninger.

## 5 PROSJEKTBESKRIVELSE

### 5.1 *Problemstilling - målsetting - hensikt*

Målet med denne oppgaven er å lage en rigg som skal styre et ledd. I utgangspunktet vinkles riggen mot styring av ankel, men det er som mål å gjøre riggen mer fleksibel til bruk for andre ledd.

Det første delmålet er å lage en fysisk rigg, senere skal ledd-styringen reguleres for å kunne utføre naturlige bevegelser. Oppgavemålet er derfor bestående av to hoveddeler. Disse to delene er den fysiske riggen og ledd-simulering gjennom system kontroll. Gruppen skal først lage riggen, deretter gjennom systemgjenkjenning finne en approksimert modell for systemet som deretter kan kontrolleres for simulering av naturlige bevegelser. Selve systemgjenkjenninga kan ikke bli gjort mot en generell løsning for alle ledd siden de oppfører seg for ulikt, og vil derfor heller bli siktet mot kun ankelen, men metoden vil være brukbar for andre ledd også.

I tillegg må gruppen velge ut hva riggen skal være laget av og lage den. Det økonomiske og praktiske valget, samt fysiske sammensettingen av riggen blir også essensielle deler av oppgaven.

Som resultat vil gruppen da ha som mål å klare å ha laga en rigg som er fleksibel angående hvilket ledd det skal bli koblet opp mot, og som i hvert fall i utgangspunktet også er i stand til å kontrollere en hvert fall noenlunde naturlig leddbevegelse av en ankel.

Effekten gruppen har som mål å ha er at det er en god start til ledd-styring som studenter i ettertid kan jobbe videre på for å gjøre mer presise ledd-simulasjoner og faktisk ende opp med å kunne bli brukt av ortopedier eller andre som vil benytte seg av ledd-styring.

I løpet av prosessen er målet å være fleksible for muligheten til at arbeidssteder blir låst ned på grunn av Covid-19 og være klare for å måtte fortsette oppgaven til en god løsning via for eksempel simulasjon, og underveis klare å holde ett godt tempo likevel.

### 5.2 *Krav til løsning eller prosjektresultat – spesifikasjon*

Gruppen kommer til å følge NTNU sine krav til kvalitet ved gjennomføring av Bacheloroppgave.

Prosjektet har i utgangspunktet få spesifikke krav til spesifikasjoner. Dette åpner for mer frihet til hvordan gruppen velger å gjennomføre oppgaven. Men denne økte friheten legger også mer ansvar på planleggingen av oppgaven. På grunn av prosjektets tilknytning til medisinsk forskning så er det visse

krav til spesifikasjoner som bør møtes. Det mest fremtredende er at riggen skal kunne brukes på kadaver (menneskelige kroppsdeler). Dette stiller krav til renslighet og desinfisering, det bør derfor være et mål å lage en løsning som tilrettelegger for dette.

Prosjektet har gode økonomiske rammer og vil i utgangspunktet ha mulighet til å anskaffe de komponenter som er nødvendige. Ved prosjektets fullføring siktes det på å levere en funksjonell prototype som utfyller de kravene vi fastsetter under planleggingen. Det skal også legges ved dokumentasjon og bruksanvisning, dette for å tilrettelegge bruk og videreutvikling av riggen.

### **5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)**

Prosjektstyring betyr å bruke en prosess eller metodologi til å fullføre en midlertidig oppgave med et definert mål. Til denne oppgaven har vi planlagt å bruke metoden “kritisk bane” (på engelsk: “Critical Path Method”)

Metoden “kritisk bane” er en type trinnvis prosjekttidslinje. Den kommer best til sin rett når minst én av oppgavene som kreves for å fullføre et prosjekt, avhenger av en annen oppgave.

Når man bruker kritisk bane, lister man først opp oppgavene, og deretter fastslår det hvilke oppgaver som avhenger av hverandre samt hvilke oppgaver som kan bli forsinket uten at hele prosjektet stopper. Med denne metoden er man hovedsakelig på jakt etter den raskeste måten å fullføre et prosjekt på med minst mulig nedetid mellom oppgavene. Den kronologiske sekvenseringen av oppgaver blir dermed svært viktig, ettersom det er den estimerte varigheten for hver oppgave.

### **5.4 Informasjonsinnsamling – utført og planlagt**

Informasjonsinnsamling for dette prosjektet vil være en kombinasjon av faglitteratur, nettbaserte databaser og kontakt med fagpersonell.

Den informasjonen som omhandler biomekanikk og anatomi vil hovedsakelig bli hentet fra medisinsk faglitteratur. Denne typen litteratur er det godt utvalg av ved den Ålesund biomekaniske lab. Det vil også bli brukt rapporter fra andre liknende prosjekter som er utført ved laboratorier i utlandet. Nettbaserte ressurser som Oria og Google Scholar vil også bli brukt. Gruppen har også god tilgang til fagpersonell, både innen ortopedi og det ingeniørfaglige.

### **5.5 Vurdering – analyse av risiko**

Prosjektet har stor mulighet for å kunne realiseres innenfor den gitte rammen. Design og konstruksjon av riggen bør gjennomføres så raskt som mulig for å forsikre at man har nok tid for testing og justering. Om fremgangen her blir for treg kan dette føre til at den reguleringstekniske delen av oppgaven blir lammet.

For å lykkes med denne oppgaven er det særlig viktig med god koordinasjon internt i gruppen. Man er også avhengig av god kommunikasjon med oppdragsgiver og resurspersoner. Dette forsikrer at fremgangen holder seg jevn og innenfor prosjektets rammer. Skulle dette feile kan det føre til misforståelser som gjør at prosjektet ikke møter kravene som er satt.

Den pågående COVID-19 pandemien er også et stort risikoelement, særlig på grunn av labens nære tilknytning til sykehuset. Skulle den nåværende smittesituasjonen forverre seg kan man bli tvunget til å måtte gjøre alt arbeid på prosjektet “off site”, og i verste fall måtte gjøre mye av arbeidet via simulasjoner.

### **5.6 Hovedaktiviteter i videre arbeid**

#### **5.6.1 Ansvarsfordeling og ressursestimat**

Nr	Hovedaktivitet	Ansvar	Kostnad	Tid/omfang
		ANH = Andreas Nonslid Håvardsen	-	EL = Erlend Lillevik



(uker)			
A1	Design av rigg	ANH & EL	4.0
A2	Valg av deler	EL	3.0
A3	Konstruksjon av rigg	EL	8.0
A4	Oppkobling	EL	2.0
A5	Systemgjenkjenning	ANH	4.0
A6	Systemkontroll	ANH	6.0
A7	Simulasjon	ANH	3.0

Den totale kostnaden ligger på 5000, og tiden krevd er 37 uker som blir overlappa til å bli gjort i løpet av semesteret.

## 5.7 Framdriftsplan – styring av prosjektet

### 5.7.1 Hovedplan

Hovedtrekka i oppgaven er som tidligere nevnt design og gjennomføring av rigg og system kontroll for simulering av naturlig bevegelse av ledd.

A1	–	Design av rigg	–	Plantegninger av rigg, og utkast for hvordan den endelige riggen blir
A2	–	Valg av deler	–	Valg av material med tanke på økonomisk effekt og fremtidig bærekraft
A3	–	Konstruksjon av rigg	–	Konstruksjon av rigg etter plantegninger og/eller eventuelle oppdateringer av planene
A4	–	Oppkobling	–	Oppkobling av styringsmekanismen og test- ankel som skal kalibreres og systemet skal kontrollere
A5	–	Systemgjenkjenning	–	Sending av ulike signal til system for å se hvilke responser systemet gir fra seg i respons til de forskjellige, og deretter ut ifra dette gjenkjenne systemets funksjoner
A6	–	Systemkontroll	–	Kontroll av det estimerte systemet ut ifra punkt A5. Dette innebærer å lage en regulator som gjør at posisjonene i leddet kan bli styrt
A7	–	Simulasjon	–	Tilkobling av selve styringsmekanismen til testleddet og systemregulatoren der det prøves å utføre en så naturlig ledd bevegelse som mulig

#### 5.7.1.1 Styringshjelpemidler

Gruppen bruker Microsoft Teams for organisering med veiledere, og til skylagring av innsamlede data eller andre diverse prosjektressurser. Dette er derfor plassen oppdateringene for prosjektframgangen vil bli organisert.

Angående selve planleggingen vil gruppen siden det kun er to gruppemedlem ta alle valg etter diskusjoner der det blir nådd en enighet om beste framgangsmåte, men veiledere og andre ressurspersoner vil også bli kontaktet om viktige valg til å kunne påvirke. Dette vil sannsynligvis være gjennom Zoom videomøter spesielt med tanke på Covid-19, men kan potensielt være gjennom fysiske møter eller andre løsninger i tilfelle Zoom er utilgjengelig.

Fremgangsoversikten vil være utført i form av store og generelle milepæler som gruppen setter opp at skal bli oppnådd i løpet av prosjektet.

Innkjøp av komponenter og deler til riggen vil være gjennom veiledere eller den labbansvarlige ved NTNU Ålesund.

### **5.7.2 Utviklingshjelpemidler**

Gruppen har god tilgang til utviklingshjelpemidler. På Ålesund biomekaniske lab er det verksted med nødvendig utstyr. Det er også tilgang til 3D-printer. På NTNU er det tilgang til alt av verktøy man kan få bruk for.

### **5.7.3 Intern kontroll – evaluering**

Gruppen har planlagt møter annenhver mandag gjennom hele prosjektperioden. På disse møtene blir det gjennomgang av fremdrift og fremtidige planer. Når prosjektet har nådd delmål kan det være aktuelt med ekstraordinære møter for evaluering.

### **5.8 Beslutninger – beslutningsprosess**

Siden gruppen bare har 2 medlemmer, er man avhengig av en ryddig beslutningsprosess. Derfor skal alle sentrale beslutninger og presiseringer tas i felleskap. Om det skulle oppstå en situasjon der gruppen ikke blir enige, bør det tas opp i plenum med styringsgruppen for å få tilførsel av andre meninger. Alle valg og beslutninger gruppen tar skal noteres av sekretær.

## **6 DOKUMENTASJON**

### **6.1 Rapporter og tekniske dokumenter**

Gruppen skal utarbeide framdriftsrapporter som skal presenteres annenhver uke under møtet med styringsgruppen. Disse framdriftsrapportene skal inneholde all relevant informasjon for fremdriften på prosjektet. I tillegg skal det også utarbeides delmålsrapporter når visse delmål er nådd. Alle rapporter blir fortløpende levert for godkjenning fra styringsgruppen. Alle dokumenter og rapporter skal lagres på Microsoft Teams eller i OneDrive for å forsikre mot tap av dokumenter.

## **7 PLANLAGTE MØTER OG RAPPORTER**

### **7.1 Møter**

#### **7.1.1 Møter med styringsgruppen**

- Det er planlagt møter med styringsgruppen annenhver mandag klokken 13:00, gjennomgang av framdriftsrapport.

#### **7.1.2 Prosjekt møter**

- Hver mandag klokken 12:00, gjennomgang av forrige ukes fremdrift og planlegging av neste ukes fremdriftsplan.

### **7.2 Periodiske rapporter**

- Leveres fortløpende.

#### **7.2.1 Framdriftsrapporter (inkl. milepæl)**

- Framdriftsrapporter, annenhver mandag.
- Delmålsrapport, ved fullføring av milepæl.

## **8 PLANLAGT AVVIKSBEHANDLING**

Om prosjektet ikke skulle gå som planlagt skal det meldes fra til styringsgruppen, dette kan tas opp fortløpende eller på møter. Sammen med styringsgruppen skal man prøve finne en løsning på utfordringene. Her er det også viktig at alle på gruppen tar på seg ansvar og jobber best mulig for å minimere eventuelle problemer

## **9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING**

For å gjennomføre denne oppgaven etter den nåværende plan trenger gruppen tilgang til Ålesund biomekaniske lab og NTNU. Av programvare vil det være bruk for program for 3D CAD og program for koblingsskjema. Det meste av nødvendig programvare er tilgjengelig fra NTNU.

Det kreves også en del utstyr til konstrueringen av riggen. For selve konstruksjonen er det planlagt å bruke Bosch Rexroth aluminiumsprofiler. Det trengs også aktuatorer m/kontrollere for den aktive styringen. I den sammenhengen trenger man også strekksensorer for å måle lasten på vaierne/senene.

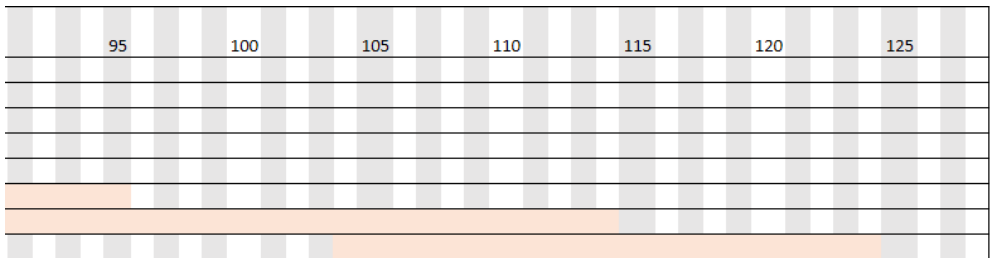
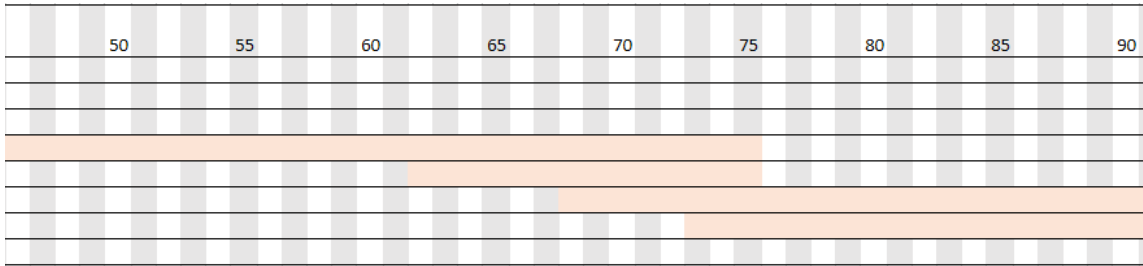
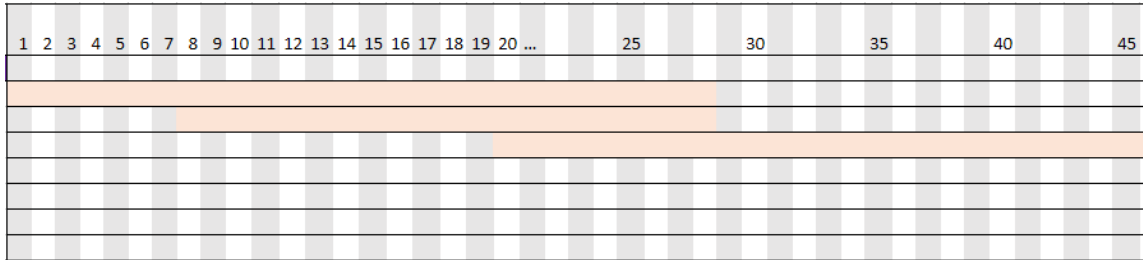
## **10 REFERANSER**

### **VEDLEGG**

Vedlegg 1 – Prosjektplanlegging Gantt Diagram

## **D Gantt diagram**

Prosjektplanlegging	Ver 1.0				
	Planlagt start	Planlagt varighet	Faktisk start	Faktisk varighet	Prosent fullført
Aktivitet					
Design av rigg	1	28			0%
Valg av deler	8	21			0%
Konstruksjon av rigg	20	56			0%
Oppkobling	56	14			0%
Systemgjennkjennning	68	28			0%
Systemkontroll	73	42			0%
Simulasjon	104	21			0%



## **E Meeting reports**

# Veiledningsmøte 8. Februar

Generelle notat underveis:

- RS - Forskjellige prisklasser
- Ledd tåler 2-3x egen vekt
- Veiersystem/modell
- Lastceller integrert er veldig dyrt
- Lastceller 200/300 kg
- Excel - Pris - Merke/Spesifikt element - Anders - NTNU Rammeverk
- Aktuatorer må ha innfesting i Rexroth
- Skrive i rapport at vi vurderte andre muligheter
  - Hydraulikk altfor tungvint for eksempel...
- Festene i ankelen, ta utgangspunkt i 4 sener
  - Er en drøss av sener, velger 4:
    - Tibialis posterior
    - Achilles
    - Fibularis brevis
    - Tibialis anterior

Biomekanikk bøker

Anatomi

Summary e-poster

High level

## Veiledningsmøte 22. Februar

Generelle notat underveis:

- Vurderer å endre typen rigg-fundament / struktur
- Ekstra sterk strømforsyning gir mulighet for ekstra aktuatorer i framtida
- Kontroll -> Arduino, Kommunikasjon -> Raspberry pi
- Innkjøpsliste OK
- Legge inn billigere alternativ og ta bort strukturdelene
- Grey box
- Kinematisk modell - fjærer dempere osv
- Begynne på styring med arduino og raspberry pi
  - Programmeringsspråk og arbeidsfordeling mellom de
- Kinematikk solvers er tilgjengelig
- Plassere tilknyttingspunkt på foten
- Øystein snakke med Andreas Dahlen om innkjøp
- Profilene kommer fort, kan utsettes for videre utvikling av design i forkant



# Veiledningsmøte 22. Mars

Generelle notater:

Velge billigkrok

Treng ikkje løfte noe

Biltema, Clas Ohlson etc

Bruke ordentlige materialer for selve riggen

*"Gjort skikkelig når det først blir gjort"*

Hvordan feste aktuatorene

Bolter - Enkel løsning

Låne deler for testing til våre kommer

Bruke Raspberry til komplisert styring

Enkel styring / IO modul

Python i Raspberry pi

Rapportskriving

Overleaf

30-50 sider standard

Innhold er viktigst

Bakgrunn, formål, grunn til formål, metoder, teknologier, beskrivelser av løsninger (Rigg & Simulering), diskusjon bra dårlig kunne blitt gjort annerledes, konklusjon (final remarks / ettertanker)

Skrive på ingeniørnivå / ikkje forklare grunnleggende konsept, men heller nevnt brukte verktøy

Teorikrav / Forventa teori av leser, ikkje forklare for eksempel matriseregning, men kanskje forklare kinematikk, bruk av bibliotek for å utføre løsning på kinematikk krever forklaring på løsning av kinematisk oppgave, men ikkje implementeringa.

Ta med grunnleggende enkel anatomi og hvorfor modellen som blir brukt blir brukt, kinematikken blir naturlig her

Utforming av rigg er også en naturlig del

Utgangspunkt for rigg og hvorfor endte den som den gjorde

Kinematikk kan bli forklart med ord, men burde ha masse figurer som kan vise mest mulig

Innhenting av figurer er lov, bare vis til kilder

Kan bruke photoshop etc, og deretter skrive figure adapted from "source"

Diskusjonsdel kan inneholde hvorfor metoder ble brukt som for eksempel at grunnen var lockdown

Ingen subjektivitet frem til diskusjonsdel, alt objektivt

Treng ikke skrive oppgaveteksten siden vi har vært tro til den

Frank H. Netter - Illustrasjoner til ankel - Gi credit

Dele zotero bibliotek og direkte input til Overleaf

# Veiledningsmøte 6. April

Generelle notater:

Venter på polulu spenningsvern - kan bruke batteri

Koder i matlab - konvertere til C

Venter på fot - ukjent ventetid

10-12 april åpnes lab - kan bli forlenget nedstenging

Knagger til rapport

# Veiledningsmøte 19. April

## Generelle notater:

- Matlab - Visu & simu
  - Kraft vektorer
  - Punkter på sidene av grønne linje
- Epost til Andreas Labb for å hente ankel
- Polulu - Har komt
- Masse-fjær-damper system
  - 1 frihetsgrad
  - Styre fra oppå bak og sidene helt forenklet
- Fiskesene klar til testing av modell på fot

## Rapport

- Introduksjon
    - Kontekst sykehus etc
  - Teori
    - Anatomi
  - Materialer metode
    - Deler løsningsmetode
  - Simulering
  - Fysisk prototype
    - Hardware
    - Software
  - Diskusjon / Drøfting
    - Ka som ble tatt med og vekk
- 
- Innkjøp av resterende deler / Selve rigg skjelett
    - Lav sendingstid for å rekke bachelorfrist

## Veiledningsmøte 3. Mai

Generelle notater:

Simu/Visu:

- Legge på bilder
  - Legge bokser om bilder ikkje går

Prototyp:

- Setter opp på campus
  - Legger inn limits i drag

Rapport:

- Kinematikk
  - Grunnleggende
- Koblingskjema
- Feedbackskjema
- Teoridel
  - Kan skrive mye ekstra som heller kan skippes

# Veiledningsmøte 10. Mai

Generelle notat:

Resultat del med forklaring av matematikk

Chapter 6 headline i appendix

Si visu ferdig og fokusere på rapport og prototyp

Legge data collection i background

Vanlig bruk av reference

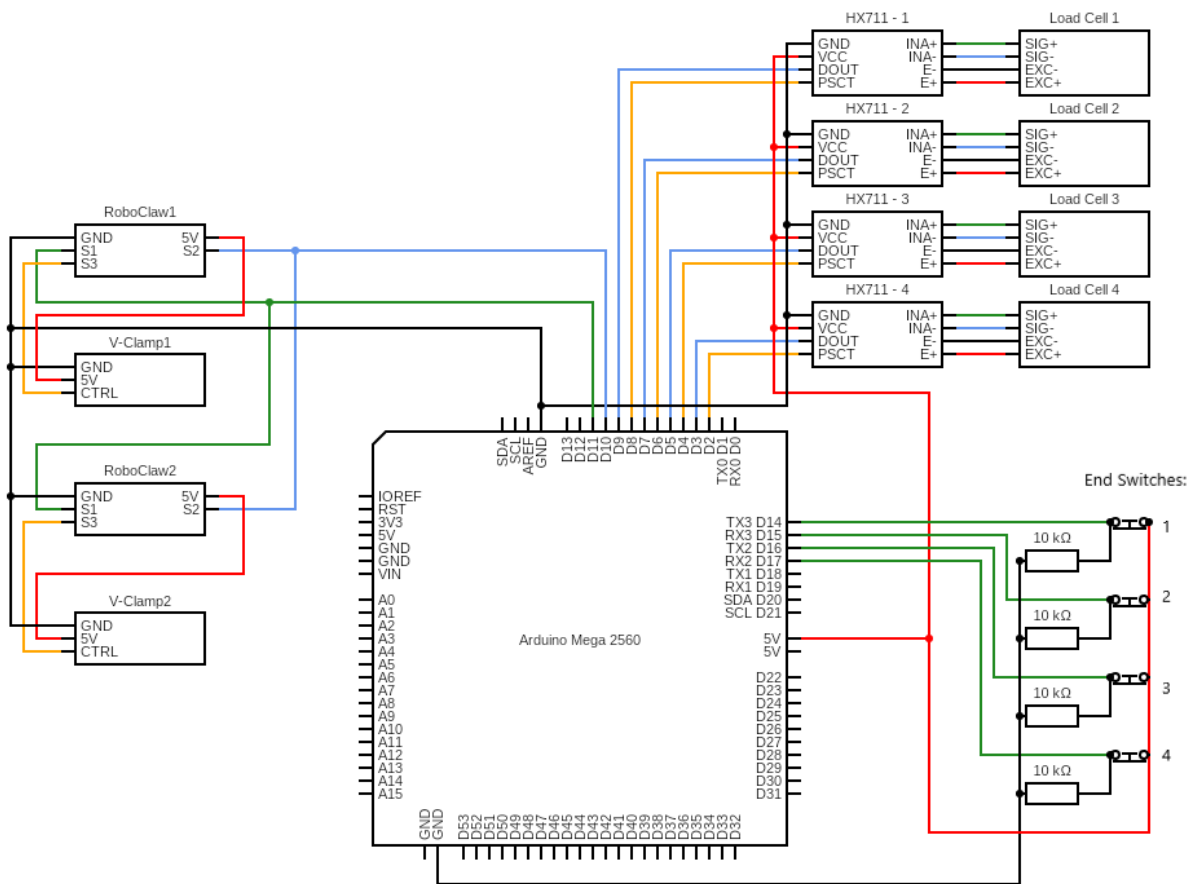
Reference per uttalelse i høyere grad

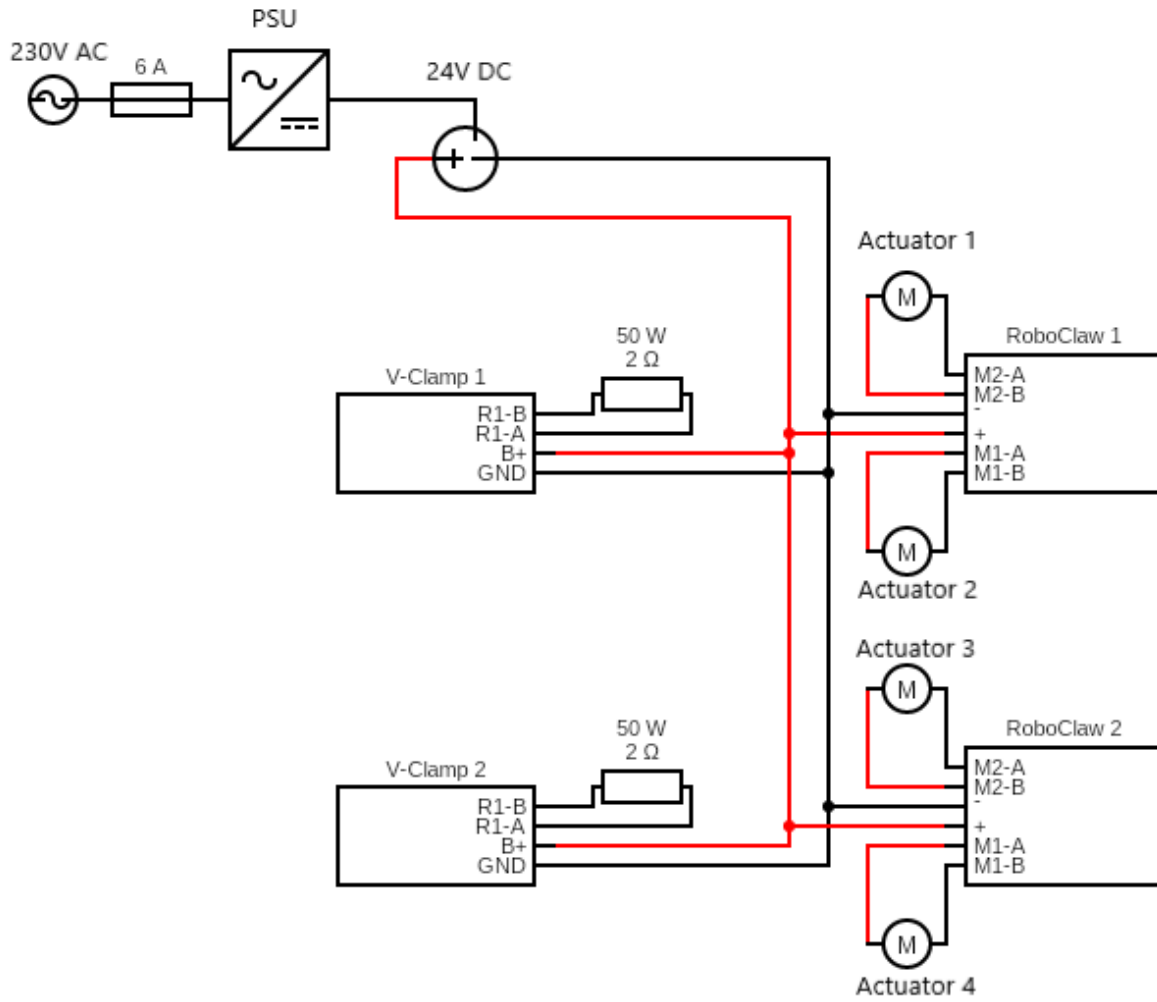
Referere til tekstboken på en mer moderat måte

Tydeliggjøre hvor "content" kommer fra

Alltid greit å ha full sources i referanseliste, og tittel under bilde

## F Electrical drawings







# Bibliography

- [1] Anatomy of a Joint - Health Encyclopedia - University of Rochester Medical Center, . URL <https://www.urmc.rochester.edu/encyclopedia/content.aspx?contenttypeid=85&contentid=P00044>.
- [2] Ankle Mobility, Ankle Instability, and Exercises to Strengthen Your Ankle, . URL <https://www.hoagorthopedicinstitute.com/blog/2018/july/ankle-mobility-ankle-instability-and-exercises-t/>.
- [3] Controlling the Speed of Linear Actuator, . URL <https://www.progressiveautomations.com/blogs/how-to/connect-dc-speed-controller-linear-actuator>.
- [4] Getting Started with Load Cells - learn.sparkfun.com, . URL <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all>.
- [5] Load Cells & Force Sensors, . URL <https://www.omega.com/en-us/resources/load-cells>.
- [6] Load Cell Accuracy, . URL <https://www.variohm.com/news-media/technical-blog-archive/load-cell-accuracy>.
- [7] Picking the Right Linear Positioning Device, . URL <https://www.design-engineering.com/features/linear-positioning-device/>.
- [8] What are the working principles of a beam, or shear beam, load cell..., . URL <https://www.flintec.com/weight-sensors/load-cells/how-does-a-beam-load-cell>.

- [9] What are the working principles of tension and s -type load cells, . URL <https://www.flintec.com/weight-sensors/load-cells/how-does-a-tension-load-cell-and-how-does-it-work>.
- [10] What is a Load Cell Amplifier?, . URL <https://www.omega.com/en-us/resources/load-cell-amplifier>.
- [11] What Is a Microcontroller? The Defining Characteristics and Architecture of a Common Component - Technical Articles, . URL <https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/>.
- [12] The working principles of a compression load cell manufactured by..., . URL <https://www.flintec.com/weight-sensors/load-cells/how-does-a-compression-load-cell>.
- [13] Load Cell Working Principle, May 2016. URL <https://instrumentationtools.com/load-cell-working-principle/>.
- [14] Belt-Driven Versus Ball Screw Actuator: Which Is the Best Choice for Your Application?, November 2019. URL <https://www.isotechinc.com/belt-driven-versus-ball-screw-actuators/>.
- [15] Orthopedic surgery, April 2021. URL [https://en.wikipedia.org/w/index.php?title=Orthopedic\\_surgery&oldid=1018961702](https://en.wikipedia.org/w/index.php?title=Orthopedic_surgery&oldid=1018961702). Page Version ID: 1018961702.
- [16] Admin. Using a Voltage Clamp with RoboClaw, July 2019. URL <https://resources.basicmicro.com/using-a-voltage-clamp-with-roboclaw/>.
- [17] Britannica. Control system. URL <https://www.britannica.com/technology/control-system>.
- [18] by. Ankle Anatomy, July 2015. URL <https://eorthopod.com/ankle-anatomy/>.

- [19] Benjamin R. Freedman, Joshua A. Gordon, and Louis J. Soslowsky. The Achilles tendon: fundamental properties and mechanisms governing healing. *Muscles, Ligaments and Tendons Journal*, 4(2):245–255, July 2014. ISSN 2240-4554. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4187594/>.
- [20] Dogan Ibrahim. Chapter 6 - Intermediate PIC18 Projects. In Dogan Ibrahim, editor, *PIC Microcontroller Projects in C (Second Edition)*, pages 173–325. Newnes, Oxford, January 2014. ISBN 978-0-08-099924-1. doi: 10.1016/B978-0-08-099924-1.00006-X. URL <https://www.sciencedirect.com/science/article/pii/B978008099924100006X>.
- [21] National Center for Biotechnology Information, U. S. National Library of Medicine 8600 Rockville Pike, Bethesda MD, and 20894 Usa. *How does the ankle work?* Institute for Quality and Efficiency in Health Care (IQWiG), December 2017. URL <https://www.ncbi.nlm.nih.gov/books/NBK279301/>. Publication Title: InformedHealth.org [Internet].
- [22] National Instruments. Pid theory explained. URL <https://www.ni.com/en-no/innovations/white-papers/06/pid-theory-explained.html>.
- [23] Lumen. Basics of kinematics. URL <https://courses.lumenlearning.com/boundless-physics/chapter/basics-of-kinematics/>.
- [24] B. R. Mehta and Y. J. Reddy. Chapter 9 - Serial communications. In B. R. Mehta and Y. J. Reddy, editors, *Industrial Process Automation Systems*, pages 307–339. Butterworth-Heinemann, Oxford, January 2015. ISBN 978-0-12-800939-0. doi: 10.1016/B978-0-12-800939-0.00009-7. URL <https://www.sciencedirect.com/science/article/pii/B9780128009390000097>.
- [25] Norman S. Nise. *Control systems engineering, 7th edition*. Wiley, 2015.
- [26] Prof. Larry Francis Obando. Mass-spring-damper system dynamics. URL <https://dademuch.com/2018/12/28/mass-spring-damper-system-dynamics/>.
- [27] Wikipedia. Degrees of freedom (mechanics). URL [https://en.wikipedia.org/wiki/Degrees\\_of\\_freedom\\_\(mechanics\)](https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)).