



# Bachelor's thesis



Faculty of Technology, Natural Sciences and Maritime Sciences  
Campus Kongsberg



**Course:** TS3000 Bacheloroppgave

**Date:** May 21, 2024

**Title:** DuoArm

This report forms part of the basis for assessment in the subject.

**Project group:** 9

**Group members:** Theo Magnor,  
Ask Lindbråten,  
Lorentz Tinney Rasmussen,  
Marte Overgaard,  
Adrian Kristiansen,  
Sophus Gjerstad

**Internal Supervisor:** Amir Safari

**External supervisors:** Ming Kit Wong

**Project partners:** Tronrud Engineering

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

---

## Acknowledgements

We would like to express our deepest gratitude to those who have supported us during the completion of the bachelor thesis. A special thanks to our supervisors Ming Kit Wong and Amir Safari for their unwavering support and guidance during this project. They have proved a valuable resource throughout the project, encouraging us to expand our view.

---

## Abstract

In this report, DuoArm presents the process of researching and developing a pick-and-place robot for stacking chip bags. Our proposed concept is a biaxial robot on rails that would substitute the current robots used in the packing industry. By working towards Tronrud Engineering (TE)s' requirements, we have developed a software user control system in ROS for controlling a down-scaled exhibition model via Human Machine Interface (HMI).

# Contents

Acknowledgements . . . . .	2
Abstract . . . . .	3
1 Introduction . . . . .	18
1.1 Overview . . . . .	18
1.2 Group members . . . . .	18
1.3 Writer and proof-reader . . . . .	18
1.4 Tronrud Engineering . . . . .	20
1.5 Task description . . . . .	20
2 Project management . . . . .	21
2.1 Group roles . . . . .	21
2.2 Supervisor and examiner responsibilities . . . . .	23
2.3 Choosing a project framework . . . . .	23
2.4 Management tools for tasks, documentation and communication . . . . .	26
2.5 Organization of meetings . . . . .	26
2.6 Use of artificial intelligence in academic writing . . . . .	28
2.7 Daily logging . . . . .	28
2.8 Website implementation . . . . .	28
3 Defining a scope . . . . .	30
3.1 Deciding the critical project path . . . . .	30
3.2 Tronrud Engineering's need . . . . .	32
3.3 User story . . . . .	32
3.4 Defining requirements . . . . .	33
3.5 Risk analysis . . . . .	36
4 Concept development . . . . .	42
4.1 Concept selection . . . . .	42
4.2 Pugh matrix . . . . .	47
5 Junior development . . . . .	56
5.1 Introduction . . . . .	56
5.2 Mechanical design . . . . .	56
5.3 Electrical design . . . . .	69
5.4 Configuring the fundamentals for software development . . . . .	70
5.5 Dynamic payload capacity challenge . . . . .	78

6	Senior development . . . . .	87
6.1	Mechanical . . . . .	87
6.2	The electrical setup . . . . .	191
6.3	The software development process . . . . .	204
7	Economy . . . . .	234
7.1	Product economy . . . . .	234
7.2	Project economy . . . . .	237
8	Conclusion . . . . .	238
	References . . . . .	240
	Bibliography . . . . .	249
	<b>Appendices</b>	<b>250</b>
	<b>A Additional explanations figure for the requirement table</b>	<b>251</b>
	<b>B Requirement table</b>	<b>253</b>
	<b>C Project timeline</b>	<b>259</b>
	<b>D Concept examination</b>	<b>261</b>
	<b>E Rapid Risk Analysis - reporting form</b>	<b>267</b>
	<b>F Individual pugh matrix - Delta robot</b>	<b>269</b>
	<b>G Individual pugh matrix - Duopod on rails</b>	<b>271</b>
	<b>H Individual pugh matrix - Two directional articulated robot arm on rails</b>	<b>273</b>
	<b>I Individual pugh matrix - Criteria description</b>	<b>275</b>
	<b>J Results from the initial dynamic payload capacity tests without gearing</b>	<b>277</b>
	<b>K Results from the dynamic payload capacity tests with gearing</b>	<b>285</b>
	<b>L Mechanical design</b>	<b>289</b>
	<b>M Electrical designs</b>	<b>290</b>
1	Controller . . . . .	291
1.1	Controller iterations . . . . .	291
1.2	Controller paths . . . . .	295
1.3	Led light testing . . . . .	297
2	Motors . . . . .	302
2.1	Torque requirement . . . . .	302
3	Bill of materials . . . . .	303

<b>N Software design</b>	<b>304</b>
1 Final lower-level architecture . . . . .	305
1.1 UC2-E1 - An operator presses the reset or stop the active process button when the system is in any of the other states besides <i>joystick_arm_control</i>	305
1.2 UC2-E2 - An operator presses the reset/"Stop the active process" button when the system state is equal to <i>joystick_arm_control</i> . . . . .	307
1.3 UC3&4-E1 - An operator moves the joystick towards the north or northeast	309
1.4 UC3&4-E2 - An operator moves the joystick towards the east or southeast	312
1.5 UC3&4-E3 - An operator moves the joystick towards the south or southwest	316
1.6 UC3&4-E4 - An operator moves the joystick towards the west or northwest	320
1.7 UC3&4-E5 - An operator pauses the movement of the joystick . . . . .	324
1.8 UC4&5-E1 - An operator presses the incorporated button of the joystick	328
2 Code documentation . . . . .	331
2.1 Arduino software . . . . .	331
2.2 The first iteration of the mapper node . . . . .	337
2.3 Command system . . . . .	340
2.4 GUI buttons . . . . .	341
2.5 Documentation generated by Doxygen . . . . .	341

# List of Figures

1	Scrum sprint illustrated . . . . .	25
2	Current week's distribution of responsibilities . . . . .	27
3	Next week's distribution of responsibilities . . . . .	27
4	Main principles and steps in a RRR . . . . .	37
5	Original and downscaled design space . . . . .	57
6	Delta concept . . . . .	58
7	Duopod concept . . . . .	59
8	Bill of materials for Junior Assembly . . . . .	62
9	Order placement to Item including article no., name, specification, link, description, order date, estimated delivery date and the arrival date. . . . .	62
10	Half of the Fastening hub assembly . . . . .	63
11	Active arm . . . . .	65
12	Passive arm . . . . .	65
13	Axle adapter . . . . .	66
14	6mm Arm spacer . . . . .	66
15	Short stabilizer arm . . . . .	68
16	Long stabilizer arm . . . . .	68
17	Effector . . . . .	69
18	GitHub or workflow operations illustration . . . . .	74
19	Remote - SSH workflow visualization . . . . .	76
20	Original mass . . . . .	81
21	Mass after downscaling and mass reduction . . . . .	81
22	Designed gears . . . . .	84
23	Implemented gears . . . . .	85
24	Design iteration 1 . . . . .	89
25	Design iteration 2 . . . . .	90
26	Design iteration 2.1 . . . . .	90
27	To the left validating production and design by measuring dimensions using a caliper gauge. To the right a fitting test with an M4 bolt. . . . .	91
28	Galvanic cell [1] . . . . .	93
29	Galvanic cell . . . . .	96
30	Minimum height and effector offset . . . . .	98

31	Maximum height . . . . .	99
32	Effector offset at conveyor position . . . . .	100
33	Frame . . . . .	102
34	Frame bill of materials . . . . .	103
35	Linear motion system using linear motion guide units with a lead screw and ball screw, driven by a rotary motor. . . . .	105
36	Example from Igus of a belt driven linear motion system. . . . .	106
37	Example from Igus of a linear motion guide with shafts[2] . . . . .	107
38	Rails assembly iteration 1 . . . . .	108
39	Rails assembly iteration 2 . . . . .	109
40	Motor connection rails assembly iteration 2 . . . . .	109
41	Interface rails assembly iteration 2 . . . . .	110
42	Rails assembly iteration 3 . . . . .	110
43	Top view of the rails assembly iteration 3 . . . . .	111
44	Rails assembly iteration 4 . . . . .	112
45	To the left previous motor bracket, to the right new motor bracket . . . . .	112
46	Parts in the Rails assembly . . . . .	113
47	Fastening hub iteration 1 . . . . .	115
48	Fastening hub iteration 2 . . . . .	116
49	Fastening hub iteration 3 . . . . .	117
50	Fastening hub iteration 3.1 . . . . .	118
51	Fastening hub iteration 4 . . . . .	119
52	Fastening hub iteration 5 . . . . .	120
53	Fastening hub iteration 5.1 . . . . .	121
54	Fastening hub iteration 6 . . . . .	122
55	Fastening hub iteration 6.1 . . . . .	122
56	Fastening-hub exploded view . . . . .	123
57	Bracket iteration 1 . . . . .	126
58	Bracket iteration 2 . . . . .	127
59	Bracket design iteration 3 . . . . .	128
60	Adapter design iteration 1 . . . . .	128
61	Adapter design iteration 2 . . . . .	129
62	Adapter design iteration 3 . . . . .	130
63	Fastening-bracket exploded view . . . . .	131
64	Axle design iteration 1 . . . . .	134
65	Axle design iteration 2 . . . . .	134
66	Axle design iteration 3 . . . . .	135
67	Axle design iteration 4 . . . . .	136
68	Axle design iteration 4.1 . . . . .	136
69	Machine key . . . . .	137

70	Axle sub-assembly exploded view . . . . .	138
71	Active arm iteration 1 . . . . .	141
72	Active arm iteration 2 . . . . .	141
73	Active arm iteration 3 . . . . .	142
74	Passive arm iteration 1 . . . . .	142
75	Passive arm iteration 2 . . . . .	143
76	Bracing . . . . .	143
77	Axle adapter iteration 1 . . . . .	144
78	Axle adapter iteration 2 . . . . .	144
79	Axle adapter iteration 3 . . . . .	145
80	Stabilizer arm assembly . . . . .	146
81	Arm assembly . . . . .	147
82	Bill of materials: stabilizer arm assembly . . . . .	148
83	Bill of materials: Arm assembly . . . . .	149
84	Stabilizer triangle . . . . .	153
85	Stabilizer triangle iteration 2 . . . . .	153
86	Stabilizer short arm: iteration 1 . . . . .	154
87	Stabilizer long arm: iteration 1 . . . . .	154
88	Stabilizer short arm: iteration 2 . . . . .	154
89	Stabilizer long arm: iteration 2 . . . . .	154
90	Stabilizer short arm: iteration 4 . . . . .	155
91	Stabilizer long arm: iteration 4 . . . . .	155
92	Stabilizer exploded view . . . . .	156
93	Stabilizer Bill of Materials . . . . .	157
94	Interface connection . . . . .	160
95	Tool hub - Arm interface . . . . .	161
96	Section view - Connection . . . . .	162
97	Camera mount [3] . . . . .	164
98	Transducer knob [4] . . . . .	165
99	Alternative 3 . . . . .	166
100	Alternative 4 . . . . .	167
101	Alternative 5 . . . . .	168
102	Iteration 1 . . . . .	169
103	Iteration 2 . . . . .	170
104	Iteration 3 . . . . .	171
105	Iteration 3.1 . . . . .	172
106	Iteration 4 . . . . .	173
107	Iteration 4.1 . . . . .	173
108	Iteration 5 . . . . .	174
109	Iteration 5.1 . . . . .	174

110	Iteration 6 . . . . .	175
111	Iteration 6.1 . . . . .	175
112	Tool hub exploded view . . . . .	176
113	Assembly mass . . . . .	177
114	Hinge pin . . . . .	178
115	Passive arm with constructed grip . . . . .	180
116	Connections . . . . .	181
117	Roller/slider . . . . .	181
118	Fixed . . . . .	182
119	Roller/slider . . . . .	182
120	Force 5 000N . . . . .	183
121	Mesh quality . . . . .	184
122	FOS . . . . .	185
123	Linear Elastic Isotropic . . . . .	186
124	Linear Elastic Orthotropic . . . . .	187
125	Rule of Mixture[5]. Where $\rho_c$ is the density of the composite, $\rho_f$ is the density of the fiber, $\rho_m$ is the density of the matrix, and $v_f$ is the volume fraction of the fiber. . . . .	189
126	Bonding agents for combinable elements [6] . . . . .	191
127	Linear power supply [7] . . . . .	192
128	Switch-Mode Power Supply (SMPS) block diagram.[8] . . . . .	193
129	Block diagram for the RD-85A power supply. [9] . . . . .	194
130	The controller design broken up. The red wires are 5 V, the black wires are ground, and the purple wires are for communication. . . . .	195
131	First iteration control design . . . . .	196
132	Prototype PCB board . . . . .	197
133	First iteration prototype controller . . . . .	198
134	Second iteration control design. . . . .	198
135	The controller design broken up. The red wires are 5 V, the black wires are ground, and the purple wires are for communication. . . . .	199
136	Controller design using the Arduino Mega. . . . .	200
137	LED circuit using MOSFET for current adjustment for the LEDs using an external power source. . . . .	201
138	The electrical cabinet layout. . . . .	203
139	Use case diagram - Operator . . . . .	205
140	System states - Overview . . . . .	206
141	Software architecture diagram . . . . .	208
142	Activity diagram describing how the operator interacts with the system during the mapping sequence. . . . .	214

143	Sequence diagram describing how the mapping sequence works from a lower level. Here we can see each node in operation during the sequence and the messages and function calls between them. . . . .	215
144	Sequence diagram describing how the mapping sequence works from a lower level. Here we can see each node in operation during the sequence and the messages and function calls between them. . . . .	216
145	Activity diagram describing the flow of path execution in the Motor Controller node, each diagram represents a function . . . . .	218
146	Sequence diagram describing the interaction between nodes during the path ex- ecution. . . . .	219
147	Analog value ranges in a cardinal and ordinal direction system . . . . .	223
148	Activity diagram - An operator presses the incorporated button of the joystick .	226
149	Activity diagram - An operator moves the joystick towards the north or northeast	228
150	Visualization of the cardinal and ordinal direction system in arm and joystick movements . . . . .	229
151	Activity diagram - An operator presses the stop the active process button . . .	231
152	Expenses provided by TE 1 . . . . .	235
153	Expenses provided by TE 2 . . . . .	236
154	Expenditures provided by the group members . . . . .	237
M.1	First iteration control unit design with a integrated controller design within the model. . . . .	291
M.2	Second iteration control unit design with an external controller with a connection in the front. . . . .	292
M.3	Third iteration control unit design with an unremovable cable that goes behind the model inside the electrical cabinet. . . . .	293
M.4	Fourth iteration control unit design with a USB port at the top of the model by the led logo using either Type-A/B or USB-mini port for the controller. . . . .	294
M.5	Fifth iteration control unit design using USB-Mini to USB connection with three buttons. . . . .	295
M.6	The red lines show the route the voltage travels between the microcontroller to the components. . . . .	296
M.7	The black lines show the route the ground goes between the microcontroller and the components. . . . .	296
M.8	The purple lines show the digital paths the signal take to the microcontroller whereas the yellow lines show the analog paths. . . . .	297
M.9	LSS voltage specifications.[10] . . . . .	303
N.1	Sequence diagram - An operator presses the reset or stop the active process button when the system is in any of the other states besides <i>joystick_arm_control</i>	306

N.2 Sequence diagram - An operator presses the reset or stop the active process button when the system state is <i>joystick_arm_control</i> . . . . .	308
N.3 Sequence diagram - An operator moves the joystick towards the north or northeast	311
N.4 Activity diagram - An operator moves the joystick towards the east or southeast	313
N.5 Sequence diagram - An operator moves the joystick towards the east or southeast	315
N.6 Activity diagram - An operator moves the joystick towards the south or southwest	317
N.7 Sequence diagram - An operator moves the joystick towards the south or southwest	319
N.8 Activity diagram - An operator moves the joystick towards the west or northwest	321
N.9 Sequence diagram - An operator moves the joystick towards the west or northwest	323
N.10 Flowchart - An operator pauses the movement of the joystick . . . . .	325
N.11 Sequence diagram - An operator pauses the movement of the joystick . . . . .	327
N.12 Sequence diagram - An operator presses the incorporated button of the joystick	330
N.13 Mapped workspace visualized with matplotlib . . . . .	337
N.14 Visualization of the robot or with the script made for testing the inverse kinematics equation. . . . .	339
N.15 Sequence diagram describing the first iteration mapping sequence. . . . .	340
N.16 Command window . . . . .	341
N.17 GUI buttons . . . . .	341

# List of Tables

1	Group members . . . . .	19
2	Pros and cons - full-scale product vs. downscaled prototype . . . . .	31
3	RRR - Level of consequence . . . . .	38
4	RRR - Level of probability . . . . .	39
5	RRR - Risk matrix . . . . .	40
6	RRR - Level of risk . . . . .	41
7	Criteria explainations for the ranking of scores. . . . .	48
8	Combined rating - Pugh Matrix . . . . .	50
9	Galvanic voltage table [11] . . . . .	94
10	Material combinations [11] . . . . .	95
11	Bill of Materials for the rails assembly . . . . .	114
12	Fastening-hub Bill of Materials . . . . .	124
13	Fastening-bracket Bill of Materials . . . . .	132
14	Axle sub-assembly Bill of Materials . . . . .	138
15	Tool hub Bill of Materials . . . . .	177
16	Overarching use case table . . . . .	209
M.1	Bill of materials for electrical components . . . . .	303
N.1	Sequence diagram table overview - An operator moves the joystick towards the north or northeast . . . . .	310
N.2	Flowchart table overview - An operator pauses the movement of the joystick . .	324

# Acronyms

**ACM** Abstract Control Mode. 212, 213

**ADC** Analog-to-Digital. 224

**AI** Artificial Intelligence. 28

**BUE** Built up edge. 190

**CAD** Computer-aided design. 77

**CFRP** Carbon Fiber Reinforced Polymer. 32, 44, 45, 95, 96, 100, 143, 154, 159, 161, 162, 179–181, 185, 189, 190

**CVD** Cemichal Vapro Deposition. 190

**DC** Direct Current. 191–194

**DNV** Det Norske Veritas. 36

**DT** Destructive Testing. 187, 189

**EEPROM** Electrically Erasable Programmable Read-Only Memory. 210

**EMI** Electromagnetic Interference. 192, 202

**FDM** Fused Deposited Modeling. 63, 67, 68, 91

**FEA** Finite Element Analysis. 179, 180, 184

**FOS** Factor of Safety. 180, 185

**GPIO** General-purpose input/output. 211, 212, 224

**HMI** Human Machine Interface. 3

**IC** Integrated Circuit. 192

**IDE** Integrated Development Environment. 75, 213

**JSON** JavaScript Object Notation. 214, 217, 328

**LED** Light Emitting Diode. 10, 192, 200–202, 210–213, 334

**LSS-HS1** Lynxmotion Smart Servo High Speed. 78, 79, 82, 83, 85, 86, 88, 115, 117, 119, 121, 125–127, 129, 135, 162, 173, 175, 210, 221, 224, 230, 309, 310, 328

**MACF** Modular Automated Case Filler. 20, 30, 32, 42–46, 56, 164, 167

**MDF** Medium Density Fiberboard. 66

**mm** Millimeter. 33, 79, 83

**MMR** Material Removal Rate. 125, 151, 158, 178

**MOSFET** Metal-Oxide-Semiconductor Field-Effect Transistor. 10, 192, 200, 201

**MVP** Minimum Viable Product. 18, 56

**NASA** National Aeronautics and Space Administration. 35, 71

**NH** Norsk Hydro. 36

**PCB** Printed Circuit Board. 10, 194, 197

**PID** Proportional integral derivative. 217, 219–221

**PLA** Polylactic acid. 63, 64, 66, 67, 69, 80, 85

**PLC** Programmable logic controller. 30

**PMC** Ceramic Matrix Composite. 188

**PMC** Metal Matrix Composite. 188

**PMC** Polymer Matrix Composite. 188

**PWM** Pulse-Width Modulation. 192, 200, 211, 221

**RGB** Red Green Blue. 200

**ROI** Return on investment. 46

**ROS** Robot operating system. 3, 70–72, 75–77, 201, 207, 211, 212, 224, 225, 232, 233

**RPM** revolutions per minute. 82, 221

**RRR** Rapid Risk Ranking. 7, 13, 36–41

**SMPS** Switch-Mode Power Supply. 10, 192–195, 202, 303

**SSH** Secure Shell. 73, 75, 233

**SW** SolidWorks. 22, 83, 84, 180

**TE** Tronrud Engineering. 3, 11, 20–22, 30, 32–34, 38, 44–46, 54, 56, 70, 71, 104, 120, 122, 154, 158, 179, 205, 217, 234–238

**TEA** Techno Economic Analysis. 234

**TH** Through Hole. 196

**UI** User Interface. 341

**UML** Unified Modeling Language. 21, 204

**URDF** Unified Robot Description Format. 76, 77

**USB** Universal Serial Bus. 210, 232

**USN** University of South-Eastern Norway. 22, 179, 189

**VM** Virtual Machine. 75

**VSC** Visual Studio Code. 75, 233

**WSL** Windows Subsystem for Linux. 75

# Glossary

**C++** A high-level programming language created in 1983 as an extension of the C programming language. 75

**ChatGPT** An artificial intelligence chatbot, that is based on a large language model, enabling it to respond to questions from users and compose written content. 28

**Eduroam** An international internet access roaming service for users in research and education. 75

**GitHub** A platform that allows software engineers to store, develop, share and manage their code. 72, 73, 75, 336

**Grammarly** A type of AI to help improve texts depending on the genre.. 28

**IP address** An address that is assigned to a device on a computer network. 75

**Junior** The first iteration of the exhibition model. 56, 59, 78–80, 85, 86

**LaTeX** A document preparation system used to compose scientific documents. 28, 336

**project life cycle** The project's life span from initiation to delivery. 26

**Python** A high-level programming language focusing on simplicity, that is widely used for web development and scientific computing for instance. 75

**Remote - SSH** An extension in Visual Studio Code that allows you to edit files on remote file systems using a SSH server. 75

**Senior** The final iteration of the exhibition model. 56, 78, 80, 86, 238, 302

**SMD 5050** A type of RGB light that gets its name from its 5x5 mm dimensions.. 200

**stakeholder** A person or group with a vested interest in a system. 204

**Ubuntu** An open-source Linux distribution that is, for instance, used for personal computers and cloud computing. 75

**USNexpo** A technology fair with projects in all engineering disciplines. 29, 44, 199

# 1 Introduction

## 1.1 Overview

**LTR | AK**

Robotic technology is evolving rapidly, with specialized robots now integrated to advance manufacturing processes like chip production. Within this domain, distinct robotic systems are employed for critical tasks such as spillage detection, stacking, and boxing. This report aims to identify the most effective robot for executing precise pick-and-place operations, a key function in the chip boxing process. We will evaluate various robots based on their performance, efficiency, and adaptability to this specific task. Further, we will outline the development of a downscaled display model, illustrating our findings and providing a representation of a more sophisticated MACF model in chip boxing automation. By creating two iterations of the model we will use the knowledge obtained by the MVP to create an even more capable model with functions fit for a display model. Through detailed testing and practical demonstration, we will uncover the most effective and capable design for the purpose of stacking chips in the MACF packing machine.

## 1.2 Group members

**LTR | AK**

The team, Deltaarm, comprises six members, each with a specific area of expertise. It includes three mechanical engineers, Marte, Adrian, and Sophus, alongside two in computer science, Theo and Ask, and an electrical engineer, Lorentz. Detailed information about their full names, initials, and respective roles within Deltarm can be found in Tab. 1.

## 1.3 Writer and proof-reader

**LTR | AK**

In the report, the names of both writers and proofreaders are indicated on the right side next to each heading. The writer's initial is highlighted in bold, while the rightmost proofreader's initial is displayed using an italic font. This approach ensures that every section is grammatically sound, well-organized, and error-free. Proofreaders are tasked with enhancing the overall quality of the document and removing minor errors. However, they do not assume responsibility for the content's original creation or take credit for the work of the writers. Their focus is on refining the document to ensure its high-quality

	<b>Name</b>	<b>Theo Magnor</b>
	<i>Initials</i>	TM
	<i>Discipline</i>	Software engineer - Cyber physical systems
	<i>Role</i>	Leader & Software-documentation
	<b>Name</b>	<b>Ask Lindbråten</b>
	<i>Initials</i>	AL
	<i>Discipline</i>	Software engineer - Cyber physical systems
	<i>Role</i>	Project economy & Software-architecture
	<b>Name</b>	<b>Lorentz Tinney Rasmussen</b>
	<i>Initials</i>	LTR
	<i>Discipline</i>	Electrical engineer - Cybernetics
	<i>Role</i>	System engineer & Electronics & Report
	<b>Name</b>	<b>Marte Overgaard</b>
	<i>Initials</i>	MO
	<i>Discipline</i>	Mechanical engineer - Product development
	<i>Role</i>	Production & Design & Sprint organizer
	<b>Name</b>	<b>Adrian Kristiansen</b>
	<i>Initials</i>	AK
	<i>Discipline</i>	Mechanical engineer - Product development
	<i>Role</i>	Mechanical & Social media & Product economy
	<b>Name</b>	<b>Sophus Gjerstad</b>
	<i>Initials</i>	SG
	<i>Discipline</i>	Mechanical engineer - Product development
	<i>Role</i>	Materials technology & Documentation

Table 1: Group members

## 1.4 Tronrud Engineering

**LTR | SG**

TE has been a prominent player in the packaging industry since its establishment in 1977. The company prides itself on its commitment to engineering excellence and started by providing custom-made machinery solutions. Over the years, TE has expanded its areas of expertise, catering to a wide range of industries, including food, pharmaceuticals, and industrial products. Today, TE offer various packaging solutions, ranging from snacks, dairy, bakery, and seafood products to specialized equipment for railway production and wax applicator machines.

One of TE's latest innovations is the MACF machine, designed explicitly for chip bag packaging. This advanced machine features leakage detection sensors, conveyor belts for efficient transport, and stackers for precise chip placement. It also includes third-party delta arms to facilitate chip transfer between the conveyor belt and stacker. However, TE is researching and developing a more cost-effective alternative due to the delta-arm components' high costs and overlapping functionalities. The goal is to enhance the efficiency and affordability of TE's packaging machinery, building on our research of innovative engineering. [12]

## 1.5 Task description

**AL | SG**

The students, on behalf of TE, will explore production possibilities and the functioning of 'Pick and place' robots, as well as develop a downscaled solution. The purpose is to provide the company with a basis for further development of such robots for their machines. The product should only prove certain parts of the concept, but it is a requirement that the product is made of composite, as the company wishes to acquire more knowledge in the area. Therefore, extra emphasis is placed on production methods using carbon fiber composite. In addition, great importance is also placed on thorough documentation, as well as minimizing costs. Typical for 'pick and place' machines is that they move products from A to B using a retrieval tool, where point A is often in motion. The development of the retrieval mechanism is not a part of the task.

## 2 Project management

AL | AK

The following section deals with how the team handled the execution of administrative- and organizational tasks during the project development. This includes defining the specific roles assigned to each group member, detailing the responsibilities of our supervisors and examiners, describing the process of selecting a project framework and how it was enacted, defining the project tools used for communication, explaining how the meetings were organized and the daily logging was carried out, and lastly, discussing the implementation of the team's website.

### 2.1 Group roles

SG, AL | LTR

Each group member has been assigned two or more main areas of responsibility. This facilitates hands-on experience for each member with various aspects of engineering projects and optimizes work efficiency by evenly distributing the workload among participants.

#### 2.1.1 Theo Magnor

***Group leader:***

This role involves having the main responsibility for scheduling meetings with the group members, maintaining good communication within the group, and controlling discussions to keep everyone on track.

***Software-documentation:***

This responsibility involves creating structured and understandable documentation for the developed software.

#### 2.1.2 Ask Lindbråten

***Project economy:***

This area of responsibility involves maintaining control of all expenses provided by the group members, while providing outlays for reimbursement to TE.

***Software-architecture:***

This task involves having the main responsibility for using UML to visualize software design and create a solid foundation for further development.

#### 2.1.3 Lorentz Tinney Rasmussen

***Main report:***

This area of responsibility involves having the main accountability for structuring the group's bachelor's thesis, which entails maintaining a professional look and quality control of the document and writings. Additionally, it also includes ensuring that the quality of documents

delivered to examiners and supervisors is at an acceptable level.

### ***Systems Engineer:***

This role entails maintaining control and oversight over the entire development process and its various phases, ensuring that we remain on track and can reach our milestones.

### ***Electronics:***

This area of responsibility involves having the primary accountability for the electrical design and architecture of the exhibition model.

#### **2.1.4 Adrian Kristiansen**

### ***Social media and promotion:***

This role entails being the main responsible for the group's Instagram page, which involves promoting the group and the project work done throughout the semester.

### ***Mechanical:***

This task involves being the main responsible for the mechanical functionality and interface, ensuring that both designed and produced components will fit and work as intended.

### ***Product economy:***

This task involves maintaining control of all expenses related to the product, which includes the costs of material investment and projected production.

#### **2.1.5 Marte Overgaard**

### ***Production:***

This task entails being primarily responsible for the production of parts. It includes keeping an overview of what components we need to produce at both TE and USN and how long this will take.

### ***Design:***

This task involves being the main responsible for designing parts. It includes maintaining a good structure of related documentation and assemblies within the SW feature tree.

### ***Sprint organizer:***

This role involves being primarily responsible for adding and structuring tasks at the beginning of the groups sprints.

#### **2.1.6 Sophus Gjerstad**

### ***Documentation:***

This responsibility includes facilitating structured plans and templates for documentation, log-

ging joint daily reports, ensuring that agenda and meeting minutes are logged in a structured and professional manner, and reminding participants to write individual daily reports and maintain timekeeping.

***Material technology:***

This area of responsibility involves having the main accountability for material testing and their specifications, focusing on how they are manufactured, shaped, processed, joined, and used for components.

## **2.2 Supervisor and examiner responsibilities**

**AL | SG**

### **2.2.1 External supervisor**

The external supervisor's main responsibility is to provide advice on the formulation and delimitation of the topic and research question, discuss and assess hypotheses, methods and results, and keep abreast of the progress of the students work.[13]

### **2.2.2 Internal supervisor**

The internal supervisor will follow up with the project group to support and guide students while writing their bachelor's thesis. This includes reviewing and commenting on the project documents provided by the group members as needed, conducting an individual meeting with each student to review their contributions after the second presentation, and participating in a panel of three people who assign individual marks to the project members.

### **2.2.3 External examiner**

The external examiner will assess the project group's task, presentations and final thesis.

### **2.2.4 Internal examiner**

The internal examiner's responsibility includes assessing the project group's documentation, evaluating each participant's contributions to the project work, in accordance with the competence targets, and being present at the pre-meetings and the ones that come after each presentation. [14]

## **2.3 Choosing a project framework**

**AL | SG**

A project framework refers to a set of methods, principles, and procedures that establishes the basis of executing a project from initiation to delivery [15]. It also brings the following five benefits: consistency, clarity, collaboration, continuity, and communication. A framework imparts consistency and continuity to the entire project life cycle due to its strict emphasis on detailed project planning. This emphasis reduces ambiguity during each stage of development, eases the transition between phases, and increases clarity among participants. As a result, the

end product becomes more coherent. Facilitating a cohesive and collaborative approach among project members and stakeholders, from the initial stages to delivery, also helps to smooth the project's workflow, alleviate potential barriers to communication, and ensure the fulfillment of important milestones. Ultimately, this leads to improved quality of the end product. [15]

Establishing a project framework requires careful evaluation of the project's scope and scale [15]. Given that undertaking a project of this magnitude is a relatively new experience among the participants of our group, we chose to adopt a project framework that supports agile methodologies and test-driven development. This decision ensures efficient allocation of human resources, adept management of unexpected challenges, and facilitates a continuous feedback loop, allowing us to successfully learn from previous mistakes and plan the next steps in the project process.

After thorough research, we initially decided to utilize a combination of crucial aspects from two project frameworks: Scrum and Unified Process. We planned to incorporate sprints and task logging from Scrum alongside Unified Process's defined project plan and component-based architecture. However, the complexity of blending these elements and the resulting confusion and obscurity among our team members led us to settle with a single project model: Scrum, combined with a custom-made project plan. As for the component-based architecture, we will keep it in mind through design and development.

### **2.3.1 How was it enacted?**

Our project process is structured around three phases: The initial "planning and deciding of scope" phase, the "concept and Junior development" stage, and the "Senior development" phase (see Fig.C). The former spanned a monthly period from mid-January to mid-February, while the latter consisted of two or more bi-weekly sprints. The reasoning behind utilizing two weekly sprints is that our mechanical engineers cannot work as iterative compared to other disciplines and require more time for design and production efforts.

Undertaking a new sprint is initiated by planning, discussing, and dividing the primary tasks for that sprint. These encompass newer main assignments and carryover tasks from the administrative- and product backlog. Furthermore, design, development, implementation, and testing efforts ensue, in which tasks are checked off continuously and others are added individually. Daily scrums and weekly meetings with supervisors are intended to be held during this stage. The former is conducted to clarify the daily work agenda, while the latter is done to provide updates on technical and academic work efforts. The last stage consists of a review, during which any unfinished tasks are added to the administrative and product backlog, and the sprint is evaluated to assess the project development status and discover potential areas for improvement. Fig.1 illustrates how a sprint can be visualized.

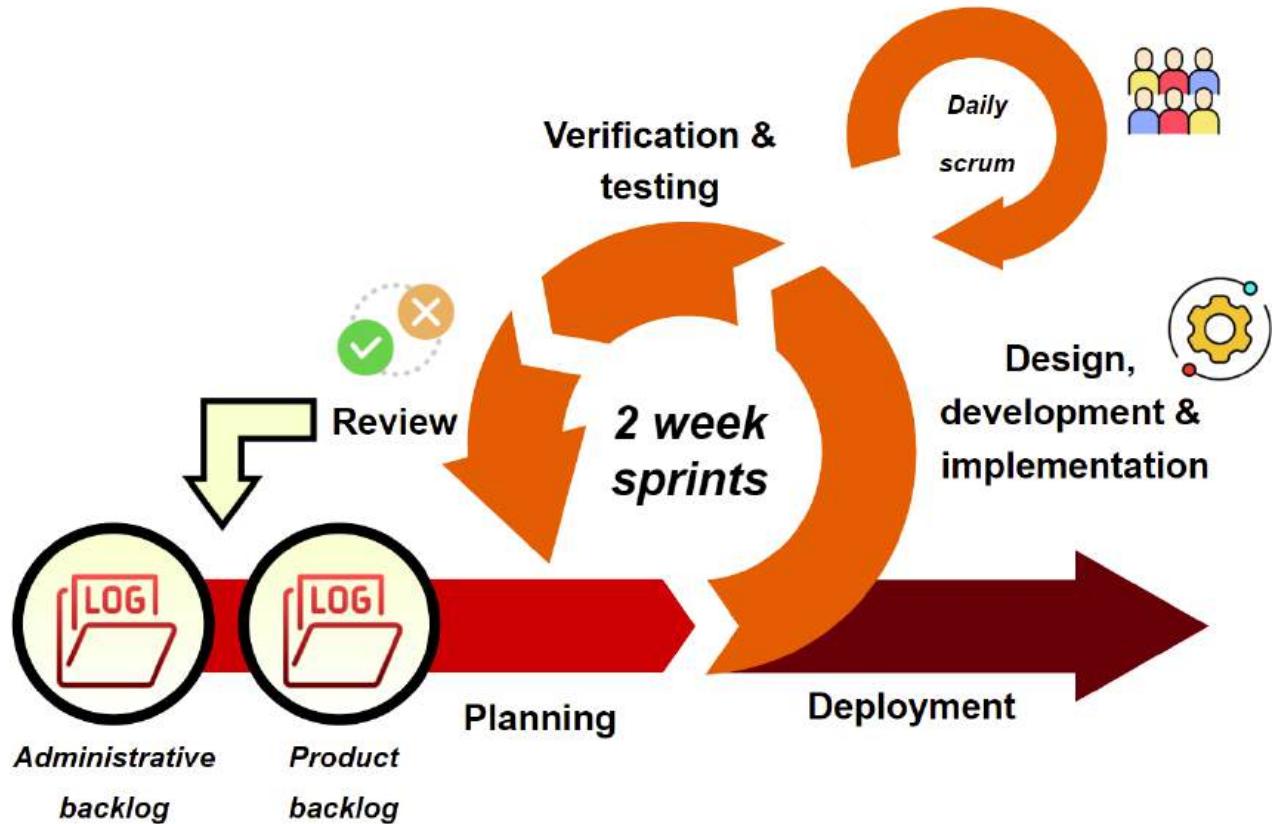


Figure 1: Scrum sprint illustrated

"Log" icon reference: [16]

"Gear" icon reference: [17]

"Review" icon reference: [18]

"People" icon reference: [19]

## **2.4 Management tools for tasks, documentation and communication**

**AL | SG**

Throughout the project life cycle, our team have utilized "Messenger" for non-project related communication and "Microsoft Teams" to manage tasks, handle documentation, and facilitate communication related to the development of the prototype. By confining ourselves to a single platform for project-related work, we reduce the number of potential areas of friction and enhance efficiency in work-related collaboration. In regards to task management during sprints, we have made use of Teams' designated task field. This option provides columns for backlogging, to-do assignments, and in-progress-, finished-, and temporary paused tasks, while keeping an overview of efforts from earlier sprints. As for project-related documentation and communication, Teams facilitates arrangement of online meetings, and provides folder structures that allow for live updates from multiple contributors and storage of various file types.

## **2.5 Organization of meetings**

**AL | LTR**

The weekly meetings with supervisors represented unique opportunities to gain valuable administrative experience. Hence, we decided to organize the meetings by splitting the administrative responsibilities into two separate roles with weekly rotations. These roles included a meeting leader, responsible for leading the meeting and creating the agenda, and a minutes of meetings responsible, tasked with documenting the minutes of each meeting. This way, we facilitate hands-on experience with work-related meeting formalities for each member, while ensuring an even distribution of the additional workload. Fig. 2 and Fig. 3 visualize how the areas of responsibility were rotated weekly.

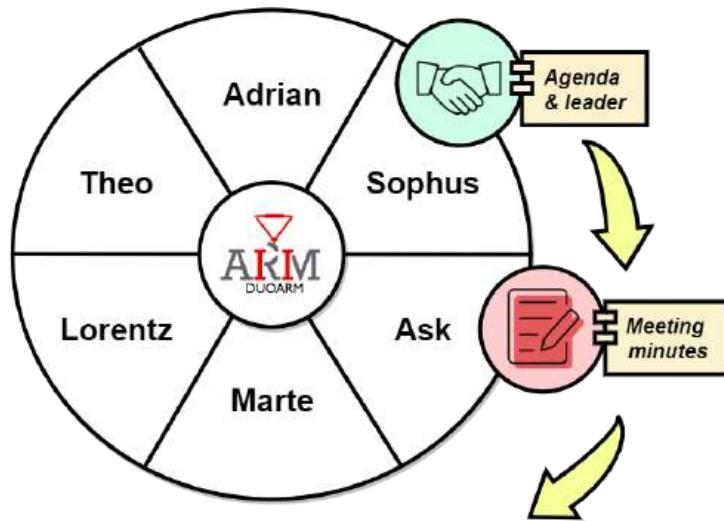


Figure 2: Current week's distribution of responsibilities

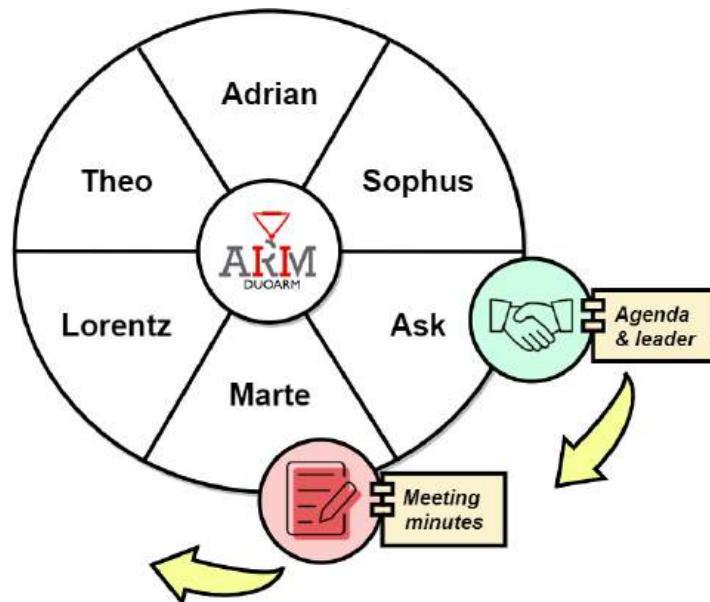


Figure 3: Next week's distribution of responsibilities

"Handshake" icon reference: [20].

"Tablet and pencil" icon reference: [21]

## 2.6 Use of artificial intelligence in academic writing

AL | SG, LTR

Throughout this project, our team has utilized ChatGPT and Grammarly to aid in formulating paragraphs and gathering synonyms. However, this does not imply that any information has been copied directly and utilized in either texts, figures, templates or tables in this thesis. Instead, they have been used as tools to maintain an academic and continuously professional writing style by prompting the writer's paragraph and asking for inputs to improve the flow, clarity and grammar. The potential responses from the AIs were then utilized as guidelines on how to rewrite the paragraph in a more professional manner, while maintaining the writers' genuine sentence structure and original content. Additionally, we have used it to acquire code to perform specific actions in LaTeX and obtain technical information for purely practical applications.

## 2.7 Daily logging

SG | AL

Throughout the project period we have decided to log every activity and task performed. This enables us to backtrack our progress and activity, giving us an overview of our daily accomplishments. Additionally, it provides a snapshot of what we have done and the different challenges that need to be addressed. Furthermore, it also highlights what we have done to work around those challenges.

We have also chosen one person to document our daily report for common workdays when the whole team works together, as well as doing individual logging for tasks given to specific members. In addition, the individual logging is a valuable document that can be referred to as a performance evaluation. By reviewing the daily report, supervisors can assess productivity, identify individual and group-related strengths and weaknesses, and provide feedback to team members as needed. In addition to the daily log, members are obligated to document work hours in a timetable. This gives an overview of each individual's work efforts, along with the total time utilized in different phases and throughout the whole process. Hence, this will promote a sense of responsibility and accomplishment during the project and help to ensure that everyone is aligned and working towards a common goal. In conclusion, documenting our daily activities and time spent will promote ownership, transparency, and ensure that every member contributes and participates in the project.

## 2.8 Website implementation

TM | LTR

One of the tasks included in the bachelor thesis is making a website for our project, it would be a platform designed to serve multiple purposes. Primarily, it serves as a central point of information where interested parties can learn about the project's scope and the solution we developed. Recognizing the importance of transparency and ongoing communication throughout the project's lifecycle, a blog section was integrated to share updates on our development process, allowing followers to track our progress. Additionally, you can find the complete report

and contact information for all participants, making it a valuable resource for those seeking detailed insights or wishing to connect with the team. The website also provides us with great value, as it showcases our capabilities to future employers. It will also be featured at the USNexpo, alongside our exhibition model, providing a digital complement to our physical presentation. Implementing the website involved selecting a free template online, which was then customized to better align with our project's identity. The website was built using HTML, JavaScript, and CSS.

## 3 Defining a scope

### 3.1 Deciding the critical project path

AL | SG

On January 11th, in the project's early stages, we were invited by our external supervisor to visit TE at *Eggemoen* in *Hønefoss*. Briefly summarized, the visit involved a guided tour of their different warehouses and a deeper insight into the technical and practical functionality of the packaging machine MACF, with the main focus being on how the chip bags were transported from the pattern builder to the boxes. Following the tour, a joint meeting comprised the project team, our external supervisor, TE's department manager, and an employed constructor. Here, we were prompted with a crucial pivot point that demanded thorough consideration: Whether to develop a full-scale product of the delta robot that the client could potentially integrate into their packaging machines or a downscaled prototype, referred to as "senior" or the "exhibition model". This process required multiple iterations of discussions, and the considerations for each path are detailed in Tab. 2 below.

To clarify some of the main points listed, opting to develop a full-scale product would result in the client receiving a more applicable product with lower needs for continuous development. Moreover, TE would also facilitate greater accessibility to the required hardware for our mechanical engineers and provide a pre-built simulator for our software engineers. However, choosing this path also has its disadvantages.

Firstly, according to the course plan, the software efforts would appear less academically relevant. A significant amount of time would have been spent on DevOps practices due to the necessity of utilizing the unfamiliar Siemens Programmable logic controller (PLC) technology platform. Two counterarguments to this could be that our team would acquire invaluable experience with specialized industry technologies and that greater follow-up opportunities would be facilitated due to the client's familiarity with the technology. But, they do not outweigh the disadvantages.

Secondly, our electrical engineer would be restricted to utilizing Siemens engines and existing power solutions provided by TE. This leaves little room for research and justifications regarding voltage regulation and engine selection. Furthermore, employing other engine options would lead to lowered voltage levels overall, aligning more closely with his academic efforts and practical applications encountered throughout his studies.

Thirdly, all the hardware would need more complex manufacturing methods, and our mechanical engineers would greatly depend on using TEs workshop. This would result in less flexibility for the affected students, and much time would have been spent commuting. Lastly, by opting to develop a downscaled prototype, the mechanical engineers can also benefit from applying a

broader range of knowledge from their previous subjects while still utilizing the same materials as they could with the full-scale product. Ultimately, with these facts and more considered, deciding to develop a downscaled prototype seemed like the most reasonable option. It is also important to note that while this decision was specifically made with the delta-robot concept in mind, the considerations also remain applicable to other concepts.

Table 2: Pros and cons - full-scale product vs. downscaled prototype

<b><i>Considerations for choosing between developing a full-scale product or downscaled prototype:</i></b>			
<b>Pros</b>		<b>Cons</b>	
<b>Full-scale</b>	<b>Downscaled</b>	<b>Full-scale</b>	<b>Downscaled</b>
<ul style="list-style-type: none"> <li>✓ The customer would receive a better and more applicable product, with lower needs for further development.</li> <li>✓ Facilitates closer follow-ups, given the customer's familiarity with the technology.</li> <li>✓ Our team would gain more experience with unique industry technologies.</li> <li>✓ Our team would have easier access to the required hardware.</li> <li>✓ Access to a pre-built simulator for the product.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Easier for the software engineers to apply previous knowledge.</li> <li>✓ The mechanical engineers can construct a prototype, while applying a broader range of knowledge.</li> <li>✓ The mechanical engineers can still utilize the same materials as they could with the full-scale product.</li> <li>✓ There would be a greater workload for the electrical engineer.</li> <li>✓ The customer would benefit from their ability to draw broad knowledge from our efforts.</li> <li>✓ Implementing familiar technologies would minimize the time spent on DevOps practices compared to adapting to a new technology platform.</li> </ul>	<ul style="list-style-type: none"> <li>✗ The software-work would be less academically relevant according to the course plan.</li> <li>✗ More travel time to "Hønefoss" or "Tronrud Engineering".</li> <li>✗ Reduced workload for the electrical engineer.</li> </ul>	<ul style="list-style-type: none"> <li>✗ The mechanical engineers need to promptly construct an initial prototype so the software engineers can run tests iteratively.</li> <li>✗ Demands continuous product development from the customer.</li> </ul>

## 3.2 Tronrud Engineering's need

SG | LTR

TE requires a new robot to fit inside their already existing packaging machine Modular Automated Case Filler (MACF). This robot will replace the work of the one used today. It must be designed and implemented to fit inside the layout while maintaining the same functionality as the current robot. The need is derived from the current robot being too large and unnecessarily expensive for the given task. Furthermore, there is a wish to serial-produce one of their own to keep expenses down. This could be done by critically choosing the weight, materials, gears, and motors more cost-effectively. In addition, they want to minimize the work area to ensure that the robot is kept within its assembly line. This will ensure the safety of individuals doing service and repairs on one side of the assembly line without shutting down both lines. The firms using MACF will also save much money and packaging time with a new solution. In addition, TE would like to keep most of the production in-house, so the importance of cost-efficiency can't be stated enough. The reason behind this is to keep the value creation within Norwegian borders. Additionally, TE is a knowledge-seeking company that wants to acquire new insights within composites directed at Carbon Fiber Reinforced Polymer (CFRP), so we need to implement this into our design and ensure it is documented exemplary.

## 3.3 User story

TM | SG

User stories are useful in defining the scope of the project. It gives an important perspective on the product that can't be neglected. Specifically, a user story helps in capturing needs of the user or customer. They articulate what the user wants to achieve or what problem they need solved, ensuring the product development is driven by user demand rather than assumptions. This is our user story:

*In the heart of Tronrud Engineering's bustling lab, Jack, a resourceful engineer, stands before their latest challenge. A chip-producer in Europe wants a packaging machine to revolutionize the way bags of chips are boxed. His task is intricate, yet vital, creating a system that seamlessly packs the chip bags from a conveyor and precisely stacks them into cardboard boxes of varying sizes.*

*Every day, Jack observes the harmony of conveyor belts with their speed and delicacy. The machine needs to be more than just efficient. It must handle the chip bags gently, while ensuring adaptation to their different shapes.*

*Tronrud Engineering's ethos of efficiency and cost-consciousness are important to Jack. He pictures a safe and user-friendly smart system equipped with fine-tuned sensors and agile robotics, all working in unison. It's not just about packing chips; it's about maintaining productivity without neglecting the production quality. As an engineer, Jack needs to develop a product to meet the demands from his customer, the chip-producer.*

### 3.4 Defining requirements

AL | SG

Defining requirements is an essential part of any development process. It can guide a project through its various stages and details the product's features, capabilities, behaviors, and functions to be developed [22]. It also forms the basis for ensuring the project team knows their goals and limitations and what they need to achieve to meet stakeholders' needs [23].

Specifying requirements is often a thorough process, particularly in the initial stages, and our project is no exception: It demanded multiple iterations and weekly feedback, validation, and clarifications provided by our external supervisor to ensure that TE's needs were well accounted for. Some of the most notable requirements were predefined early on and adjusted to align with the task description and user story in addition to our client. These revolved around the following key areas: Its ability to perform a linear and horizontal "pick and place" movement in the three coordinate axes, its composition and the fact that it should consist of carbon fiber and aluminum and be cost-efficient, as well as being user-friendly and potentially topology-optimized for handling external stressors and completing a desired amount of "pick and place" movements within 60 seconds. Furthermore, some requirements were solely derived from TE, the task description, or the user story. The ones that solely originated from TE referred to the fact that the prototype should have a manual control system, be transport- and exhibition-friendly, be driven by power from a socket and have an emergency procedure. The ones that were solely derived from either the task description or the user story revolved around the prototype's ability to have an option of mounting different gripping tools, and a tool attached to the implement adapter.

Throughout January, the weekly feedback sessions were conducted online via Teams. However, on the 1st of February, we decided to host our first physical meeting with only the external supervisor. The purpose was to validate the content for our first presentation on the 8th of February and finalize the most crucial administrative- and project-related tasks that needed to be completed before that date. Details about the main discussions from this day can be found in the daily report from pages 10 to 18. Still, about this subsection, we were prompted with a few additional requirements regarding the Senior prototype: It is supposed to have a rig construction surrounding it that does not exceed (500x500x500)(Later changed to (500x500x600) Millimeter (mm) in the concept development stage) Mm, as well as a maximum total weight of 10 kg, including the actual prototype, the rig construction, electrical cabinet and control system. The prototype's working area (defined as the area where the prototype should be able to manipulate chip bags) in height should also have a 10 to 20 percent error margin to account for further design at TE. Still, it should not exceed 500 mm (Later changed). The working area in the x- and y-directions will be decided in the conceptual development phase.

Furthermore, it was evident that we should document requirements for both the full-scale

product and the exhibition model and divide them between those that apply to the full-scale product or the exhibition model separately in an Excel table. The idea was to create a document that not only differentiated between the requirements of the full-scale product, for which we were only tasked with designing, and Senior, for which we were supposed to both design and develop but also facilitated simple control, referencing, verification and updates of the requirements should the project parameters change or issues arise [22]. The former meant that the requirements for Senior demanded the most attention due to the need for physical verification, but that documenting and theoretically verifying the requirements for the full-scale product, through design in SolidWorks, remained necessary to reduce the potential workload TE would have to undertake upon taking over our project and academic research.

Despite this, we were called in for an online meeting the following day together with our external supervisor, which brought about a critical change of events: We agreed to opt for only designing and developing the Senior prototype, as the workload would pass beyond that of a bachelor's thesis otherwise. Consequently, this meant solely documenting, addressing, and verifying the requirements for the exhibition model, which can be listed and specified as follows:

- The prototype shall be able to perform a vertical and horizontal "pick and place" movement in the three coordinate axis directions (x, y, z).
- The prototype must have an option of mounting different gripping tools.
- The prototype must conform to TE's material bonding and composite requirements.
- The prototype must be cost-efficient.
- The prototype must be powered with electricity from a socket (voltage of 220-240).
- The prototype must be user-friendly.
- The prototype must be able to be transported.
- The prototype must be able to be used for exhibition.
- The prototype shall have a manual control system.
- The prototype's working area in height (movement in the z-direction) must have a 20 percent safety margin to account for further design.
- The prototype's potential building area in height must not exceed 600 mm.
- The prototype must be exhibition-friendly by design.
- The prototype must have an emergency procedure.
- The prototype should be topology optimized.

- The prototype should have a tool attached to the implement adapter.

The listed sub-subsections below discuss the key aspects of our requirement table (App.B). Additionally, see Fig. A for explanations of essential terms, along with visualizations of how the priorities and completion statuses were color-coded.

#### 3.4.1 Notation

AL | SG

Each requirement has been formulated based on NASAs guidelines for writing effective requirements [24]. This implies choosing the appropriate terms to distinguish between the most critical requirements of our project and the ones that are merely desirable or intended to enhance the design. The most critical ones are phrased with the modal verbs "shall" and "must", while the others are composed with the term "should". We have also utilized their validation checklist [24] to ensure clarity, completeness, and verifiability. This involves conveying one single thought per requirement, making them concise, simple, and clear, and formulating them to be verifiable.

#### 3.4.2 Priorities

AL | SG

Regarding distinguishing between critical and optional requirements, we have assigned each one a rating of either A, B, or C. An "A" rating represents the most fundamental ones that need to be in place to successfully meet our client's expectations, while a "B" or "C" rating indicates optional enhancements. A "B" rated requirement is prioritized above a "C".

#### 3.4.3 Identification-marking and verifiability

AL | SG

To simplify referencing and facilitate effective control of requirements, we have assigned each one a unique identifier. The same principle applies to each verification method, where the rightmost number corresponds to the number in the identifier of the associated requirement. Regarding the requirement table, some requirements have also been assigned more than one verification method. The idea behind this is that each verification has to be completed for the requirement to be fulfilled.

#### 3.4.4 Completion status

AL | SG

The completion status column has been added to indicate if a fulfillment of a requirement, or rather a completion of a specific verification method, has not been started if it has been initiated or is under testing or completed.

## 3.5 Risk analysis

MO | AL

A risk analysis should be conducted to identify potential risks to a project. During this process, the team should consider possible risks that could affect the project, as well as risks arising from the project, such as those affecting other people and the environment. By assessing risks early on through this process, it is possible to prevent or mitigate major hazards. [25]

### 3.5.1 Rapid Risk Ranking

MO | AL

We will use Rapid Risk Ranking (RRR) when reviewing different aspects that can go wrong in our project. RRR is a method used to efficiently assess and prioritize risks based on their consequence and probability. We have assembled our own RRR setup, with a lot of inspiration from Norsk Hydro (NH) and Det Norske Veritas (DNV)'s report "Methodology for RRR of H2 Refuelling Station concepts" [26]. We used some of their tables and created modified versions to fit our project. RRR is suitable for early concept assessments and can help to identify the need for more detailed analysis or measures.

#### Main principles:

- Identify and categorize risks according to their source type and impact.
- Assessing the probability and consequence of each risk using a scale.
- Multiplying the probability and consequence to get a risk score for each risk
- The risks will be ranked to be either low, medium, or high.
- Develop measures to reduce or avoid the risk.
- Increases awareness and understanding of the risks among all parties involved.

See Fig.4 for further explanations.

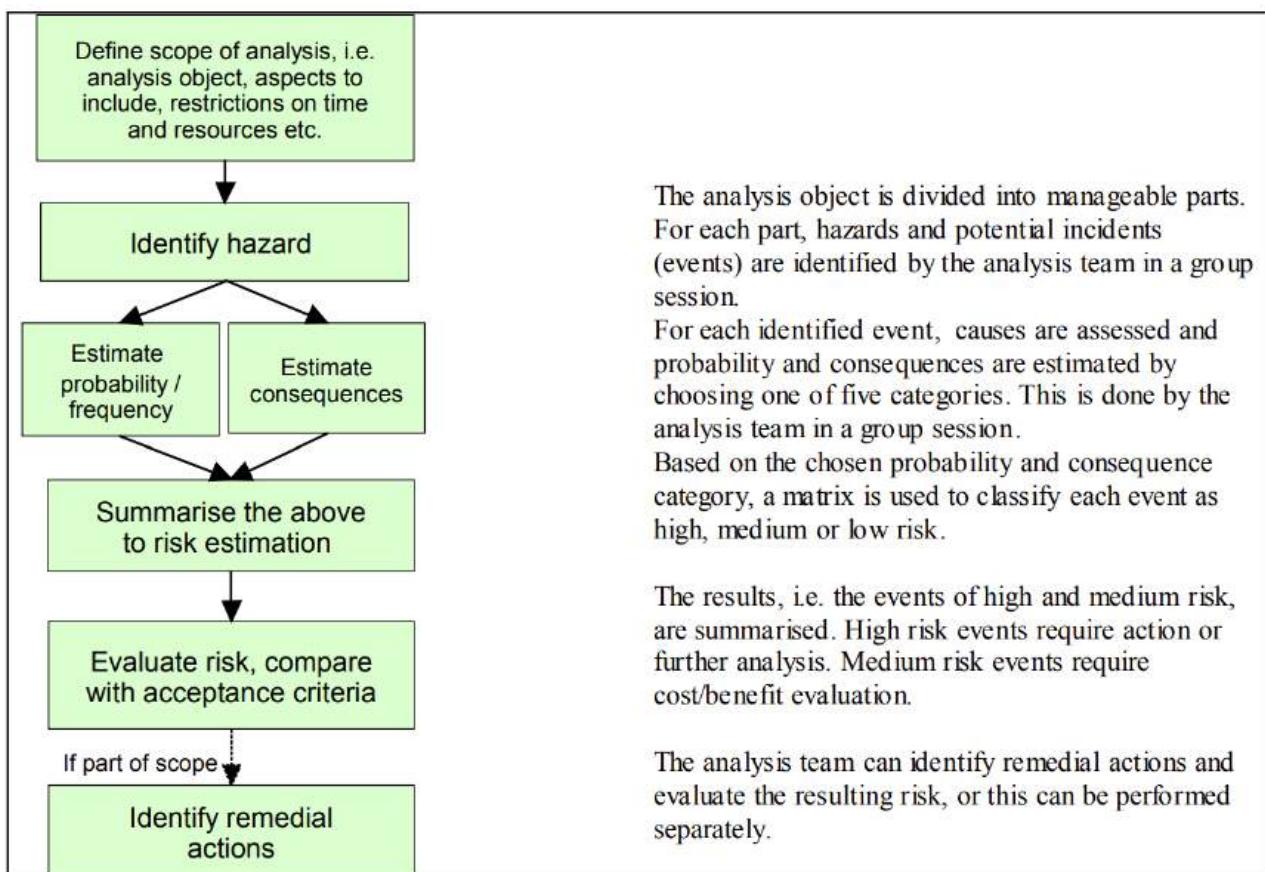


Figure 4: Main principles and steps in a RRR

[26]

**Table structures:*****Level of consequence:***

Table 3: RRR - Level of consequence

Level of consequence						
Level	Definition	Description				
		People	Environment	Materialistic property	Project group	Tronrud Engineering
1	Insignificant	Minimal impact on individuals, possibly unnoticed or having negligible changes in their lives.	Negligible or no environmental impact with minimal changes to ecosystems or natural resources.	Minor or negligible impact on material possessions, with minimal loss or damage.	Minor or negligible impact on the project group, with minimal disruption to workflow or objectives.	Minimal impact on the firm, with negligible consequences for operations and stakeholders.
2	Minor	Minor effects on people's lives, perhaps causing small inconveniences and adjustments.	Minor impact to the environment, possibly affecting local ecosystems.	Little damage or loss to material possessions requiring minor repairs or replacement.	Slight disruptions to the project group, potentially requiring adjustments or additional efforts to overcome challenges.	Slight disruption or challenges for the firm, requiring minor adjustments and/or responses.
3	Significant	Significant impact on individuals, potentially affecting their well-being and daily routines.	Significant environmental impacts, such as pollution of waterways and air, that affects larger areas or populations.	Significant damage or loss to material possessions leading to inconveniences or substantial costs.	Noticeable impacts on the project group, potentially impacting the project timeline, resources and internal dynamic etc.	Noticeable impacts on the firm, affecting stakeholder relations and reputation.
4	Major	Substantial disruptions or changes to people's lives, possibly leading to noteworthy discomfort or stress.	Substantial damage to the environmental, resulting in significant loss of biodiversity, pollution or degradation of ecosystems.	Substantial loss or destruction of material possessions, resulting in significant financial loss.	Substantial impacts or setbacks for the project group, requiring large efforts and revisions to address.	Substantial impacts for the firm, potentially leading to significant financial and reputational damage.
5	Severe	Severe consequences for individuals, potentially resulting in physical harm, loss of lives or emotional damage.	Severe or irreversible harm to the environment that causes longterm ecological damage, loss of biodiversity or even ecological collapse.	Severe and irreparable damage to material possessions, causing extensive financial loss of valuable assets.	Severe impacts or failures within the project group, leading to project abandonment and/or dissolution.	Severe threats to the firm, potentially leading to bankruptcy or irreparable damage.

Tab. 3 assesses the degree of consequence, from 1 - Insignificant to 5 - severe. For a given incident, the consequence may depend on the underlying cause and can vary from people, environment, materialistic property, our project group, and project partner - TE.

**Level of probability:** Tab. 4 can be used to grade the probability of a given event occurring.

Table 4: RRR - Level of probability

Level of probability			
Level	Definition	Description	Number of occurrences
A	Rare	A "rare" probability rating suggests a low likelihood of the scenario occurring, with the possibility of it happening zero to one time within the project's life cycle from initiation to delivery.	0 - 1
B	Unlikely	An "unlikely" probability rating suggests a higher likelihood than rare but that the scenario is not expected to occur frequently. It allows for the possibility of the scenario happening two times within the project's life cycle from initiation to delivery.	2
C	Moderate	A "moderate" probability rating implies a moderate likelihood of occurrence, suggesting that the scenario may happen a few times but not excessively. It accommodates the possibility of the event occurring three times within the project's life cycle from initiation to delivery.	3
D	Likely	A "likely" probability rating indicates a relatively high likelihood of occurrence, suggesting that the scenario is expected to happen four times within the project's life cycle from initiation to delivery.	4
E	Almost certain	An "almost certain" probability rating suggests a very high likelihood of occurrence, indicating that the event is expected to happen five or more times throughout the project's life cycle from initiation to delivery.	>= 5

This is done by comparing it to the frequency of similar events. The probability is graded from "A" to "E", where "A" represents a low probability and "E" is a high probability. [26]

**Risk matrix:** Tab. 5 showcases a risk matrix, which combines Tab. 3 and Tab. 4. The risk

Table 5: RRR - Risk matrix

Risk matrix		Level of probability (Number of occurrences over the bachelor's period)				
		A (0 - 1)	B (2)	C (3)	D (4)	E (>= 5)
Level of consequence	1 (Insignificant)	L	L	L	L	M
	2 (Minor)	L	L	M	M	H
	3 (Significant)	M	M	H	H	H
	4 (Major)	M	H	H	H	H
	5 (Severe)	H	H	H	H	H

of an event is assessed by adding the degree of consequence and probability. Tab. 6 explains and defines how the risks are ranked.

*Level of risk:*

Table 6: RRR - Level of risk

<b>Level of risk</b>		
<b>Level</b>	<b>Name</b>	<b>Description</b>
L	Low	The occurrence of a scenario represents a low level risk. Mitigation measures are not necessary but can be considered.
M	Medium	The occurrence of a scenario represents a medium level risk. Mitigations measures should be warranted and considered.
H	High	The occurrence of a scenario represents a high level risk. Mitigations measures are required and must be implemented.

See App. E for a visualization of the RRR - reporting form.

## 4 Concept development

### 4.1 Concept selection

SG | LTR

During the concept development phase, a problem-solving team must research as many potential concepts as possible. Firstly, it will help the team to get a broader understanding of different options and what other teams have done in the past in varying applications. Secondly, this will ensure that the final product fully understands and meets the specific requirements. Thirdly, by exploring various robot types, the designers will gain insight into different technologies, software, materials, motors, gears, and sensors used in various applications. As a result, the team will be able to ensure that decisions in choosing hardware and software for the robot are right for its intended use. At the same time, the proposed concepts are technically feasible and implementational in the MACF. In addition, the team will be able to consider the cost of producing different types of robots by evaluating existing ones and making an informed decision regarding the budget constraints. Furthermore, evaluating efficiency and performance metrics is crucial to determine each type of robots unique characteristics. Understanding factors such as speed and accuracy will also help us select the most suitable design for the MACF. Besides the points mentioned above, it is important to consider the needs of the operators. This ensures that the robot is operator-friendly, meets safety demands, and is easy to maintain. In summary, researching different concepts enables us to make informed decisions, allowing us to choose correctly by excluding certain ideas and including others in our concept development.

#### 4.1.1 Excluded concepts

SG | LTR

We have considered nine potential concepts (See App. D) for a brief description and visualization of each robot concept), and we have chosen to exclude six of them as viable concepts for various reasons. Firstly, the Cartesian robot has several limitations that make it unsuitable for our purpose. They offer limited movement in the z-axis, and the mounting opportunities are not well suited for the MACF. In addition, the design of the robot will raise the overall design area that we are given if it reaches the desired workspace area and performs the pick and place function. Additionally, the mounting process is complex due to the specific design of MACF, which will add an unnecessary logistical challenge to the implementation. They are also not very user-friendly regarding service and maintenance. The robot must be taken apart, element by element, which is a time-demanding and costly process. Furthermore, even though these types of robots are suitable for repetitive tasks, such as packaging chips, they tend to operate at a slower speed than others, impacting productivity and efficiency in a fast-paced packing machine like the MACF. On the other hand, due to their simplified linear movements, programming these types of robots involves processes that are among the less advanced when considering various pick-and-place robots. This makes them significantly feasible for software engineers. However, the number of limitations and challenges outweigh this advantage, hence the Cartesian robot design is not considered as a viable option for our project. The informa-

tion contained in this paragraph and the section concerning the Gantry and Cartesian robots in App. D were derived from references: [27], [28] and [29]. The picture of the Gantry and Cartesian robot in App. D was derived from [28].

Secondly, the Gantry robots offer certain advantages but present several drawbacks, like the Cartesian robot, which renders them unsuitable for the MACF and our needs. They will also encounter a height problem and require more frequent maintenance if they are going to operate at the speeds needed to maintain high productivity. The operating motion is the shortest path from A to B to ensure productivity. Due to this, the wear on rails and other parts from the repeated start and stop intervals will be excessive. Additionally, Gantry bots also have a reduced range of motion in the z-axis and a challenging mounting process, primarily due to the 4-point mounting system required. This limits their adaptability inside the MACF even though it can be adapted to perform the task within the workspace. These robots can handle more weight than the Cartesian due to the 4-point mounting and 3-rail system but are best suited for tasks that involve repetitive operation in simpler linear movements. Therefore, the Gantry bots are unsuitable for our needs. [28]

Thirdly, we have the SCARA robots, which fall short of meeting our operational requirements. The robot is known for its high operating speed. Still, it is outscored by 30% by the Delta robot due to its reliance on linear movements, which impacts the overall productivity of the MACF. While they have high precision for finer work and offer 360 degrees of rotational movement, the robot will face limitations within the given design area and workspace. Particularly on the z-axis, which will constrain our operational flexibility. Regarding their mounting options, SCARA robots can be mounted using pedestals, floors, and walls, making them applicable for various work environment setups. However, due to the design of the MACF, where spatial orientation already poses challenges, we are left with the sides of the machine as the only apparent mounting option. However, considering the doors and windows as well, even the latter option becomes problematic. Despite the mentioned limitations, SCARA robots' reduced number of axes and smaller size results in fewer components and materials, thus making them considerably more affordable compared to other types of robots. However, the mentioned advantages do not outweigh the disadvantages. Hence, the production of a SCARA robot is excluded from further project work. The information contained in this paragraph and the section concerning the SCARA robots in App. D were gathered from [30]. The picture of the SCARA robot in App. D was derived from [31].

We looked at Cylindrical robots, which present several challenges that render them highly unsuitable for high-speed volume tasks like the one in MACF. They have a limited range of motion and tend to operate at slower paces due to their vertical axes and rotational movement. This hinders their ability to be efficient and handle tasks requiring rapid movements. Regarding maintenance and cost-effectiveness, the robots fewer parts and simple design de-

crease points of failure, potentially reducing the need for maintenance fees. Additionally, fewer motor components often result in lower power consumption, potentially reducing ongoing operational costs. However, certain parts of cylindrical robots, especially the rotary joints, might experience occasional wear and degradation. This could lead to excessive downtime, negating the previously mentioned fee-reducing effects of the structural design. This type of robots also faces challenges with implementation of CFRP and the interface between CFRP and Aluminum, which is a requirement from TE, due to their shape and design. This would also complicate the manufacturing process and may result in inconsistencies, structural weaknesses, or even the inability to produce a functional robot for the USNexpo. Given these challenges and limitations, Cylindrical robots are not considered suitable for further development. The information contained in this paragraph and the section concerning the Cylindrical robot in App. D were derived from [32]. The picture of the Cylindrical robot in App. D was derived from [33].

Another concept we have considered is the Polar/Spherical robots. These robots are highly robust but lack versatility and are primarily used for heavy-duty tasks such as automotive assembly lines and autocannons. They also have several limitations for our intended application. This includes a severely large footprint, limited range of motion in the z-direction, and too slow an operational speed to maintain the current productivity of the MACF. Additionally, the complex design associated with the robot contributes to increased costs and time-demanding maintenance, making them less economical than other types. Even though they excel in their specific areas, their limitations in speed, vertical range, and footprint, along with their typical use area, render them unsuitable for our application. The information contained in this paragraph and the section concerning the Polar or Spherical robot in App. D were derived from the following references: [34], [35], [36], and [37]. The picture of the Polar or Spherical robot in App. D was derived from [38].

Furthermore, we have considered Collaborative robots that offer some advantages but ultimately faces significant limitations for our needs. They have less lifting capacity compared to other industrial robots which is acceptable since chip packages dont weigh more than a hundred grams. Their design is complex, requiring a multitude of motors to achieve movement in all three axes, as well as advanced sensors to ensure safe operability. However, these robots encounter challenges regarding extensive safety regulations and certifications. These are statutory in ensuring operability, but achieving compliance with them is significantly time-consuming and costly. Collaborative robots are also designed to be a helping hand and work side by side with humans. This saves personnel from unnecessary loads by repetitive tasks, but since theyre not made to be fully automated it renders them unsuitable to perform the required task inside the MACF. Regarding versatility, they are highly flexible and can be adapted to diverse tasks, but because of their safety protocols the speed is still limited, ultimately constraining their potential productivity. With all this in mind, we viewed Collaborative robots as unsuitable for our project and excluded it from further development. The information contained in this

paragraph and the section concerning the Collaborative robot in App. D were derived from the following references: [39], [40] and [41]. The picture of the Collaborative robot in App. D was derived from [41].

#### 4.1.2 Included concepts

SG, AL | LTR

The concepts we have considered as more viable options for our project is the Delta robot, Duo-pod and articulated robot. Starting with the Delta robot, it is highly suitable for the task at hand, but does not have the heaviest lifting capacity due to its design and circular work area. The maximum capacity is reached at the center or mounting point when the load is evenly distributed across all three arms, but it decreases the further away the tool gets from the center point, due to the excess load exerted on the arm closest to the center. The robot is also relatively complex to assemble, requiring precise fittings for all components, including the active and passive arm, motors, and gears. Ensuring the correct length for the active and passive arms is crucial to optimize functionality and reach in the z-axis. This process is more intricate mechanically, than with other linear robot concepts, but fully feasible. Despite its complexity, the Delta robot is the fastest and most efficient, thanks to its 3-axis design that is mechanically assembled in parallel. This design allows for quick acceleration and movements, making it suitable for the MACF and tasks requiring speed and agility. Delta robots are usually mounted in the ceiling and can be designed to have a small footprint, enhancing their adaptability for locations with limited workspace like inside TEs MACF. Integrating them into existing setups is relatively straightforward due to their modular design. Additionally, the standardized linear placement of motors and gears at the top, perpendicular to the innermost joint to the active arm, makes it easier to maintain and do scheduled services on the mentioned parts. Furthermore, it is easier to fulfill the requirement for the use of CFRP in this type of robot, since it can be implemented in the active and passive arms, which are not encapsulated, ensuring a light but solid structure. Regarding the required programming, Delta robots are relatively complex compared to other linear robots, since the three passive arms would require one motor each that must be controlled in synchrony to execute the desired movements. This results in challenging feasibility for software engineers, but other than that, this type of robot is well-suited for our needs and easily implementable into the MACF. It is worth mentioning that the 4-axis variant is excluded from our consideration as the additional rotational axis at the tool hub is unnecessary for our purpose. The information contained in this paragraph and the section in App. D concerning the Delta robot was derived from [40], [42], [43] and [44]. The picture regarding the Delta robot in App. D was derived from [44].

Moving over to the Duo-pod, it shares similarities with the Delta robot in terms of speed, efficiency, and the simplicity of implementing and combining carbon fiber and aluminum. One notable limitation, however, compared to the Delta robot, is its inability to move along the y-axis. To address this, and enable movement across all three axes as required, rails could be incorporated into the design. While this is feasible, it adds an extra element for service

operators to consider. However, since movement along the y-axis is only necessary when the size of chip packages changes, the additional element would not require a significant increase in maintenance. By incorporating rails, the programming complexity is also reduced, as it bypasses the need to directly incorporate y-axis movement into the arm movements. Regarding footprint, implementing rails naturally increases the robot's design area, but it enhances service safety because the working areas of the robots would not overlap when two Duo-pods are placed on separate assembly lines within the MACF. Additionally, we have decided to exclude Duo-pods with configurations of three and four axes since the additional rotational axes at the tool hub are unnecessary for the intended application. By designing and developing a Duo-pod, we can maintain feasibility and still fulfill the essential requirements from TE. The information contained in this paragraph and the section in App. D concerning the Duo-pod was derived from [40], [42], [43] and [45]. The picture regarding the Duo-pod robot in App. D was derived from [45].

Concluding with the articulated robot, one significant and notable challenge with these robots is the complexity of programming involved, making it less feasible for data engineers. Advanced algorithms are required to control all joints simultaneously and ensure that the robot execute movements along the shortest path to maintain efficiency. Additionally, articulated robots tend to have a higher initial cost due to its advanced technology and increased number of joints and components. However, this initial investment can yield a higher ROI as these robots are significantly versatile and can be utilized for various tasks beyond those in the MACF. Articulated robots are also relatively easy to implement within the design area of MACF, where orientations and safety may pose challenges. They can be mounted on the ceiling, which, combined with multiple degrees of freedom, allows for flexible positioning and ensures operability in various working environments. In terms of precision, these robots excel, making them well suited for complex tasks that demand high accuracy, although this aspect is not highly emphasized by TE. Compared to the speed of the Delta and duo-pod, its less productive, and regarding the maintenance and feasibility, the encapsulation of motors and electrical components could result in invisible wear and tear and create difficulties during both development and service work. This is because the entire encapsulation and the area(s) of failure would have to be completely disassembled, decreasing production uptime and/or lengthening downtime. In terms of incorporating rails, it is also a necessity to meet TE requirements and ensure pick-and-place movements within the entire workspace. The information contained in this paragraph was derived from [46], [47], [40],[42] and,[43]. The information and picture contained in the section in App. D concerning the Articulated robot were derived from [46] and [47].

## 4.2 Pugh matrix

AK | LTR

After finding and researching nine different pick-and-place robots, the concepts had to be ranked in regard to the project requirements. For this, a Pugh matrix was chosen as the comparison model, also called a decision matrix. A Pugh matrix evaluates concepts by scoring each concept based on predetermined criteria, before comparing their total scores. The criteria are often weighted by a modifier (1-3) because some requirements are more critical or important than others. This way we can ensure a higher accuracy. see Tab. H

The matrix is built up of a set of columns: Column one is criteria developed in collaboration with TE: cost, assembly, robustness, accuracy, maintenance, workspace, system feasibility, speed, reliability, lifting capacity, and footprint, explained further in the Tab. 7. These criteria were weighted between 1 and 3 by TE. Here 1 is not important, 2 is preferred, and 3 is important. 7

Table 7: Criteria explanations for the ranking of scores.

Criteria explenation
Description
How much does it cost, the cheaper gives better score
How easy it is to assemble
how easy is it to operate
how robust is it? The more the better.
how accurate is it?
how easy is it to do maintanance
How large area does it operate in.
The system, complexity and feasability of the robot.
The speed of which the robot can operate.
How reliable is it to do a task without failiure.
How large of an area does it use? The less the better.

The next column is scoring, here the concepts are ranked by how well they fulfill the different criteria, again between 1 and 3. (1 is poor, 2 is neutral, and 3 is good). An explanation for the score is given in the comment column. The Weight and scoring column are multiplied to give a criteria rating. This column is then summed at the bottom for the total concept score. [48]I

A Pugh matrix generally compares the different concepts to one another. The main fall-pit when utilizing the Pugh model derives from already having favored a concept prior to the comparison. In this way, the matrix can be manipulated to return a favored result. To minimize the risk of the group's influence, the relevant concepts has been individually research and ranked prior to their comparisons. 8

Table 8: Combined rating - Pugh Matrix

<b>Combined rating - Pugh Matrix</b>			
<b>Criteria</b>	<b>Type of robot</b>		
	<b>Two directional articulated arm on rails</b>	<b>Delta</b>	<b>Duopod w/ rails</b>
<b>Cost</b>	3	6	6
<b>Assembly</b>	2	2	4
<b>Robust</b>	6	4	6
<b>Accuracy</b>	3	3	3
<b>Maintenance</b>	2	4	2
<b>Workspace</b>	3	2	3
<b>System feasibility (team)</b>	3	6	6
<b>Speed</b>	6	9	9
<b>Reliability</b>	6	6	9
<b>Lifting capacity</b>	6	2	4
<b>Footprint</b>	6	6	3
<b>Total</b>	46	50	55

#### 4.2.1 Articulated robot on rails

LTR | AK

The cost criterion, with a significant weight of 3, plays a crucial role in evaluating the articulated robot arm on rails. It scored 1, indicating a relatively high expense due to the requirement of 3 motors in the arm and an additional motor for the rail. This suggests that the design's complexity leads to considerable costs, which may need to be optimized for more cost-effective deployment.

Assembly is weighted at 2 and scores a low 1, indicating that the assembly process is complex due to the high motor count and three joints and rails. This complexity could potentially impact the ease of manufacturing and assembly, posing challenges for scaling up production.

The robot's robustness is assigned with a weight of 2. It scores a 3, demonstrating a robust mechanical design and a single strong arm that contribute to the robot's overall sturdiness and resistance to wear, an essential factor for long-term operation and durability.

Accuracy is less weighted at 1 but scores a 3, suggesting that while the robot's design, with many joints offering three degrees of freedom, theoretically allows for high precision, achieving this from a coding standpoint may be more challenging, potentially impacting the robot's ability to perform tasks that require exact positioning.

Maintenance weighs 2 and scores a 1, indicating that more motors and the corresponding high torque and load on the top joint could lead to more frequent maintenance requirements. This highlights a concern for long-term upkeep and reliability.

The workspace is weighted at 1 and scores a 3. The robot can perform movements within the desired workspace, with the arm responsible for the X- and Z-directions and the rail for the Y-direction, suggesting an effective use of space within its operational parameters.

System feasibility is weighted at 3 and scores a 1, implying that while it is not impossible to design and produce such a system, the number of active joints increases the complexity, potentially requiring more specialized skills and resources for development and integration.

Regarding speed, with a weight of 3 and a score of 2, the robot is somewhat hindered by the top motor, which is responsible for a majority of the motion, resulting in slower operational speed compared to systems with more distributed or optimized motor arrangements.

Reliability is critically weighted at 3 and scores a 2. The encapsulation of mechanical and electrical components within the movable parts of the arm may decrease reliability over time, and the rail's role in linear tasks may result in mild wear, indicating a need for durable design

choices to maintain performance over extended periods.

The robot's lifting capacity is assigned a weight of 2 and scores a 3. Articulated robots have a mechanical advantage due to their jointed arms, which allow them to handle high loads, indicating a relatively high strength when performing lifting tasks.

Finally, the footprint weights 3. It scores a 2, reflecting that while the robot has a low footprint with one arm, including rails, it occupies more space.

In conclusion, with a total score of 46, the two-directional articulated robot arm on rails shows potential in robustness, workspace efficiency, and lifting capacity. However, it faces notable challenges in cost, assembly complexity, maintenance demands, and system feasibility that should be addressed to enhance its applicability in industrial settings.

#### 4.2.2 Delta robot

**LTR | AK**

The cost factor has been deemed significantly necessary with a weight of 3 and registers a moderate score of 2. This implies that the Delta robot's design, which includes three motors and a carbon fiber composition, is more expensive due to its complexity and the number of parts. However, cost efficiency could still be optimized with further design refinements.

Assembly is another crucial aspect, weighted at 2, but it scores the lowest at 1. The difficulty in assembling many small parts indicates potential challenges in manufacturing scalability and the need for skilled labor, which could impact the throughput and practicality of this robot in production environments.

Robustness and maintenance, with a weight of 2 and score of 2, respectively, indicate a balance between stability and wear issues associated with numerous joints and moving parts. While the robot's structure is firmly anchored, suggesting good stability, maintenance could be troublesome due to its intricate and delicate design.

Although accuracy is less weighted at 1, it scores a 3. Delta robots are known for their agility and movement in three axes, reinforcing their suitability for tasks requiring high levels of precision.

Workspace utilization is considered less critical, with a weight of 1 and a score of 2. The evaluation points out that the robot's arms consume a considerable area during operation, which could limit its application in confined spaces.

System feasibility is heavily weighted at 3 but scores only a 2. This reflects the complexity of coding and mechanical design, which diminishes the system's feasibility in a team setting,

possibly alluding to the need for specialized skills and extended development time.

In terms of speed, the robot excels with a score of 3, matched with a high weight of 3, benefiting from a lightweight arm design that facilitates rapid maneuvering, a crucial feature for high-speed operations.

Reliability, another critical criterion weighing 3, scores a moderate 2. While the robot's design promotes less wear, which enhances reliability, the presence of many small components that could fail tempers this advantage, suggesting a need for improved design robustness.

Lifting capacity has a significant weight of 3. However, it scores the lowest at 1, underscoring a notable limitation in the robot's ability to handle heavy loads, a considerable constraint for material handling applications.

Lastly, the robot's footprint, crucial with a weight of 3, achieves a score of 2. The evaluation indicates that while the robot's arms require space to operate, its overall space consumption is moderated by an overhead mounting design, making it suitable for specific constrained environments.

Overall, the Delta robot scores 50, which suggests that while it demonstrates excellent speed and accuracy, its assembly, lifting capacity, and maintenance can be improved.

#### 4.2.3 Duo robot on rails

LTR | AK

In evaluating the Duopod on rails, the cost factor is assigned a significant weight of 3 and has achieved a moderate score of 2. This suggests that the design, which includes two motors for the arm and one for the rail, is moderately expensive. However, since the rail motor does not need to be as powerful as the arm's, there is potential for cost optimization.

Assembly weighs 2 and scores 2, indicating that assembling the total system may be demanding due to the requirement of mounting rails and assembling the robot itself. This points towards a moderate complexity in the assembly process that could affect the production scale and the need for qualified assembly labor.

The Duopod's robustness is weighted at 2. It scores a 3, demonstrating a robust biaxial design that provides solid stability and potentially less sway due to a solid connection interface rail/arm. This suggests a good balance between strength and the possible maintenance cost.

Accuracy is weighted lower at 1 but scores a 3. The Duopod's two-axis design makes it relatively simple to achieve high accuracy, which could be adequate for specific applications requiring moderate precision.

With a weight of 1, maintenance scores a low 1, signaling that while the biaxial arm design could simplify maintenance, the rail connection could present challenges, depending on the design complexity.

Workspace utilization is considered less critical with a weight of 1. It scores a 3, reflecting Duopod's ability to efficiently work within the defined space, a positive aspect for operations in limited areas.

System feasibility has a heavy weight of 3 and a score of 2, indicating that while the software, electrical, and mechanical design for a two-axis system simplifies development, challenges may still reduce overall system feasibility.

With a weight of 3 and a score of 3 in terms of speed, the Duopod excels, thanks to the top placement of the motors, which allows for fast and agile movement. This is particularly advantageous for tasks that require quick movement.

Reliability is critically weighted at 3 and scores a high 3. With few moving parts and the rail performing a linear task, the system's design promises high reliability, suggesting infrequent wear and maintenance.

The lifting capacity has a notable weight of 2. However, it scores a 2, indicating that the Duopod may not offer optimal lifting capabilities due to arm length and robustness, which could limit its application in heavy lifting tasks. Lastly, the footprint is weighted at 3. It scores a 1, meaning the Duopod has a sizeable spatial requirement due to its operational range, which could be a limitation in environments where space conservation is crucial.

In summary, the Duopod on rails obtains a total score of 55, reflecting its high speed and moderate accuracy performance, with significant considerations for its assembly and footprint. These aspects must be carefully considered when integrating the Duopod into various operational contexts, especially where space and efficiency are essential.

#### 4.2.4 Comparison

LTR | AK

By comparing the results obtained by the Pugh matrix ranking, we have determined the duopod on rails to be the best result. Although there is no significant difference in the total scores, they were the last choices we landed on and can all be considered reasonable solutions. The scores serve as a guide for their capabilities and differences rather than indicating the correct choice, and a significant score difference is not expected. Though TE has given the weight in the matrix, it is important to note that small changes in the weighting could alter the outcome of the rankings. It is, therefore, essential to reconsider the most critical demands. While

all three alternatives are feasible for the task, our time for the project shall be utilized efficiently.

The articulated robot has multiple axes in the form of joints in the single arm, making it complex to design and test due to the electrical and mechanical components that must be constructed for realistic software testing. The delta robot has a third axis, making it complex to program because three actuators must work together. This makes it less feasible to complete as it requires more testing. On the other hand, the duopod on rails has a simple two-axis design, which is more feasible to complete. Therefore, the duopod on rails remains the best choice due to its simple and practical design.

## 5 Junior development

### 5.1 Introduction

LTR | SG

Before beginning the development of the final product, our team decided to construct an MVP to test and demonstrate the core functionalities of our exhibition model, an iteration we refer to as the 'Junior.' This preliminary phase allows us to test and validate various components and features, discerning what is effective and what is not. The insights obtained from this stage are essential as they will guide us in designing and preventing potential obstacles in developing our finalized product, 'Senior.'

### 5.2 Mechanical design

#### 5.2.1 Design space

AK | MO

For the design space development there had to be further research in regards to the robots environment. Utilizing the MACF model from TE, some main points of interest was extracted. This being: Top frame, length of conveyor belt, height of conveyor, depth of stacker as well as the door opening width. The lowest reference being the bottom of stacker while the top was determined by the frame height (excluding added top frame). The robot shall be roof mounted without colliding into roof and deliver chip bags to the bottom of stacker. Therefore, the height: Bottom stacker to top frame = 1655mm. The utmost left point at left side doorframe and right at the end of stacker. The effector shall move from the top of the conveyor belt to the inside of the stacker. Hence, the width: Left doorframe to end of stacker = 1150mm. All essential components are located in their respective halves of the MACF, therefore the depth: Half the MACF = 1100mm. For advantageous scaling, a 1:2 model was chosen. As the dimensions had considerable space for adjustments this was approximated to 800x500x500. From this, it was evident that the design space had a rectangular shape and that the original cube was impractical, therefore the dimensions were altered to 500x500x600mm.

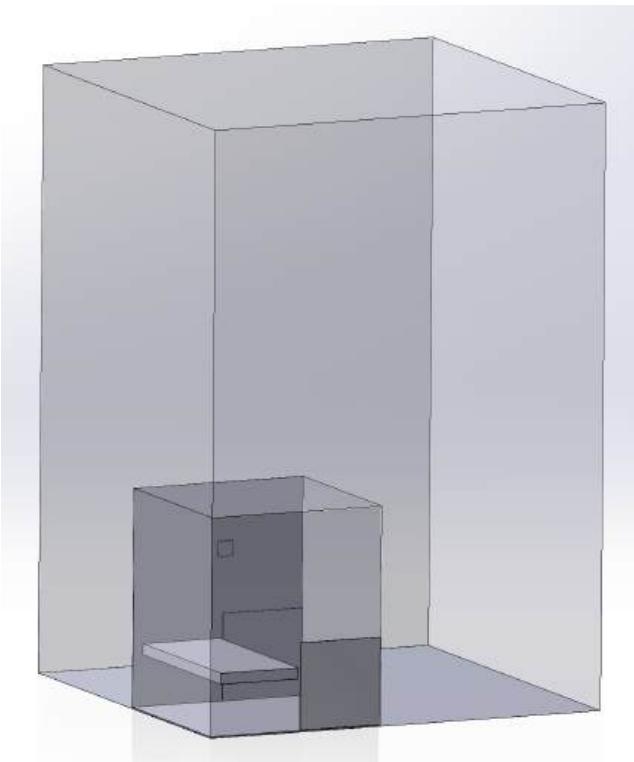


Figure 5: Original and downscaled design space

### 5.2.2 Concept modeling

AK | MO

Two of the concepts researched in the concept selections was modelled in SolidWorks to visualize the functionality and behavior within the required workspace. These are modeled using simplified components and focused on arm lengths and joint movement. The first one was the initially considered delta robot:

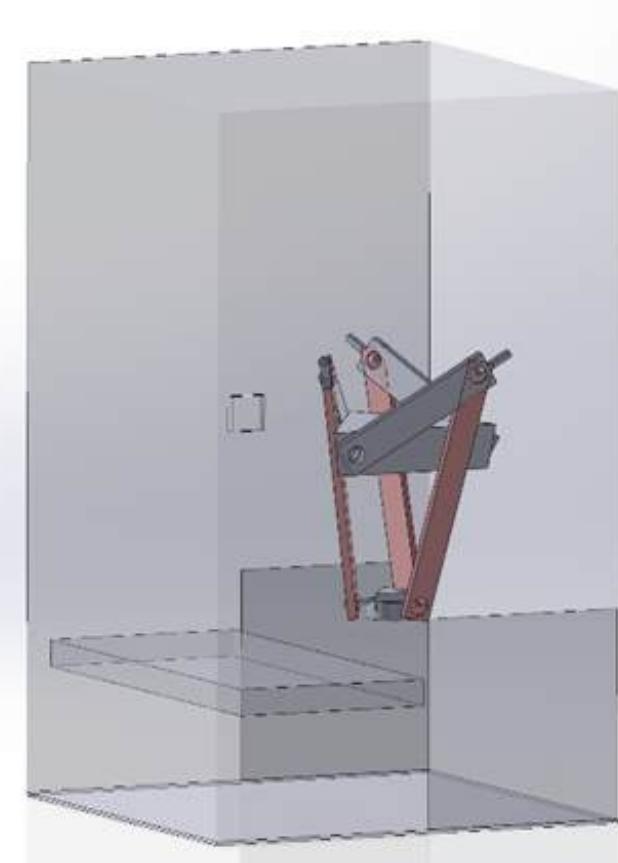


Figure 6: Delta concept

The delta robot features three active arms generally motorized by servo motors, fixed at the main hub. At the elbow joint, the passive arms are free to rotate, extending downward to the end effector where the tool is mounted. Therefore, the modeling focused on these main components and their respective joints. Active arms are mounted parallel to the hub sides and ball joints at the elbow point. End effector joints also feature the same ball joints as mentioned, these allow for free movement. On some models this agility is provided by two hinges, but for time conservation and simplicity the ball joints were chosen. Lastly, the model was imported to the design space where different joint configurations were simulated.

Duopod concept:



Figure 7: Duopod concept

A Duopod is a two-degrees-of-freedom robot used for linear pick and place operations. Similarly to the delta robot, this design utilizes an active top arm with a passive lower arm, which again manipulates the end effector. Experimenting with the different joint configurations can be summed to: Short Active with longer passive arms, restricted movement in Z-axis (height) as well as X-axis. Long active with short passive, is inoperable without extra motors in elbow joints. The optimal length relations for active:passive, were approximately 1:2 +/- 10 percent depending on the task and workspace. On the concept model above: Active = 110mm and passive = 250mm (0.44:1).

### 5.2.3 Junior modeling

AK | MO

Upon Junior's development, different Duopod versions have been further researched. As a continuation of concept modeling, the focus continued on the joints and arm dimensions. The arms were projected for laser-cutting, so the thickness was set to 3mm. When the concept model performed as desired, the detailing process was initiated. This included modular design for later adjustments of the arms, hub, and easy release of the motorized axles. While moving through the workspace it was inevitable that the tool rotated with the arms. Therefore, a stabilizer was constructed, mounted at the base, to the left elbow, and down to the tool hub. A problem encountered was regarding the placement of the robot, which was initially placed in the middle of the design space. This resulted in the robot stretching significantly off the

center line when moving down into the stacker. This movement proved to be impractical for the robot, especially due to the restriction caused by the stabilizer. As a solution, the robot's position was adjusted such that the center line was positioned above the center of the stacker. [49]

### **Frame assembly: MO | AK**

#### ***Introduction:***

A requirement from TE was to have the prototype movable and used for exhibition. A frame was designed with aluminium profiles and fastening technology from Item, within the design space of 500x500x600mm. The frame was intended to be used in the Junior Assembly and later in the Senior, but due to not meeting the weight requirement, a new frame was designed for the Senior.

#### ***Interfaces:***

##### ***Fastening Hub Assembly:***

The Fastening Hub Assembly is fastened to the frame by six hubs. Each hub contains of two parts that are connected with Ø6mm bolts surrounding two Item profiles on the top.

#### ***Design and development:***

Using Item profiles and fastening technology makes the frame modular, and making changes to the size of the frame or position of the perpendicular profiles on top is possible. Item profiles with the dimensions 40x40mm was chosen, on the background of TE often having this in stock. The frame was designed to have outer dimensions 480x480x590mm, not maximizing the design space and facilitating the implementation of walls in a later design. The total length of aluminium profiles needed for the design was 6200mm, and unfortunately the specific weight of the profile was not considered in the design. A 40x40mm profile with a length of 6200mm and a specific weight of 1,7kg/m results in a total mass of 10.6kg. A requirement for the whole prototype was not to exceed a weight limit of 10kg. The realization of this requirement not being met happened after parts were ordered and the frame was assembled. The requirement was assessed and changed for the Senior, but designing and building a new frame was decided. The frame for the Senior is reviewed in chapter 6 Senior Development.

#### ***Connections:***

The Item profiles have slots on all sides, and a T-slot nut is placed in the line connecting corner brackets by tightening a bolt to the t-slot nut. In total, 20 corner brackets are used to connect the 14 Item profiles of different lengths. The four corners in the bottom have cube

connectors, which were intended to be used as a type of fastening part and secure the frame to a table.

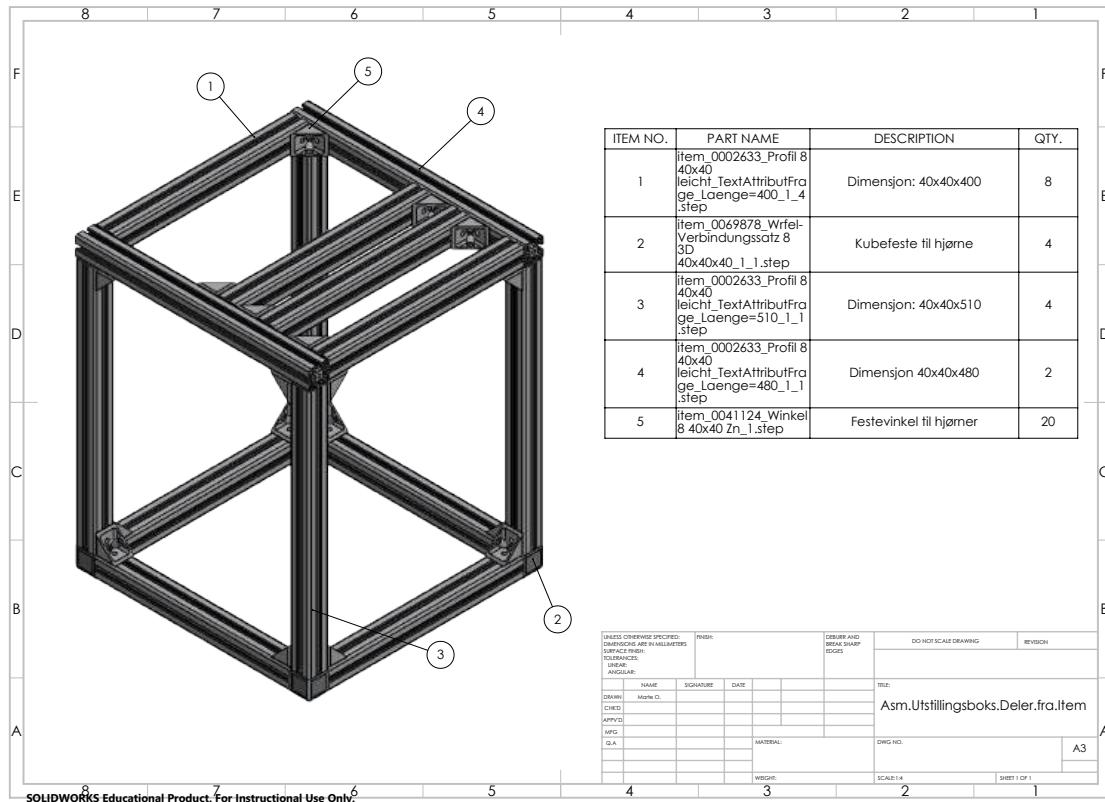
***Bill of materials and parts ordered:***


Figure 8: Bill of materials for Junior Assembly

The bill of materials is showcasing parts used in the assembly, not included are t-slot nuts(8 line) and M6 bolts.

Artikkel nr.	Navn	Lengde(mm)/spesifikasjon	Antall	Link	Beskrivelse	Bilde av Assembly	Bestillingsdato/initialer bestiller	Est. leveringstid	Mottatt
0.026.33	Profile 8 40x40	400	8	<a href="https://www.item24.com/en-de/profille-8-40x40-light-normat-2632?languageID=1&amp;languageCode=en-GB&amp;CHN=INIT&amp;categoryID=1032">https://www.item24.com/en-de/profille-8-40x40-light-normat-2632?languageID=1&amp;languageCode=en-GB&amp;CHN=INIT&amp;categoryID=1032</a>	Itemprofiler til utstillingsramme. Totalt lengde 6200 mm		21.02.2024/MKW(MO)	01.03.2024	01.03.2024
0.026.33	Profile 8 40x40	480	2	<a href="https://www.item24.com/en-de/profille-8-40x40-light-normat-2632?languageID=1&amp;languageCode=en-GB&amp;CHN=INIT&amp;categoryID=1032">https://www.item24.com/en-de/profille-8-40x40-light-normat-2632?languageID=1&amp;languageCode=en-GB&amp;CHN=INIT&amp;categoryID=1032</a>	Angle Bracket, hjørnefeste utstillingsramme		21.02.2024/MKW(MO)	01.03.2024	01.03.2024
0.411.24	Angle Bracket Zn	Line: 8	24	<a href="https://www.item24.com/en-de/angle-bracket-zn-line-8-40x40-in-white-aluminium-similar-to-al-2020-41124?languageID=1&amp;languageCode=en-GB&amp;CHN=INIT&amp;categoryID=1032">https://www.item24.com/en-de/angle-bracket-zn-line-8-40x40-in-white-aluminium-similar-to-al-2020-41124?languageID=1&amp;languageCode=en-GB&amp;CHN=INIT&amp;categoryID=1032</a>	Cap til å sette på Angle Bracket		21.02.2024/MKW(MO)	01.03.2024	01.03.2024
0.411.26	Angle Bracket Cap	Line: 8	24	<a href="https://www.item24.com/en-de/angle-bracket-cap-8-40x40-blade-41126">https://www.item24.com/en-de/angle-bracket-cap-8-40x40-blade-41126</a>	Feste i hjørnene nede for videre festmekanisme til bord		21.02.2024/MKW(MO)	01.03.2024	01.03.2024
0.688.78	Cube connection set	8 3D	4	<a href="https://www.item24.com/en-de/cube-fastening-set-8-3d-40x40-40x480-78">https://www.item24.com/en-de/cube-fastening-set-8-3d-40x40-40x480-78</a>	Feste mellom Angle Bracket og Itemprofiler		21.02.2024/MKW	01.03.2024	01.03.2024
0.480.50	T-Slot Nut St	Line: 8	100	<a href="https://www.item24.com/en-de/t-slot-nut-st-8-8-mm-height-gmc-threaded-40050">https://www.item24.com/en-de/t-slot-nut-st-8-8-mm-height-gmc-threaded-40050</a>			21.02.2024/MKW	01.03.2024	01.03.2024

Figure 9: Order placement to Item including article no., name, specification, link, description, order date, estimated delivery date and the arrival date.

As seen in the table above, the delivery arrived on time, and the 6m Item profile was cut in lengths of 7x400mm, 2x480mm, and 4x510mm. Since a length of 6200mm was needed, an additional 1x400mm profile was planned to be taken from TEs inventory. 4x510mm and 4x400 mm were threaded for fastening the cube connection.

### Fastening Hub Assembly: MO | AK

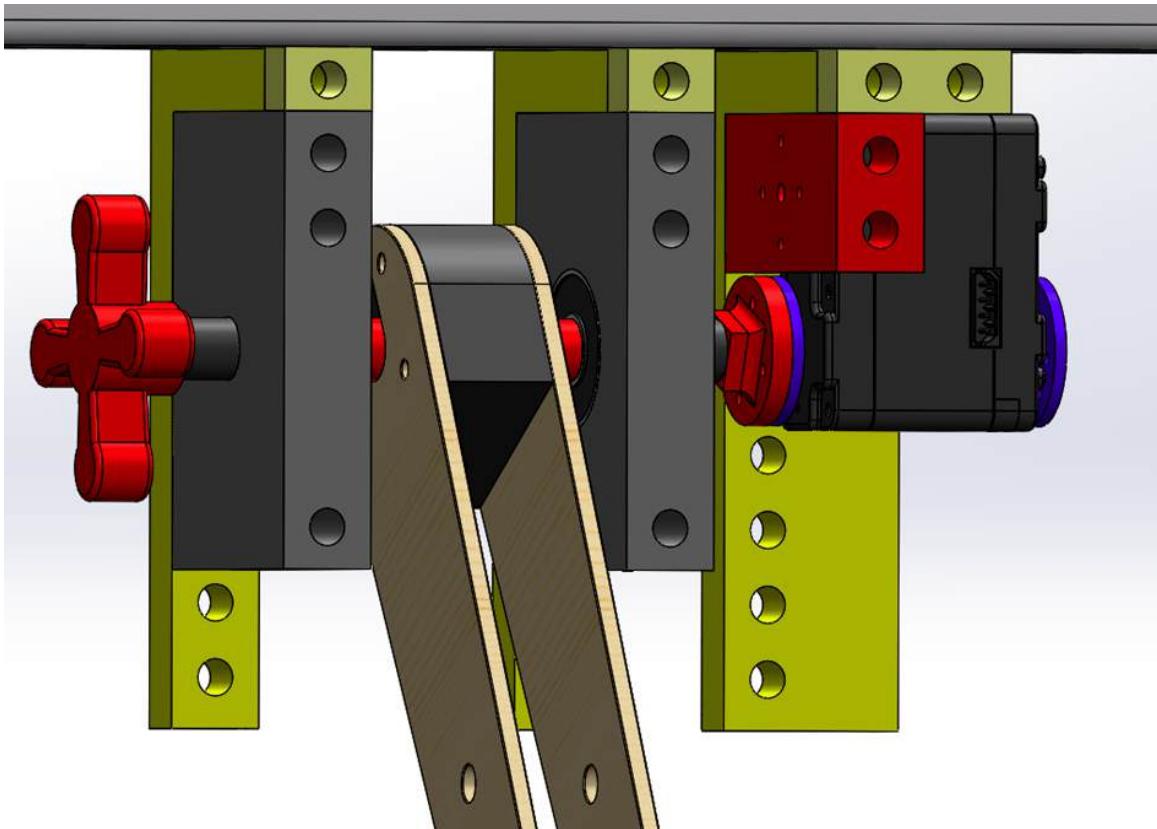


Figure 10: Half of the Fastening hub assembly

The fastening hub has an interface with the frame, where each of the six hubs contains of two parts that are connected with Ø6mm bolts surrounding two Item profiles on the top. There is an interface to the Arm assembly where it is connected to the arms by a modular axle. The axle consist of multiple parts and is press fitted together. The axle is connected to a motor adapter at one end, and a handle for manual rotation in the other. Holding up the axle are two ball bearings which are press-fitted to each hub. Besides the ball bearings, bolts and the motor, 3D printed parts are produced using FDM additive manufacturing, printed with a Bambu Lab X1 Carbon. The filament selected is PLA, and is printed with 0.2mm dynamic layer height and low infill to reduce print time.

### Arm assembly: AK | MO

#### *Introduction:*

The arm assembly will provide the robot with the desired work area. During the concept modeling and experimenting with the joint relations we arrived at lengths fulfilling the requirements. The active arms are powered by a motorized shaft utilizing two Lynx motion HS-1 smart servos. Passive arms receive no power but are manipulated by the active arms and shared end effector. The shaft, adapter, and spacer are 3D-printed, while the arm plates are laser cut. M3

and M6 Stainless steel fasteners are used at all connections.

### ***Interfaces:***

#### *Axle:*

The axle and arms are connected by an PLA adapter. The adapter features four Ø3mm holes as well as a 6.2mm squared hole in the center. The arms are connected at each side using 3mm screws, while the modular axle is inserted into the square providing rotational transfer.

#### *Effector:*

The arms are connected to the effector using M6 bolts, while the effector itself works as a spacer, fastened with a nut.

#### *Stabilizer:*

The stabilizer utilizes the 6mm elbow bolt as a hinge pin. At the stabilizer side, the bolt is longer and the triangle is spaced out using a PLA cylindrical spacer.

### ***Connections:***

Arms are connected and spaced by the PLA adapter. The adapter features four Ø3mm holes, in addition to a 6.2mm square hole in the center. the arms are mounted at each side using M3 bolts, while the modular axle is inserted into the square opening, providing a rotational fixture. At the elbows, M6 bolts function as hinge pins in the Ø6mm holes, four at active and five at passive. Cylindrical spacers printed from PLA provided the arms with an 18mm width at passive and 24mm at active. The spacers are fastened using 6mm bolts inserted at the front and matching nuts pressing the arms together.

### ***Design and development:***

#### *Active arm:*

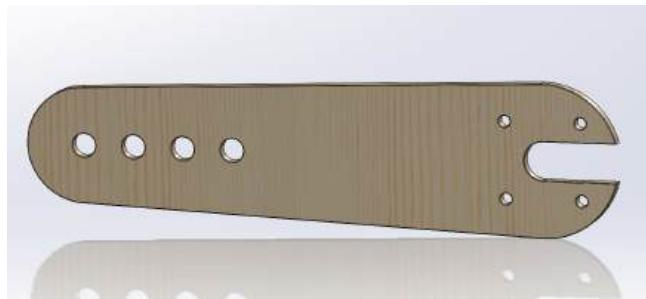


Figure 11: Active arm

The Junior active arm design has been derived from the concept modeling, with added possibilities for adjustment. Given the pressed time frame, laser cutting was the utilized manufacturing method for a rapid prototyping process. Using the concept's limb lengths, a design for the outline was formed, featuring rounded ends tapering towards the elbow point. As adjustability was of importance for data and electro, there was added adjustment Ø6mm holes at the elbow ends of the arms. For mounting to the axle adapter there are four smaller Ø3mm holes and a Ø10mm slit for the axle.

*Passive arm:*

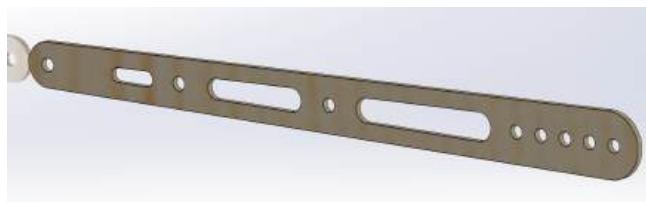


Figure 12: Passive arm

The Junior Passive arm design has also been derived from the concept modeling, with added possibilities for adjustment. laser cutting was also utilized for the passive arms. Utilizing the concept model's limb lengths, a design for the outline was formed, featuring rounded ends tapering towards the elbow point. As adjustability was important for data and electro, there was added adjustment Ø6mm holes at the elbow end of the arms. The two Ø6 towards the center are used for spacer fastening. At last, the skeletal design holes were added for mass reduction and esthetics.

*Adapter:*

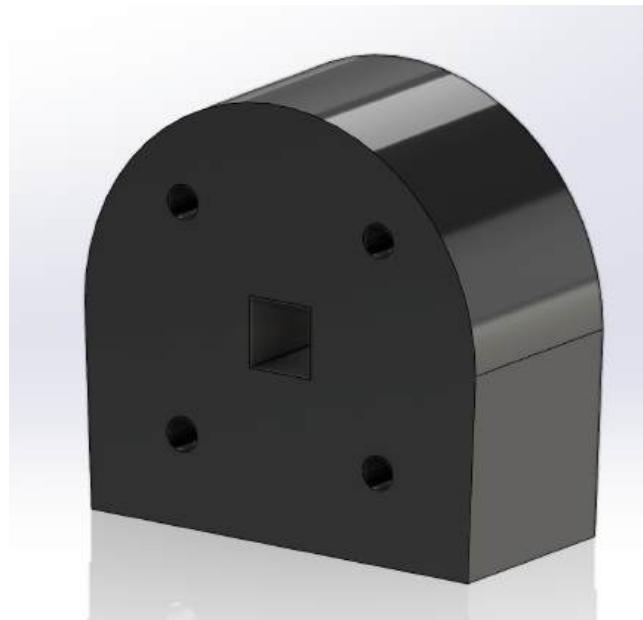


Figure 13: Axle adapter

For the transfer of rotational movement, it was essential for an adapter to function as a locking mechanism. This was ensured by a square mounting hole 6.2mm, for the modular axle. This is an excellent example on production-oriented design, were the applied 3D-printer had a tolerance offset of -0.2mm. four 3mm holes was added for even stress distribution in the MDF arms and PLA adapter, therefore also working as a spacer for the active arms.

*Spacers:*

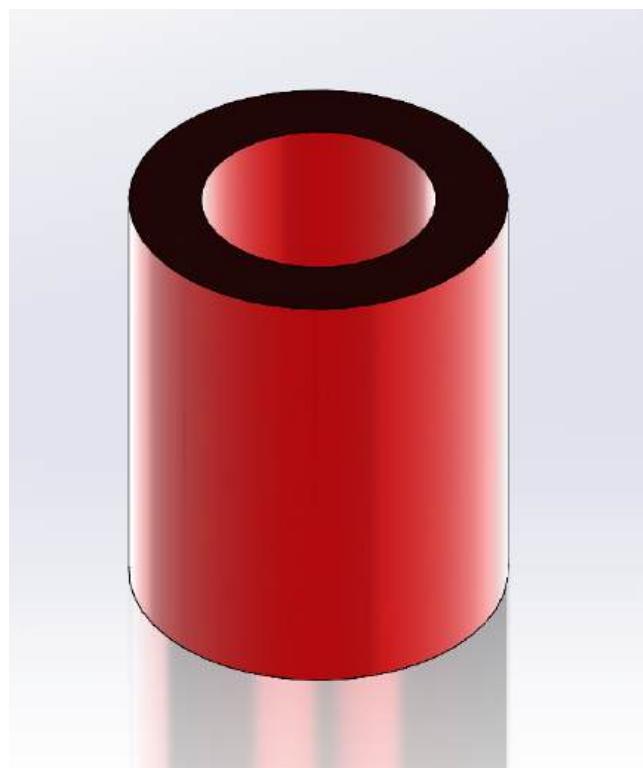


Figure 14: 6mm Arm spacer

The cylindrical elbow and passive arm spacers are also made of PLA and 3d-printed, featuring the same 0.2mm offset as the adapter.

### ***Parts ordered:***

Bolts: These are ordered from MiSUMi. Further details are found in Bill of Materials. [50]

### ***Parts produced:***

Spacers and Adapter:

3D printed parts are produced using FDM additive manufacturing, printed with a Bambu Lab X1 Carbon. The filament selected is PLA and is printed with 0.2mm dynamic layer height and low infill to reduce print time.

Laser cut parts:

Arms are laser cut using the Epilog fusion Engraver available at USN Krona.

### **Stabilizer assembly: AK | MO**

#### ***Introduction:***

The stabilizer's task is to maintain the tools horizontal position over the conveyor belt and in stacker at the bottom position. This field of study is called parallel kinematics which is used to calculate the correct length for arms and stabilizing rods by employing the correct relations between the components. This includes the stabilizer triangle which commonly has to have the same triangular proportions as the connections at the fastening hub and tool hub. While this is the case, our model is based primarily on what was suitable for our use during the Junior design process. Given that the stabilizer performed the desired movement, only minuscule changes have been implemented. Given more time, this would be one of the primary points for further research.

#### ***Design and development:***

*Connector arms:*

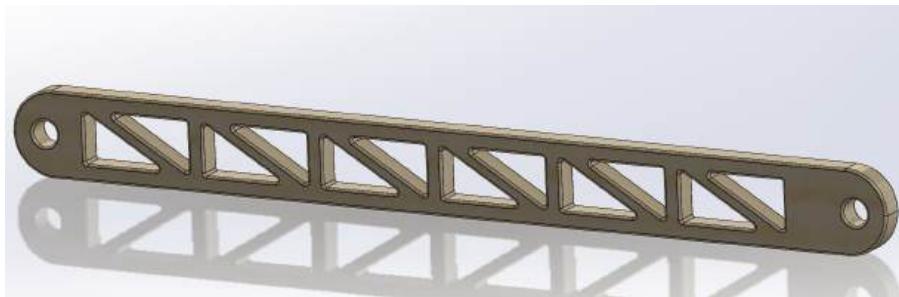


Figure 15: Short stabilizer arm



Figure 16: Long stabilizer arm

For rapid prototyping, the connector arms were both designed for laser cutting. Lengths are 295mm for the long arm and 110mm for the short, both featuring rounded ends with matching concentric Ø3mm holes. These 3mm holes are for the M3 connections on the Triangle and hub pin, but with direct contact with the cylindrical spacers. For design and mass conservation the triangular pattern was implemented.

#### *Stabilizer triangle:*

The most important feature of the triangle is the 35mm distance from elbow joint to the Ø3mm connector holes. This maintains the relation between the kinetic chain of the arms to the proximal stabilizer. [51]. The distance from main Ø6mm to connector Ø3mm is adjusted to 35mm, equal to axle - fastening hub pin. In hindsight there is a miss on the effector design where the adjacent side of the triangle (arm mount tool connector pin) should be 35mm. As of now, this is the length of the hypotenuse. Lastly, an extra design cut was removed from the center for mass reduction and esthetics.

#### **Parts ordered:**

#### *Bolts:*

These are ordered from MiSUMi. Further details are found in Bill of Materials. [50]

#### **Parts produced:**

#### *Spacers:*

3D printed parts are produced using FDM additive manufacturing, printed with a Bambu

Lab X1 Carbon. The filament selected is PLA, and is printed with 0.2mm dynamic layer height and low infill to reduce print time.

*Triangle and Connector arms:*

The triangle and connector arms are laser cut using the Epilog fusion Engraver available at USN Krona.

***Junior effector:***

The junior effector was a simplistic design made only for providing the initial connecting points for the arms as well with stabilizing. The design was made for 3d printing, for rapid prototyping the effector is made with a flat base and rectangular top. This is the presentation 1 model, it did not feature the correct holes and was later replaced.

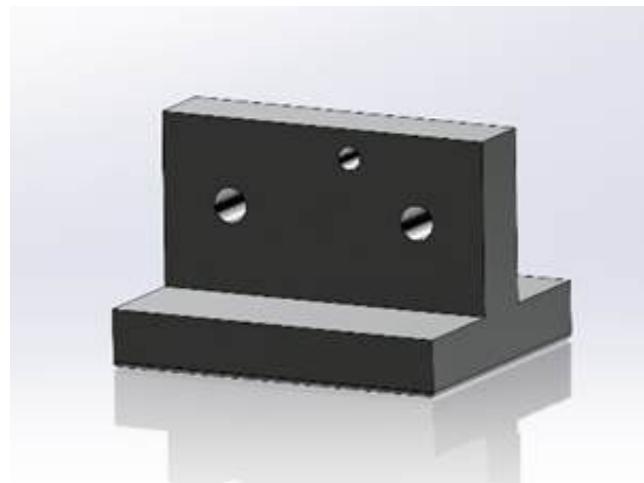


Figure 17: Effector

## 5.3 Electrical design

LTR | SG

To properly design an effective electrical design for the Junior, it is essential to define the system to ensure every aspect of the system is met. Since Junior is the MVP of the final display model, it is not necessarily a display model that comes across different environments and shall only visualize the most necessary functionalities of the system. This includes motor movement, communication between the controller and motors, and power to all electrical components.

### 5.3.1 Junior design

LTR | SG

The electrical parts for the Junior phase are constructed to show the Junior's primary purpose of testing and revaluation. Therefore, we have opted for a simple design using extension cables for the included power supplies for the powering of the computer and a communication adapter board for the motors, and no more. This is due to the importance of the software team's

reliability on a finished design, as well as testing modularity being the main importance of this model, not the aesthetics.

### 5.3.2 Motors

LTR | SG

When choosing the motors, we found smart servos to be the best choice as servos are the solution TE already uses, and they can communicate with the controller bidirectionally. This functionality will help us create a more accurate system as it will help the servo know in which position it is at all times. Motors like the HS1 from Lynxmotion are typically used for small-scale robot projects like ours. The HS1 motor is a quick motor with a dynamic torque capacity of 1.6 kg-cm and can rotate 360 degrees at high speeds [52]. The HS1 can easily be connected to Arduino, Raspberry Pi, and other controllers by using the LSS-ADA adapter board, which is also mountable to the controllers. A stepper motor could, however, be used to move the rails. Still, due to the functionality of serial coupling between the motors as well as the easy click-in functionality of the cables between the adapter board and motors of the HS1, we have decided to use the same motors [53].

## 5.4 Configuring the fundamentals for software development

### 5.4.1 Introduction

TM | AL

Building a prototype was beneficial for us on software, it allowed us to start development, integration and testing early. This has great benefits as it helps identify issues related to different components working together, which might not be evident in isolated testing. Researching these problems during development and not all at once by the end of the project also keeps us in line with SCRUM principles like iterative and incremental work.

During the Junior phase we worked on multiple aspects, this includes establishing communication with the Raspberry Pi, setting up a software workflow environment with ROS installed as well as researching ROS.

During this phase we also set up the servo motors ensuring communication with the PI, and worked on aspects not finalized until the *Senior* phase. The latter includes developing early iteration of architectural design, defining the user interface, starting definition and development of specific functionality. Further descriptions and details regarding these aspects can be found in the *Senior* development chapter.

### 5.4.2 Choosing a coding framework

TM | SG

When choosing a coding framework for our project, there were many aspects to consider. The framework needed to be mature, stable, well documented, yet flexible, preferably open-source. With this project we are not trying to reinvent the wheel, but rather develop a concept that solves TEs problem and creating an exhibition model showcasing the concept. Therefore, we were also looking for a framework with tools for handling math and controlling a robot arm. We want to simplify the process of completing the project while having full control over the system.

After searching the web for frameworks we found ROS, it provides a flexible and modular framework for writing robot control software. It's not an operating system in the traditional sense but rather a collection of software frameworks and libraries designed to simplify creating complex and robust robot behavior. In ROS and ROS functionality is split into nodes, they have various ways of communicating, where topics are the simplest and easiest to integrate. Topics allow a node to publish a message to a topic, all nodes subscribed to this topic will receive the message.

ROS provides a standardized way to integrate various hardware components. It offers interfaces for sensors and actuators, which means developers can easily add or swap components without extensively rewriting the software. ROS also integrates with the powerful simulation tool Gazebo, allowing developers to simulate their robots in a 3D environment. This allows for virtual testing algorithms and hardware configurations before deploying them on real robots, saving time and reducing the risk of damage. Libraries such as MoveIt offer motion planning designed for robotic arms. It handles the math and other complexities of path planning, collision detection, and trajectory optimization, simplifying the development. [54] [55] [55]

The real-world application of ROS extends beyond educational projects. It has been employed in diverse, large-scale projects, including some of the National Aeronautics and Space Administration (NASA) projects and autonomous agriculture systems.[56][57][58][59] This widespread adoption across varying robotic fields demonstrates the versatility and ability of ROS to handle complex tasks. Its modular design allows projects to start small, utilizing the core functionalities for basic robot control. As projects become complex, additional ROS packages can be seamlessly integrated, enabling functionalities like sensor fusion, robot to robot interaction and human to robot interaction.

This makes ROS a suitable choice for our project, providing tools to simplify the development yet having full control over the project. It also lays a robust foundation for the robot arm, allowing for potential expansion and addition.

### 5.4.3 Choosing hardware for running ROS

TM | SG

The selection of hardware to run the Robot operating system (ROS) required careful consideration. The Raspberry Pi 4 Model B emerged as the optimal choice for several reasons, among them its alignment with our project's portability, documentation, and technical specifications requirement.

The exhibition model's requirement for portability meant we needed a hardware solution that could independently run ROS, eliminating the dependency on a bulky computer or the limitations of simpler micro-controllers like Arduino. With its compact size, this is where the Raspberry Pi 4 Model B presents a significant advantage. Its small footprint ensures that our robot arm remains portable and easy to display at exhibitions and to stakeholders without compromising on computational capabilities.

Performance-wise, the Raspberry Pi 4 Model B delivers the requisite computational power to efficiently run ROS, calculate and manage the arm's movements while processing sensor data. This will ensure that our robot arm operates smoothly.

The Raspberry Pi is well-documented and backed by a large community. This vast reservoir of knowledge and resources will help us develop, enabling us to leverage existing solutions and troubleshooting insights.

Cost-effectiveness is another critical factor in selecting the Raspberry Pi 4 Model B. Given the project's scope - developing a concept and an exhibition model - maintaining budget-friendly while achieving our technical objectives is crucial. The Raspberry Pi offers a balance of affordability and performance, making it a sensible choice for this project. [60][61]

### 5.4.4 Setting up a software development workflow

AL | TM

Establishing the correct workflow among software developers is absolutely essential to ensure the fulfillment of software milestones and adherence to the project timeline. Implementing an efficient and clear workflow increases productivity, facilitates rapid development, and minimizes the risk of losing documentation and technical work by removing potential workflow barriers or uncertainties among members. In other words, the difference between implementing an efficient or less-clear workflow can be hugely decisive for whether the developers are able to succeed with their planned efforts.

Originally, as a part of striving to configure the workflow in accordance with the key points mentioned above, the team's software developers created a GitHub remote repository with two persistent branches: *main* and *development*. The repository served as our ROS workspace and was cloned twice in order for each developer to work independently while utilizing the remote

branches as a form of cloud storage to enable sharing between the two locally cloned repositories. During development the idea was to utilize the local development branches to conduct individual code creation and verification efforts, and push to the remote repository once the code feature is desired by the other, approved by both parties, deemed error-free and maintains at least partial functionality. Authentication and signing of the pushes to the remote repository are done using individual SSH keys, simplifying tracking of the push originators.

However, due to encountering several remote and local merge conflicts as a result of pushing to and pulling from the same remote development branch, we decided to add two additional branches: *devA* and *devT* (one for each developer). These branches now assume the function of conducting individual work, while the already established development branch exclusively serves as a code sharer. The last letter in the branch names also correspond to the first initial of the developer's name, in which the specific branch belong to the respective developer. This approach, combined with conducting the actions listed below when needed, ensures minimal interference between our work-efforts and reduces the likelihood of encountering remote merge conflicts. However, should they arise, they will be addressed and reviewed in collaboration to add an additional layer of safety, ensuring that nothing gets accidentally overwritten.

- Pulling from the remote development branch and merging with the local *devX* branch.
- Merging from the local *devX* to the local development branch and pushing to the remote development branch.

In regards of the *main* branch, it initially needs to be merged locally, when the developer's local development branch, or a specific feature inside, has been fully validated and deemed completely stable, before pushing to the remote *main* branch is considered permissible. Fig. 18 illustrates how different GitHub or workflow operations could look like.

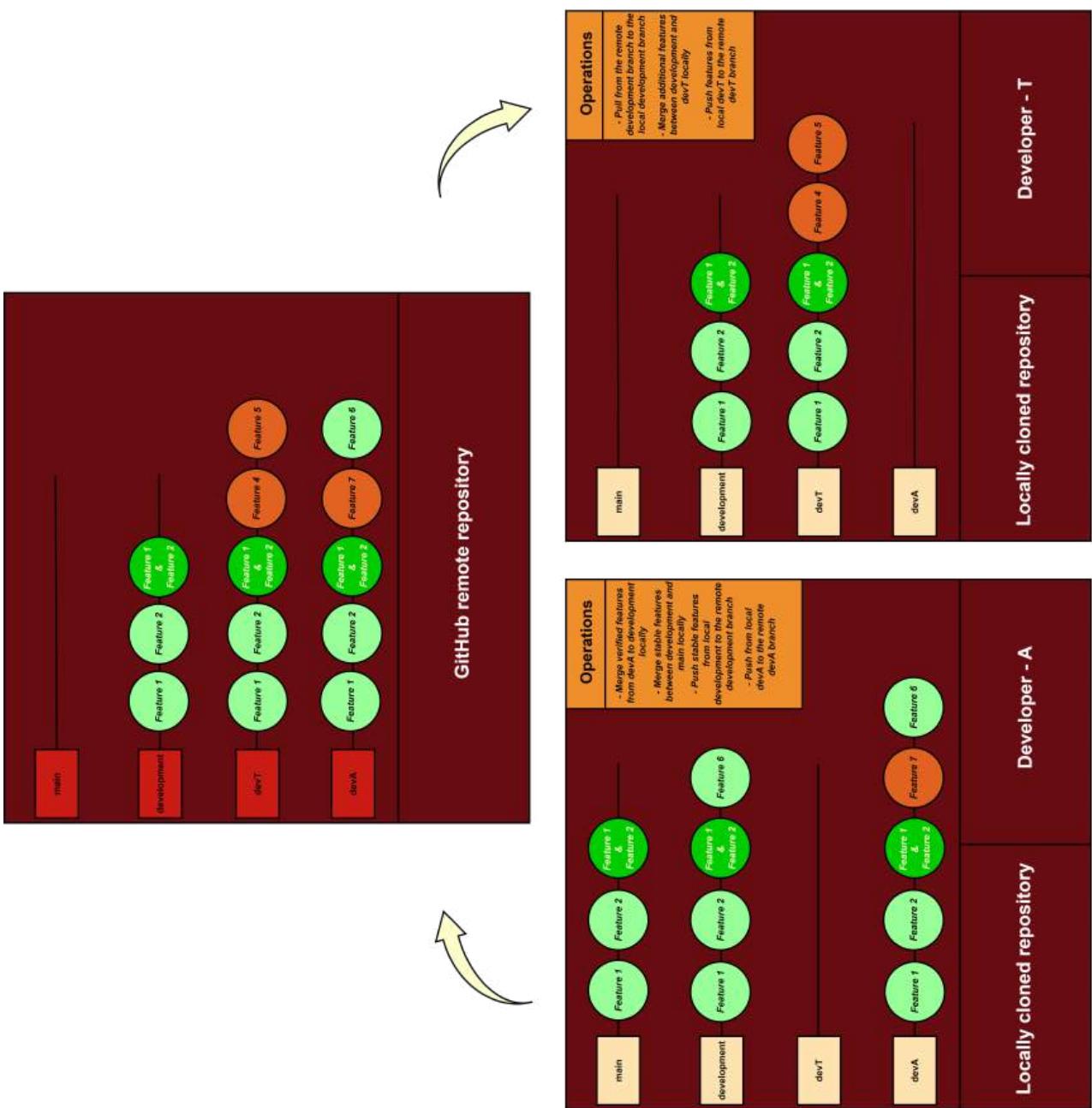


Figure 18: GitHub or workflow operations illustration

While establishing the GitHub remote repository, we also aimed to configure the IDE along with other complementary resources. Initially, we chose to utilize VSC with a VM, which ran Ubuntu and Robot operating system (ROS) and hosted a cloned repository, on each local computer. The reasons for implementing these technologies can be described as follows: Choosing Ubuntu was based on the simple fact that it was the Linux operating system that we were most familiar with, while running a VM provided us with an isolated and controlled environment separate from our local host operating system. VSC also enables easy access to the cloned repository in a tree-like manner and facilitates development in various code languages. The latter was particularly applicable to us because, up to this point, we were unsure whether to develop in either C++ or Python. Additionally, we also created individual accounts for each developer on the Raspberry PI with a cloned repository and the appropriate version of ROS. We have also configured the Raspberry Pi with a user that other operators will utilize when operating the system.

However, after discussing these choices with the developers of the bachelor group: *Kromium*, they provided us with a better idea: Utilize the Remote - SSH extension of VSC, on the local computer, to directly connect to the specific user's workspace on the Pi, using the associated username and the device's IP address. This represents the most efficient way of working, as the need to physically change from a local computer to the Raspberry PI is bypassed. Unfortunately, we encountered an issue with the Eduroam network blocking the SSH connection to the Raspberry Pi, while experiencing multiple unexpected crashes on the VM. As a result, we partially opted for a third alternative workflow while researching solutions to resolve the network issue. The former involved using a WSL on our local computers, with a cloned repository, for development, while utilizing our users on the Raspberry PI for verification. Naturally, this proved significantly less practical and productive than the most optimal alternative. Luckily, we eventually found a solution to overcome the connection blocking and implemented the second alternative: Establishing a private network at our own workspace in *Innovasjonsloftet*. A visualization of the connection can be viewed in Fig. 19. As a final note, Python was also chosen as our primary programming language at this point due to its versatility, simplicity, and ability to offer the same desired functionalities as C++.

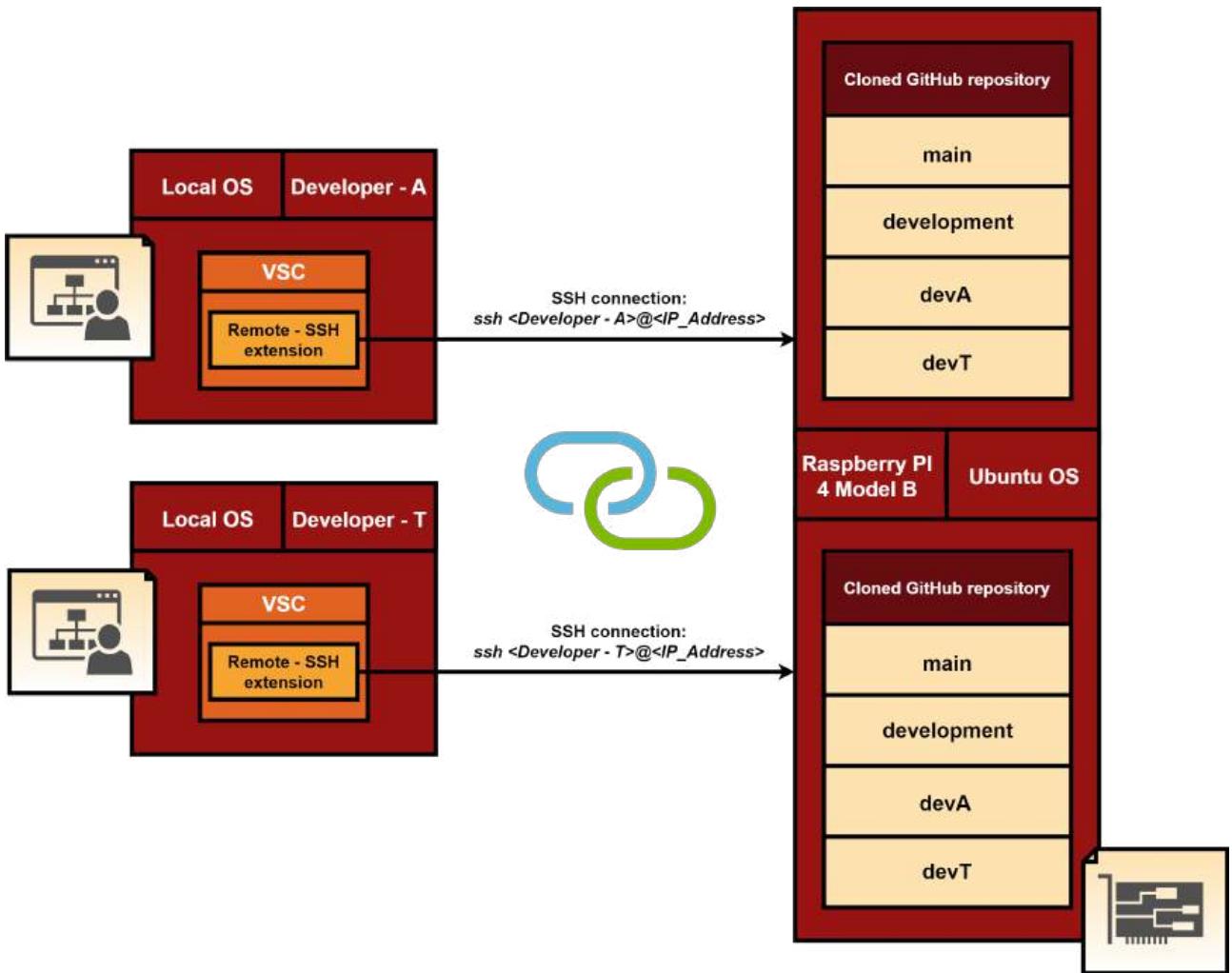


Figure 19: Remote - SSH workflow visualization

#### 5.4.5 The Unified Robot Description Format file

TM | LTR

The Unified Robot Description Format (URDF) serves as the foundational blueprint of a robot arm within the Robot Operating System (ROS). It defines the robot's architecture as a hierarchical, tree-like structure, where each branch embodies a physical segment (link) of the arm with joints connecting these links and allowing for movement. Planning algorithms use the URDF data to calculate safe and efficient trajectories for the arm to reach desired poses, while controllers can translate these trajectories into motor commands for precise movement.

The package rviz, together with a URDF file, allows for simulation and theoretical validation of a robot arm's physical design and control system. This helps identify potential issues before proceeding to physical implementation.

#### 5.4.6 How the URDF file was created

AK | TM

To simplify the servomotor coding, a decision was made to simulate the motor control and position of the effector (tool). For this, there are many existing solutions: One is a conversion from the SolidWorks model file: SLDPRT or SLDASM to a 3D model focused on joint movement and

simulation URDF File. On the website [wikiros](#) [62] there are premade add-ins for SolidWorks. The problem we encountered was that this download was developed for SolidWorks 2021 and the ROS 1 Platform. When searching for the add-in in SW it did not show up in the add-in other add-ins. There is uncertainty on whether this is a download/Software issue or if the 2021 version is not compatible with the SW 23/24 version.

Another solution was to export the Assembly as a STEP file onto a platform called Onshape. A STEP file is a Standard for the Exchange of Product Data also known as ISO 10303. This is a file type commonly used for 3D printing and saving complete CAD bodies. Onshape is a CAD Platform usable in internet-browser. The SW assembly (Junior) is saved in SolidWorks as a STEP file and imported to the Onshape platform. Note that when converting to STEP, only the design and dimensions will be exported, not the mates made in SW. The step files then needed to be mated again on the Onshape. In the browser version, the shortcut for mates (m) you get the mate connector interface. To fully define the model the base is fixed in place. The revolving joints are mated by the revolute mate, defining two circular shapes, this has the same effect as the SW mate Concentric. The non-rotating surface on the surface was connected by fastened mate connectors, which of the two surfaces to connect are selected. [63] [62]

#### 5.4.7 Roadblock during development

**TM | LTR**

The ROS ecosystem comprises of packages that can be used to simplify development as they contain functionality for path planning, simulation and much more. Our plan was to use the URDF file in these libraries to simplify the development, this did not go as expected as the URDF format did not support the physical design of our robot arm. URDF robot arms are built like trees and do not support circular connections. Without a functional URDF file we are unable to use packages: rviz(simulation), moveIt(path planning) and tf2(coordinate manager). We met this roadblock after trying to simulate the robot arm by loading the URDF on a ROS node running rviz.

By discussing the problem, we better understood the situation and an idea of how it could be solved. We concluded that ROS still was the best option, but instead of using preexisting packages, we would make the packages for joystick control, path planning, and mapping ourselves from the ground up. We believe this is doable within the timeframe, as our system is quite simple in terms of the arms movement, and it only operates in two dimensions at the time. This approach would make the project more interesting and academically relevant.

We envisioned a solution where joint angles for both motors would be stored in each point in a coordinate system. The arms' end effector would virtually move through these points, reading the stored joint angles and moving the motors accordingly. Storing joint angles in co-

ordinate points is called mapping. It's crucial that the mapping sequence is done with accuracy and consistency, as it will determine the quality of the movement.

## 5.5 Dynamic payload capacity challenge

AL | SG

During the development process of the mechanical design for the final prototype or the Senior model, our team encountered a critical challenge that significantly halted the design process: The smart servo engines were unable to perform the desired pick and place movement and handle the total weight of the Senior model simultaneously, without entering emergency mode, when additional mass was applied to the Junior model. The issue arose due to applying torque levels beyond the engines' maximum dynamic torque, which ideally should have been identified and addressed earlier. However, due to the initial lack of required data to perform such calculations at the time of purchase, conducting thorough considerations was challenging. One could naturally argue that we should have oversized our engines to account for larger torque levels, but it is easy to be wise in hindsight. To counter this challenge, we rapidly discussed different alternatives. We devised a plan to implement one of the four following approaches: Acquire new engines, downscale the Senior model, keep the present scale of the final prototype, but apply gearing to handle increased torque levels, or combine alternatives two and three.

### 5.5.1 Alternative motor research

SG | AL

Firstly, the LSS-HS1 motor had to be investigated to understand their theoretical limitations clearly. It was discovered that these motors are the high-speed variant of the LSS motors, which consists of three models [52]: A high-speed, high-torque or a standard variant, where the latter combines both torque and speed to a balanced degree. The intended purpose required a high level of torque, and in reconsideration, the high-torque model should have been bought, even though mass reduction remained essential as well. Replacing the high-speed motors with the high-torque edition was the best alternative since no additional modifications were required to implement these. Besides researching other LSS engine alternatives, additional types were also reviewed. These ranged from model-plane servo engines [64] to industrial high torque [65] motors. However, these would increase the amount of work and modifications required in all disciplines, rendering the options unfeasible. Upon reexamination, the motor research at the beginning should have been conducted to a greater degree. In addition, inadequate knowledge about the required motor specifications should have been accounted for by compensating with more powerful engines.

### 5.5.2 Initial dynamic payload tests without gearing

AL | SG

After presenting the approaches to our external supervisor, it became evident that purchasing

new engines represented a last-resort solution and that we should conduct tests to verify the maximum lifting capacity of the smart servos. The initial tests would serve as a solid foundation to determine the correct gearing ratios and assess whether physically realizing the down-scaling, alone or in combination with gears, was necessary.

The initial testing process began by moving one of the engines, connected to the active arm of Junior model, back and forth within a range of degrees from 0 to 50 while applying additional mass at four different arm lengths ranging from 83 to 122 mm from the center of the engine. The purpose of the test was to identify breaking points and clarify the utmost maximum lifting capacity at different arm lengths. The movement pattern was also executed by transmitting positions to the smart servo equivalent to the specific interval of degrees. The trial proceeded as follows, starting from the outermost arm:

- Add additional mass of 50 - 100 grams
- Verify if the motor is able to handle the load at the last arm length
  - If it is able to handle the additional weight go back to step one and continue
  - if it is not able to handle the extra weight move the additional mass one arm length increment towards the center and continue from step two

After concluding this trial, we calculated the engine's maximum holding capacity at the outermost arm length of 220 mm from the center of the left-placed engine. This length represented the utmost distance from the engine to the tool hub during a sideways movement of the robot arm. The calculation, based on a formula for static torque, provided a theoretical lifting capacity estimate of 500 to 600 additional grams (not including the weight of the nuts, bolts, active and passive arms, connections, and parallel arm of the Junior), but further testing was necessary to verify these findings physically. The subsequent test followed the same pattern as before (See App. J and the first five pages), and provided a preliminary physical lifting capacity estimate of 457 additional grams. This value represented the combined sum of the comfortable lifting capacity at arm lengths: 220 mm (101 grams) and 122 mm (356 grams), with 122 mm being the distance from the right-most motor to center of gravity on the toolhub in its out-most lifting position.

Consecutively, it became imperative to solidify the maximum amount of additional mass that could be handled by the motors during a complete pick and place movement at different speeds. The left-most degree indicates the angle associated with the left-most engine and vice versa. See App. J and the table with the title: "LSS-HS1 - dynamic payload capacity test without gearing" for the test results. Below is a list that describes the movement pattern and the corresponding angles of the smart servos:

- Move the robot arm from its bottom position to its default position (null point):

– (90, -90) -> (0, 0)

- Move the robot from its null point to the outermost sideways position:

– (0, 0) -> (63.7, 50.9)

- Move the robot from its outermost sideways position back to its null point:

– (63.7, 50.9) -> (0, 0)

- Move the robot from its null point to its bottom position:

– (0, 0) -> (90, -90)

After concluding these tests and combining the maximum weight of the additional evenly distributed mass with the total weight of the Junior model, in which the movement pattern was completed to a satisfactory degree, we determined a total dynamic payload capacity of 670 grams. See App. J and the table with the title: "Total mass during test"). This capacity was significantly lower compared to what was necessary to handle the present weight of the Senior model, which led the team's mechanical engineers to implement gears and downscale the Senior model.

Furthermore, it is worth mentioning that the presence of axle friction, between the arms of the Junior model and the PLA material, was neglected in every test of this sub-chapter. Regarding the initial test, we were unable to connect the active arm directly to one of the motors and mitigate the effects of axle friction, because the axle-connection would not withstand the tension from the additional load.

### 5.5.3 Downscaling of senior

AK | MO

Concurrently with the gear testing another alternative was explored. We opted for further downscaling with high emphasis on mass reduction. This new model was codenamed Gramps as the grandpa model (Junior-Senior-Gramps). The went with a 1/3 scale model, as this severely reduced the mass, but still had a size suitable for an exhibition model. As a group, the preferred option was to keep the 1/2 scale and hoping for success with gear testing. The downscaling and mass shedding alternative gave good results as shown in the figures below:

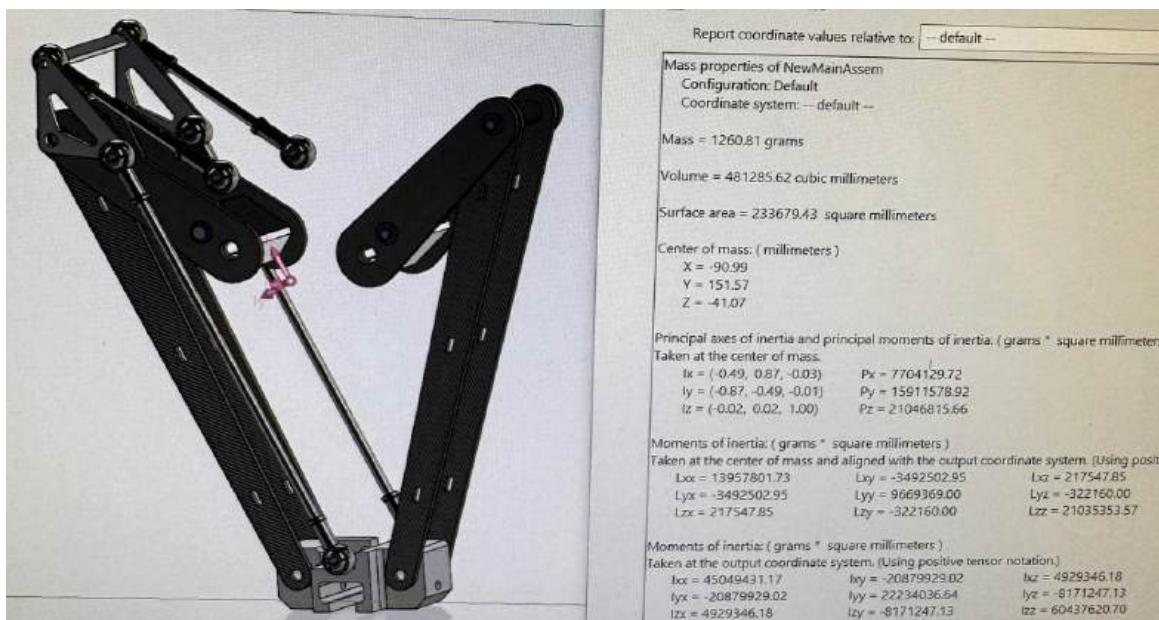


Figure 20: Original mass

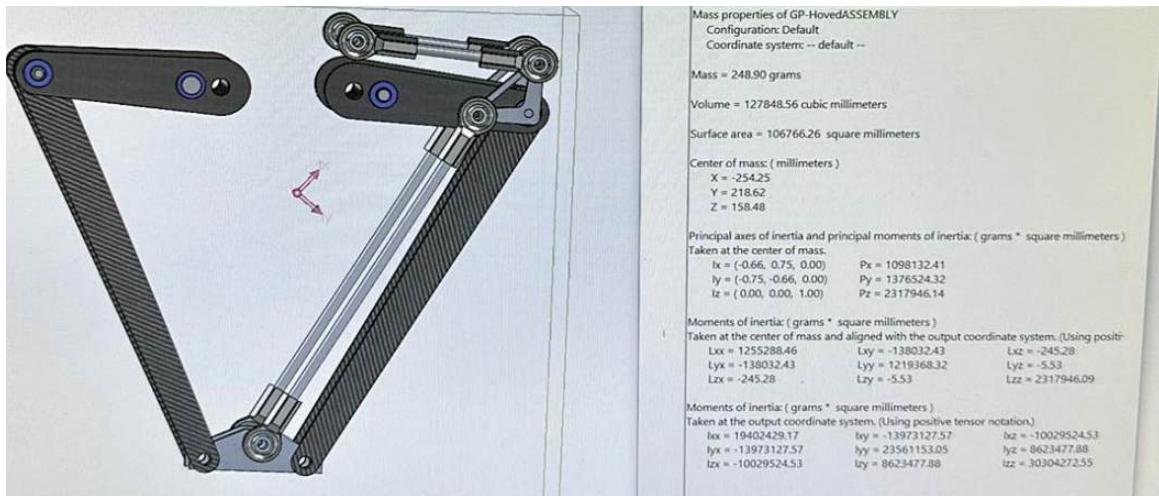


Figure 21: Mass after downscaling and mass reduction

Even with the good results, the Gramps was discarded midway as the testing also proved the gears highly efficient.

#### 5.5.4 Gear implementation

SG | AL

During testing it was made clear that research and experimenting of the smart servo's lifting capacity with gears was necessary. This was a necessity for reaching the A-requirements and showcase the robots functionality while ensuring minimal wear on internal gears inside the motors. The challenge regarding lifting capacity, had to be solved as soon as possible in the most sensible manner.

Information acquired from the static torque formula:

$$\text{Torque} = \text{Force}(N) * \text{Arm}(L) \quad (1)$$

Where:

$$\text{Force} = \text{Mass}(m) * \text{Gravity}(g) \quad (2)$$

Since both the arm and gravitational parameters were constant values in the outer-most static tool-hub position, it was obvious that they could not be changed. The only variable that could be manipulated was the mass. We wanted to increase lifting capacity and from the tests it was found that this had to be increased by a factor of at least two. This was crucial to ensure a safety factor for the lifting capacity and guarantee a full range of motion and operability. Deciding the gear ratio was dependent on the motors maximum allowable torque without entering emergency mode and/or tearing the internal gears of the LSS-HS1 engines. As mentioned above, the total maximum lifting capacity comfortably were at 670 grams and our need at this time before mass reduction was twice of that. To ensure the latter, a gear ratio of 1:2 should be suitable. See App. K and the table with the title: "Gearing parameters (1:2 scale) for an overview of various calculated parameters. Different ratios like 1:2,5 and 1:3 were also investigated but discarded due to limited space on the fastening-hubs mounting surface. Additionally, these ratios would lead to complications for our motors operability, since our test results clearly showed that an reduction in engine speed with an increase in weight resulted in operational failure. The reason being how the LSS-HS1 motors calculated applied torque through the following equation:

$$\text{Torque} = \frac{\text{Power}(Watts)}{\text{RPM} * \frac{2\pi}{60}} \quad (3)$$

Where:

$$\text{Power} = \text{Voltage}(V) * \text{Current}(I) \quad (4)$$

A reduction in RPM with a constant supply of power increases the value for applied torque drastically, which results in abrupt motor failure.

To determine the necessary torque for handling the desired weight of the Senior model and add additional assurance in implementing the correct gear ratios, we initially opted to output the torque directly from the software. Unfortunately, this proved unsuccessful as the LSS-HS1 software library lacked such functionalities. Hence, we turned to the mentioned formula of torque as the way to go, but we were again met with issues that impaired us to practically apply this formula: The RPM varied between zero and a set upper ceiling when utilizing the move functionality of the LSS-HS1 software library. We tried to solve it by utilizing different max RPM values of given ranges and draw connections to the power, current and torque values of the LSS-HS1 performance graph [52]. But the displayed values of the graph did not cohere to the theoretical calculations of output torque, current or power, rendering it impractical for further analysis. Consequently, we also tried to output current values from the software but met the same issue as with the RPM.

The current value displayed 0 milliampere at multiple occasions during operation which could be the cause for the motors to enter emergency mode and shutting down. The most efficient solution to this was to run the engines at maximum speed and accepting the fact that we could not fully figure out the reasons behind why this occurred. It is likely due to how the various internal processes are hard-coded into the smart functions for LSS-HS1 motors. Even though this issue went undisclosed, it was apparent that the output speed by utilizing gearing would be decreased by the same amount of increase in output torque. Thus, the gearing ratio most suitable for our design was 1:2, where the output speed was halved and output torque was doubled.

### **5.5.5 Gears for testing**

**SG | AL**

The gears were modeled utilizing the SW toolbox at a 1:2 ratio, where the smaller gear had 15 teeth and the larger had 30. Since the pitch diameter was (30 and 60) mm, respectively, a 14,5 degree pressure angle on the teeth was applied, causing the gears to rotate properly with the use of SW gear mates. On the basis of the successful results, these gears were made ready for print. [Bib1]

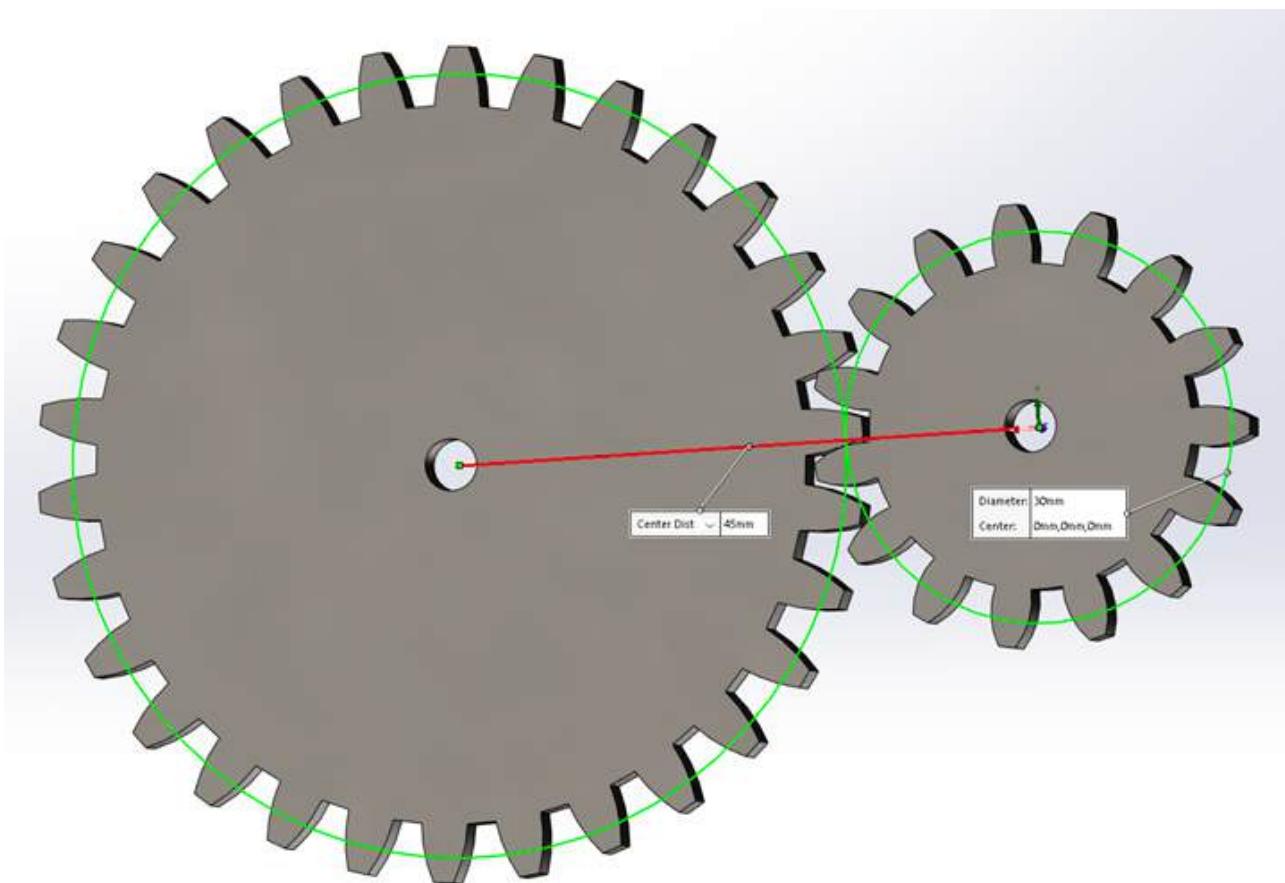


Figure 22: Designed gears

In the figure above, the green circle shows the gears pitch diameter, and the red line represents the center distance between the shafts. The gears shown lack the M2x0,4 mounting holes around the central axis due to SW toolbox complications.

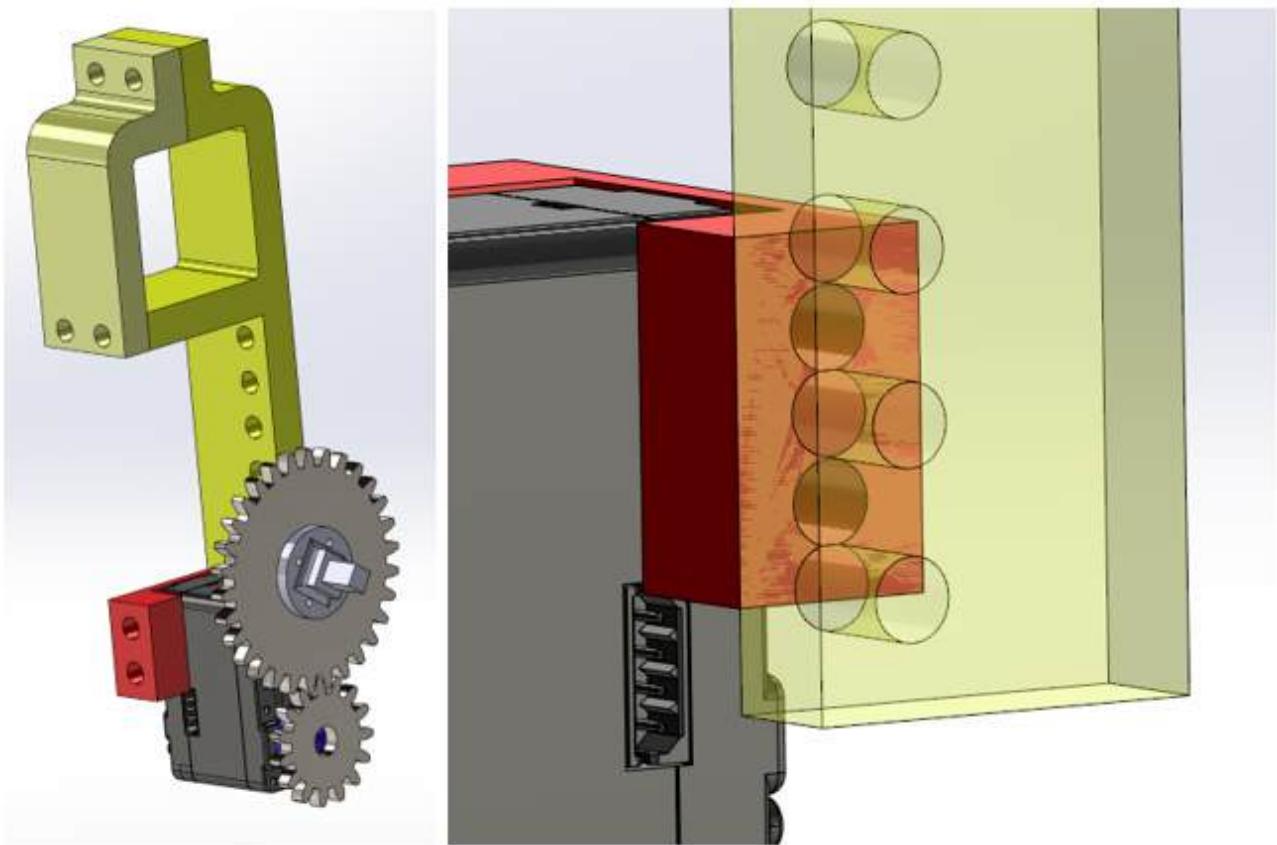


Figure 23: Implemented gears

Shown above is a compilation of the gears implemented to the already existing Junior design. The two gears are mounted directly to the drive wheel of the LSS-HS1 motor. Then fastened to the end tap of existing axle connection with M2 screws. By implementing it to the already existing model, time spent designing was saved, even though new mounting holes for the red motor bracket had to be drilled at USNs workshop. It was easier and faster to create new holes, than trying to figure out the correct gear size that would fit the design. The mounting position was easily discovered due to adequate spacing behind the existing holes. When the gears were mounted on the Junior model, a new issue arose. The model was not perfectly axially aligned, and a quick fix by locking the three different fastening supports for the motor and axle in different heights ensured sufficient contact between the gears to conduct further testing. Additionally, the gears were also printed using PLA, the same material utilized on most of the parts produced to Junior. This material should, in theory, be strong enough and well-suited for further testing.

### 5.5.6 Subsequent dynamic payload testing with gearing

AL | SG

Following the construction and implementation of gears on the Junior model, we continued where we left off with the former test scheme, validating a complete pick and place movement. Additional mass was evenly applied to both passive arms in intervals of 50 to 250 grams,

starting from 0 and ending up with a load of approximately 1400 grams, in which the Junior model was able to complete the desired pick and place functionality. See App. K and the table with the title: "LSS-HS1 - dynamic payload capacity test with gearing" for the test results. We decided to conclude our testing at this point, as the model demonstrated more than the required capacity, and we wanted to avoid the possibility of damaging the engines' hardware and the supporting framework. In total, combining the weight of the Junior model's structural design and the maximum additional mass, the dynamic payload capacity of the smart servos was more than doubled as a result of the gear implementation: The leap from 670 to 1561.08 grams (See App. K and the table with the title: "Total mass during test") granted us with a safe margin to successfully operate the Senior prototype on a 1:2 scale.

# 6 Senior development

## 6.1 Mechanical

### 6.1.1 Gear

SG | MO

#### Introduction:

Designing or choosing the correct gears for the application is important. There are a few different types of gears, like spur, helical, worm and bevel gears. For the intended purpose, it could be used spur or helical gears and spur gears were chosen, avoiding over-engineering and fewer implementation difficulties since time well spent at this process stage is paramount. The main purpose of gears is to transmit motion and power between shafts, either in parallel, non-parallel, intersecting, or non-intersecting. In addition, the ratio between spur gears is determined by the ratio of the number of teeth in the main gear to the number of teeth in the second gear. Gears are used to either decrease or increase the output rotational speed or torque on the output shaft. As an example, a 1:2 ratio implies that there is a set of two gears, where the input gear rotates once and the output gear is half of that. In the example, the input gear has half the amount of teeth as the output gear. This means that the rotational speed for the input gear is twice that of the output gear, and the input torque is half that of the output gear. [Bib1]

#### Learned from testing:

From Junior testing with gears, it was taught that the meshing of gears is critical for maintaining good contact and ensuring the most effective power transmission between the gears. It is obvious that gears in pairs must have the same pitch diameter or module and pressure angle to mesh properly and function as intended. If the gears dont mesh properly, power transmission will be reduced, and this introduces challenges like low tolerance and slack between the gears. Eventually, this leads to gear skipping, where the actively driven gear skips teeth on the connected passive gear. Resulting in premature wear, and for this instance, the robot arms would not move to the given positions in code, either by using the smart function with angels or as a stepper with a predefined coordinate system. This mechanical failure is of the most importance to avoid since it is directly linked with arms control. It could be compensated in code, but this was not a viable option due to the complication of implementing and coding it. Furthermore, the gears should mechanically work as intended, since it is easily avoided when designing them.

#### Design and development:

##### *Deciding gear parameters:*

Final gear parameters for Senior were decided from physical and theoretical results after test-

ing. In addition, gear size was limited by design space. It was sensible to use the same amount of teeth with smaller pitch diameters than the ones applied during testing. Pitch diameter was halved down to 15mm for the small gear and 30 for the larger one. The same ratio of 1:2 and pressure angle of 14,5 degrees was utilized. These parameters resulted in the smallest gears possible with a module of 1. Printing any smaller would generate difficulties during production, but these fit perfectly into the already existing designs made for Senior. [66]

*Breaking toolbox link:*

Before any further gear design, it is essential to break the link between the part and SW toolbox due to design complications when editing and mating them in an assembly. It must be done after gear parameters are finalized. The file was originally a toolbox file but saved as a part file, then the part file property status was changed, removing the toolbox utilities link.

***Design and development:***

During gear design for Senior a few iterations were created due to different mounting options. Only two of them are shown below. Mounting gears correctly is critical for its functionality. They must transfer force without rotating around their own axis on mounted shafts. In this situation, the axle and lead screw. These two shafts had a diameter of 10mm and 8mm, respectively, that presented different obstacles. The mounting option from Junior could not be employed due to different shaft designs, lack of design space, and mounting options for LSS-HS1. It could have used different types of gears, but time was limited, resulting in the same gears all over as the best option. This led to design changes on the axle and made it possible for the use of two ball bearings, one on each axle, removing the gravitational load from the arms on the motor. Furthermore, the rail lead screw could either have teeth machined on the end or a gear mounted. Ending up with the latter option due to the spline machine process. It is costly and time demanding. Since it was decided to print the gears utilizing ABS or PLA, almost any design solution could be produced. [Bib2]

*Design iteration 1:*

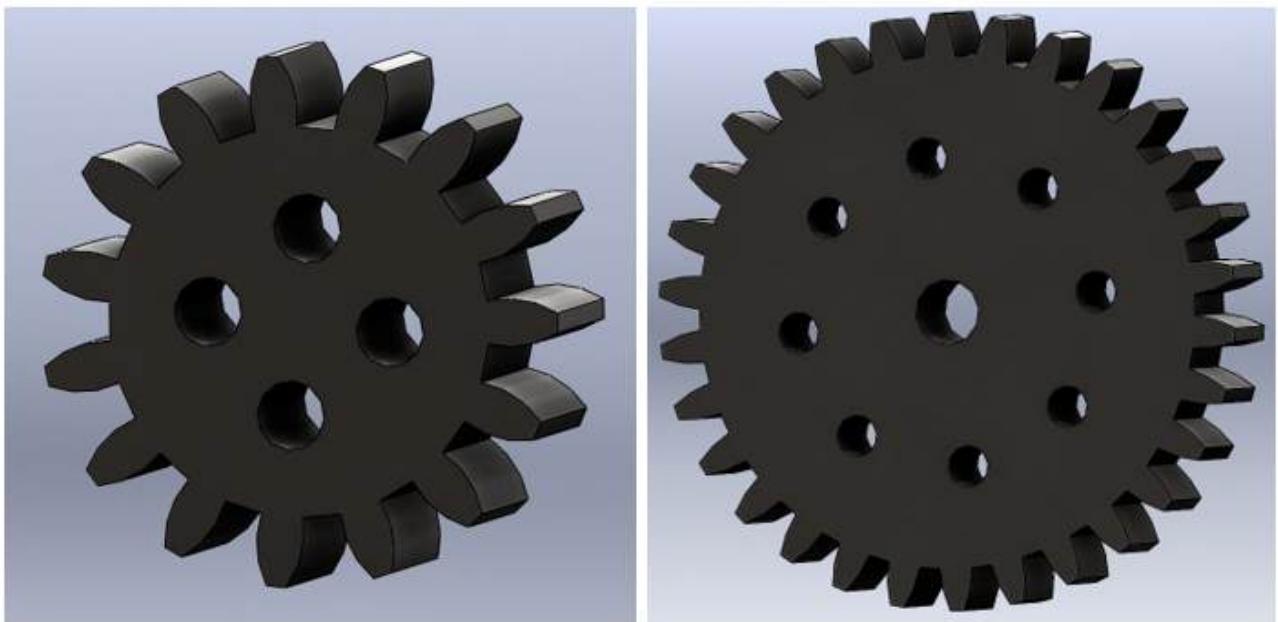


Figure 24: Design iteration 1

Firstly, the design from Junior could not be used even though it would prevent the gears from rotating around its own axis and be locked axially. An option securing the gears axially with a center M4 bolt was investigated. Additionally, the center hole on the gear could be printed with a smaller diameter than the bolt itself. Like a Ø3,5 hole where the bolt would be screwed into the polymer, letting the material friction and threads prevent it from radial rotation. But after a few cycles under load, this would eventually fail.

*Design iteration 2:*

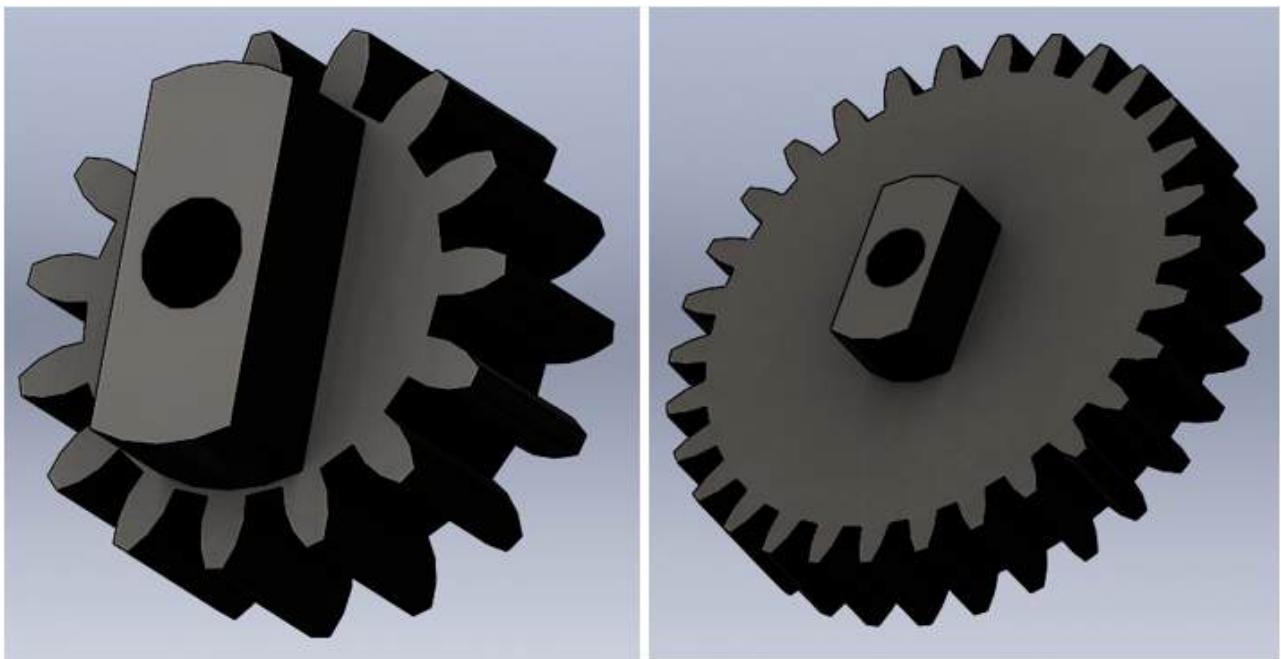


Figure 25: Design iteration 2

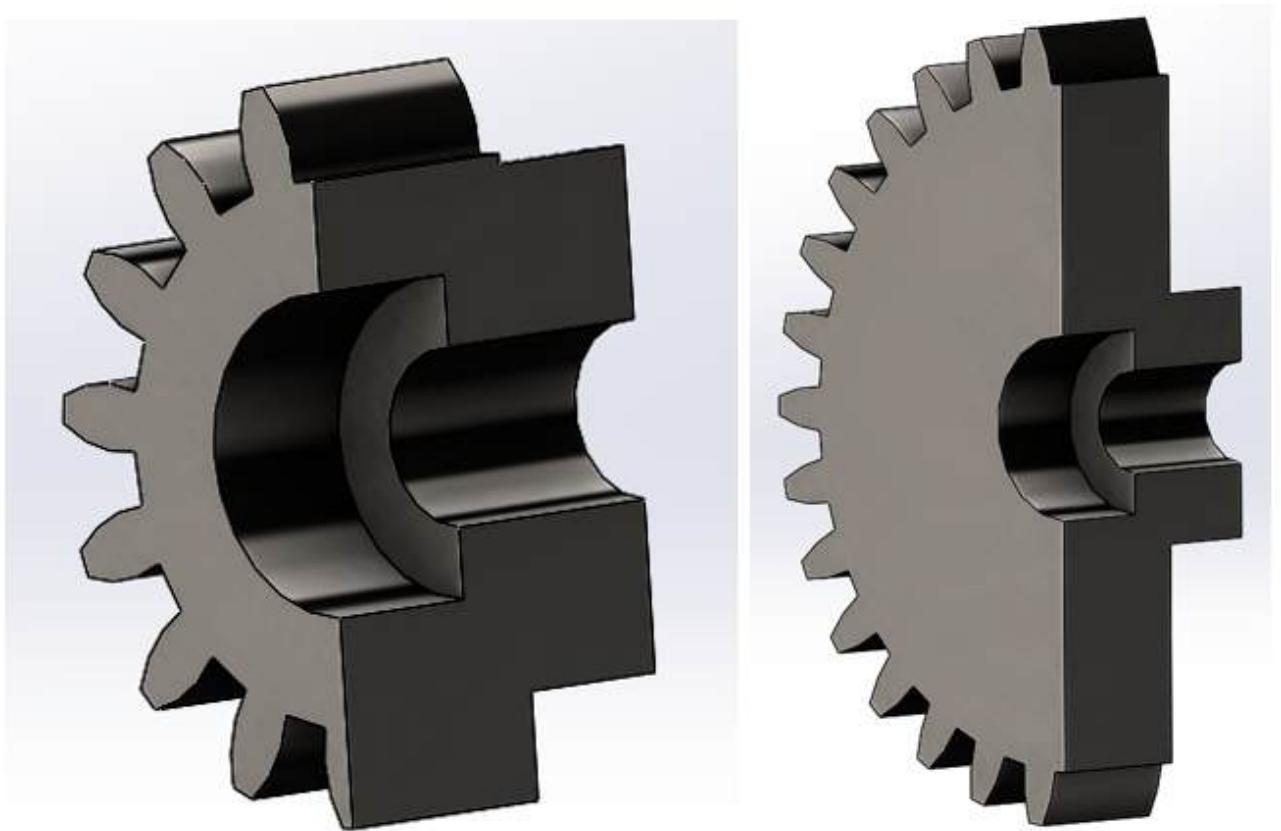


Figure 26: Design iteration 2.1

This led to a new iteration where both axle and lead screw has a milled tap integrated. Gears are printed with the positive tap shape ultimately ensuring a more secure radial lock. Additionally, the center bolt will be fastened with a preload, preventing the gear from moving axially, which also supports radial locking. This strengthening comes solely from the friction between the

bolt head and gear interface. Furthermore, the small gears had to have an indent for the bolt head due to limited space between the gear and the fastening-hub back plate. It could have been designed without it, but that would mean that the axle had to be longer. In addition, the frame-back plate had to be moved further back. Resulting in decreased space for electrical components. In other words, it would have been an unwise decision. In hindsight, a machine key should have been used instead, like the one on the axle mentioned in assemblies bellow. This would slightly change gear design, but the use of a machine key to transmit rotational forces is a more common practice and production-friendly. Additionally, it would spare the axle of an extra mounting position in the CNC machine. Because it already utilizes a machine key, this would reduce production costs further. Moreover, a nominal shaft diameter of 4,1mm was used on the gears center holes due to the printing process. It is common to add 0,2mm to holes, since the polymer will be compressed a tiny amount during the printing process. This assures a snug fit for an M4 bolt with outer thread diameter of 4.0mm.

### 6.1.2 Gear production

MO | SG

Needing three sets of gears for each motor in our design, one set was initially test printed. A test print was ordered from USN to validate the design and to test for shrinkage. The test was printed in the filament ABS by a FDM 3D printer, type Bambu Lab X1 Carbon. It was printed with standard parameter settings with a layer height of 0.2mm and reduced infill.



Figure 27: To the left validating production and design by measuring dimensions using a caliper gauge. To the right a fitting test with an M4 bolt.

In the figure above, you can see the gears being measured to Ø3,9 and Ø7,2. To compensate for shrinkage when 3D printing, 0.2 mm was added to the hole dimensions, and the produced parts had little to zero deviation from the desired hole dimensions. To make sure the required M4 bolt would fit, a fitting test was executed. As seen in the figure above, the fitting test was successful. Having success on the first try with the test gears, there was printed four sets of gears, having one spare set. The only parameter setting changed from the test gears were that we increased the infill, so the parts would be solid and have higher strength.

### 6.1.3 Galvanic voltage series

SG | MO

**Introduction:**

Corrosion is a very important factor to keep in mind when using metals, especially galvanic corrosion that can occur between different metals in contact. The corrosion is an electrochemical reaction that occurs when metals in contact are in the presence of an electrolyte, such as air humidity, water, salt water or other electrically conductive fluids. The electrochemical process occurs as a result from the flow of small electrical currents between the natural difference in electrically charged metals.

**Metal reactivity:**

Each metal has its own voltage potential which comes from their atomic bonds and crystal structure. This voltage potential is from experiments that are based on the metals relative electrochemical activity in flowing sea water. A typical rule of thumb is that a voltage difference of 0,2 volts or more generally result in a high risk for galvanic corrosion. In addition, the less noble a metals is, the higher risk for electron activity between materials, thus more exposed to galvanic corrosion. [67]

Noble means the metals reluctance to undergo chemical reactions, such as oxidation with air that occurs rapidly on polished or machined aluminum surfaces. Hence, pure aluminum is very reactive. There are different alloy compositions like Stainless Steel, which is very resistant to corrosion and rust, where the Steel is a composition of Iron and 0,2-2,11% Carbon, and making it Stainless by adding 10-30% of Chromium and sometimes other metals like Nickel or Molybdenum which makes the alloy more noble. Hence, stainless steel is much less reactive than pure Iron (Cast Iron). [68]

On the atomic level this means filling the outer most electron layer with different bonds to other atoms such as, covalent, ion, hydrogen or van der Waals bonds. [69] This ensures that the element won't easily react to other elements. As an example, air has about 78% Nitrogen and 21% Oxygen. [70] These elements have five and six electrons in the outer most electron shell respectively and will easily bond to elements lacking three or two electrons, to fill their outer shell and become noble. [71] From chemistry, all elements tend to strive filling their outer electron shell known as the valence shell with eight electrons, this is commonly known as the octet rule and topping of this shell makes the atom relatively stable from reacting to other elements. [72] Hence, stainless steel is more stable and resistant to corrosion than pure iron, much like the noble metals such as gold, silver, and platinum. [73]

**Galvanic corrosion process:**

In the galvanic corrosion process the most noble material ends up acting like a cathode and

undergoes a reduction and the less noble is acting like an anode which undergoes an oxidation. The cathode can be seen as the positive side of a battery cell and the anode as the negative side, where the electrolyte is conductor in which the current is carried as shown in the figure bellow. [1]

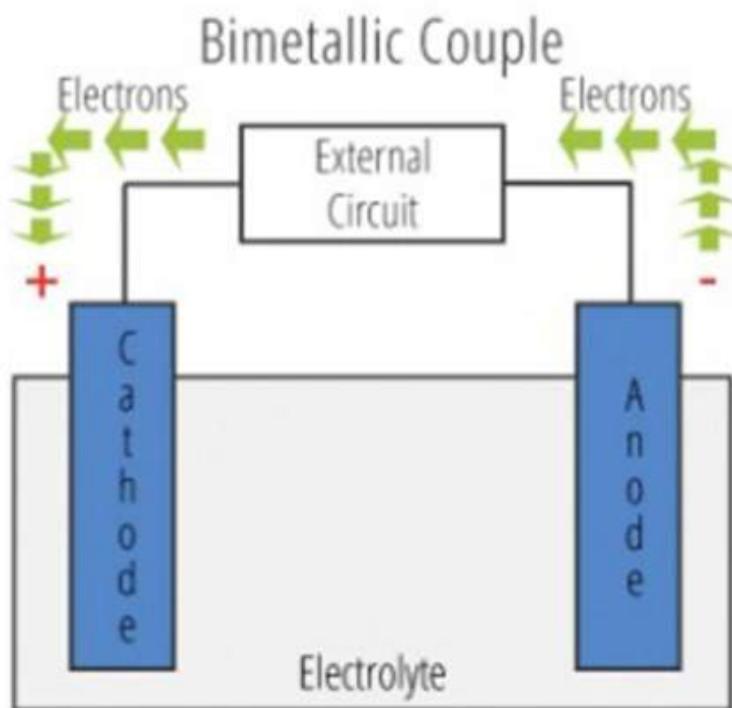


Figure 28: Galvanic cell [1]

Mitigating material deterioration:

Table 9: Galvanic voltage table [11]

No.	MATERIAL	VOLTAGE RANGE	RELATIVE POSITION
↑ LEAST NOBAL (ANODIC)	1 Magnesium	-1.60 to -1.67	
	2 Zinc	-1.00 to -1.07	
	3 Beryllium	-0.93 to -0.98	
4	Aluminum Alloys	-0.76 to -0.99	
5	Cadmium	-0.66 to -0.71	
6	Mild Steel	-0.58 to -0.71	
7	Cast Iron	-0.58 to -0.71	
8	Low Alloy Steel	-0.56 to -0.64	
9	Austenitic Cast Iron	-0.41 to -0.54	
10	Aluminum Bronze	-0.31 to -0.42	
11	Brass (Naval, Yellow, Red)	-0.31 to -0.40	
12	Tin	-0.31 to -0.34	
13	Copper	-0.31 to -0.40	
14	50/50 Lead/Tin Solder	-0.29 to -0.37	
15	Admiralty Brass	-0.24 to -0.37	
16	Aluminum Brass	-0.24 to -0.37	
17	Manganese Bronze	-0.24 to -0.34	
18	Silicon Bronze	-0.24 to -0.30	
19	Stainless Steel (410, 416)	-0.24 to -0.37 (-0.45 to -0.57)	
20	Nickel Silver	-0.24 to -0.30	
21	90/10 Copper/Nickel	-0.19 to -0.27	
22	80/20 Copper/Nickel	-0.19 to -0.24	
23	Stainless Steel (430)	-0.20 to -0.30 (-0.45 to -0.57)	
24	Lead	-0.17 to -0.27	
25	70/30 Copper Nickel	-0.14 to -0.25	
26	Nickel Aluminum Bronze	-0.12 to -0.25	
27	Nickel Chromium Alloy 600	-0.09 to -0.15 (-0.35 to -0.48)	
28	Nickel 200	-0.09 to -0.20	
29	Silver	-0.09 to -0.15	
30	Stainless Steel (302, 304, 321, 347)	-0.05 to -0.13 (-0.45 to -0.57)	
31	Nickel Copper Alloys (400, K500)	-0.02 to -0.13	
32	Stainless Steel (316, 317)	0.00 to -0.10 (-0.35 to -0.45)	
33	Alloy 20 Stainless Steel	0.04 to -0.12	
34	Nickel Iron Chromium Alloy 825	0.02 to -0.10	
35	Titanium	0.04 to -0.12	
36	Gold	0.20 to 0.07	
37	Platinum	0.20 to 0.07	
38	Graphite	0.36 to 0.19	

As mentioned earlier, the voltage difference of metals in direct contact should not exceed 0,2 volts as a rule of thumb. Greater difference in voltage potential results in faster deterioration rate of materials used. As seen in the table above, the voltage difference between wear components of either copper or brass, and aluminum or graphite is a lot greater than 0,2 which will result in excessive wear on components. This creates a galvanic cell where the humid air acts as an electrolyte. [74]

Table 10: Material combinations [11]

Anodic (Corrodes)	Cathodic	Magnesium & Alloys	Zinc & Alloys	Aluminum & Alloys	Cadmium	Steel (Carbon)	Cast Iron	Stainless Steels	Lead, Tin & Alloys	Nickel	Brasses, Nickel-Silvers	Copper	Bronzes, Cupro-Nickels	Nickel Copper Alloys	Nickel-Chrome Alloys	Titanium	Silver	Graphite	Gold	Platinum
Magnesium & Alloys																				
Zinc & Alloys																				
Aluminum & Alloys																				
Cadmium																				
Steel (Carbon)																				
Cast Iron																				
Stainless Steels																				
Lead, Tin & Alloys																				
Nickel																				
Brasses, Nickel-Silvers																				
Copper																				
Bronzes, Cupro-Nickels																				
Nickel Copper Alloys																				
Nickel-Chrome Alloys																				
Titanium																				
Silver																				
Graphite																				
Gold																				
Platinum																				

The table above shows which materials that can be combined in direct contact to ensure a low deterioration rate. It implies that it cannot be used Graphite in combination with copper, brass, or bronze, as well as aluminum alloys in combination with steel or cast-iron. Even though, a car rim is often made from aluminum, and the bolts from stainless steel. Also commonly used elsewhere and the deterioration is often calculated and considered on a regular basis. Additionally, it is often used a copper alloy on either tube or spray can, that acts as a protective layer on the bolts. This layer works as a mitigation measure also called cathodic protection. The sacrificial layer will protect the primary metal from corrosion, even though it is eventually going to happen, but at a much lower rate. This must be considered when maintain a cost-effective product and minimizing expenses by extending lifespan of components.

### Challenge:

Furthermore, in the robot there is direct contact between more elements that creates a corrosion challenge. Where one material CFRP, which is a composition of polymer and graphite. It is the most noble of all the materials used in the system. The less noble materials include aluminum

and cast-iron or steel. As mentioned, it is intended to use softer materials like copper as wearing parts in the interfaces where rotational movement is necessary. Instead of steel, stainless steel could be utilized but these materials are still more noble than aluminum and cast-iron and less than graphite. Our challenge is to allow the use of different materials in direct contact, while utilizing a softer material at interfaces requiring rotational flexibility. These components will be replaced during scheduled maintenance and the creation of a galvanic cell should be avoided, extending lifespan of machined parts and composites. Additionally improving long term cost-efficiency. The challenge must be considered appropriately, since the packaging machine is to be shipped worldwide where a variation in air humidity is guaranteed.

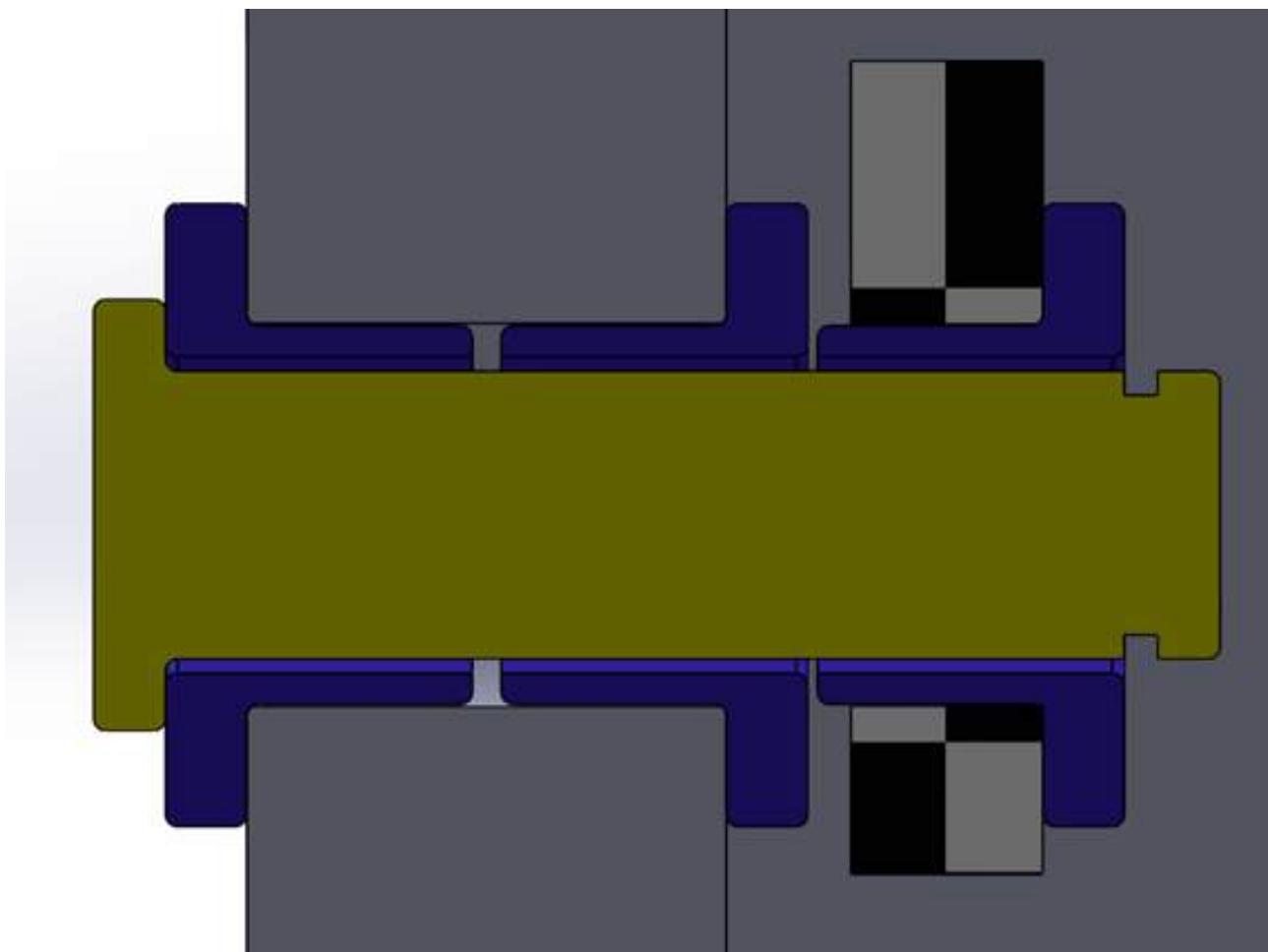


Figure 29: Galvanic cell

As shown in the figure above, the patterned part is passive arm made from CFRP, blue parts are washer with a collar thought to be ordered in copper or brass, and the yellow pin can be ordered in steel or cast-iron. This combination results in a galvanic reaction where materials deteriorate faster than at a normal rate alone.

**Solution:**

To prevent or mitigate galvanic corrosion it can be employed different strategies, such as introducing galvanic isolation, or selecting different material to disrupt the electron flow. By utilizing the polymer nylon as the blue components seen in the figure above, it ensures an insulating layer between the materials, preventing direct contact. This could also be another polymer-based material, but nylon is a good choice for this application due to its material properties. It is highly resistant to wear and chemical reactions, whilst having a low friction coefficient with the correct composition. Additionally, it is important to assure tight fit of wear parts, preventing slack and vibration while not disrupting radial rotational. [?]

#### 6.1.4 Senior modeling

AK | SG

For the senior modeling, individual concept parts were already in development, but needed alterations in regards to materials and design modifications caused by updated manufacturing processes. One of the most important design factors was the height capacity at 210mm. To prevent the arms from over rotating and crashing after stretching downwards. There was experimented with double stabilizers connected at the upmost triangle connection see Fig: 20. This proved to be too restricting as the height capacity was reduced to 197mm. In regards to the final Z-movement and effector angle offset, these are within desired the values. As seen in figures below, the final height capacity is ( $248.1 - 31.78 = 216.22$ ) and the effector is approximately parallel at the bottom (0.01 degrees), and in the conveyor position (0 degrees)

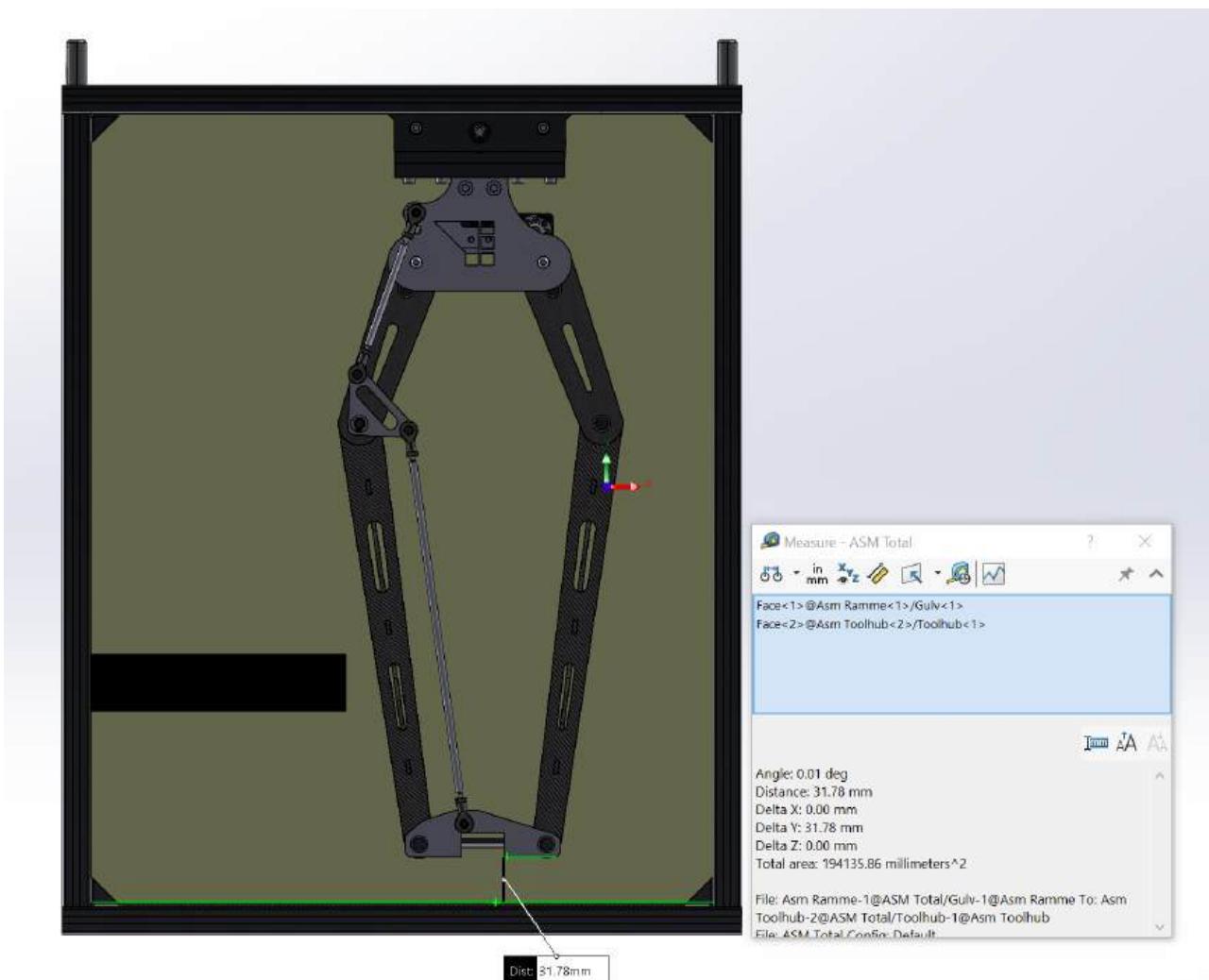


Figure 30: Minimum height and effector offset

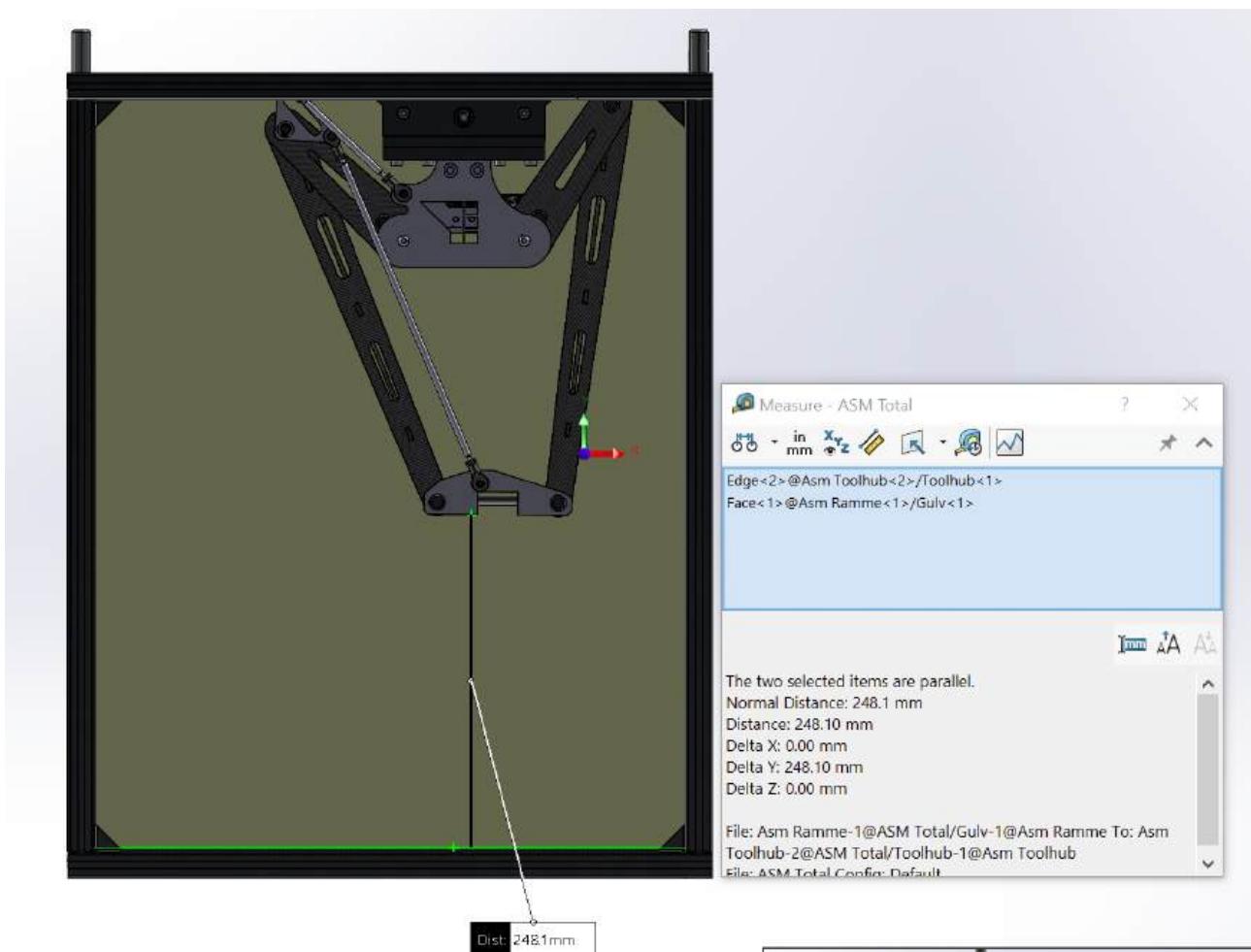


Figure 31: Maximum height

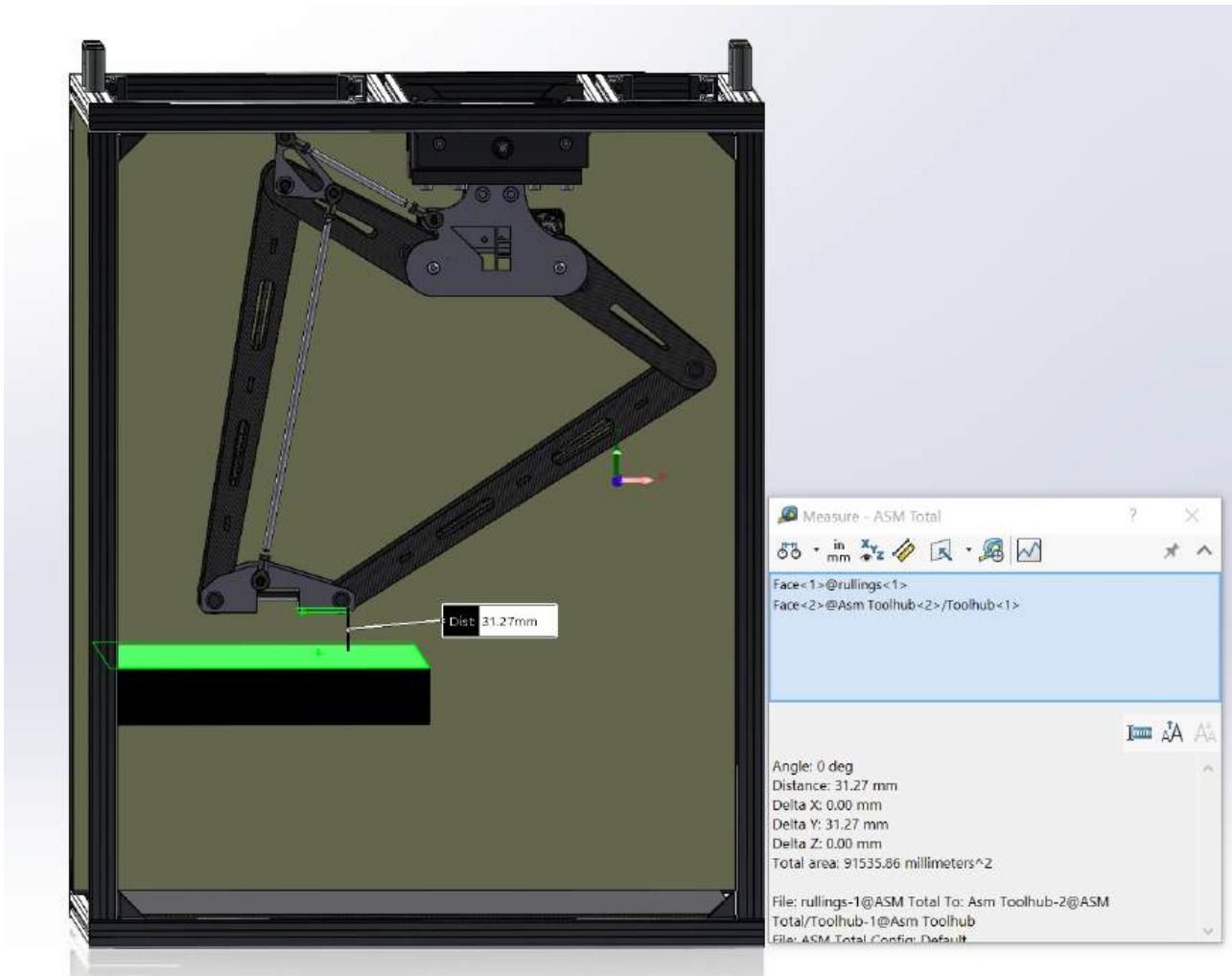


Figure 32: Effector offset at conveyor position

### 6.1.5 Frame

**AK | SG**

#### Introduction:

The frame is composed of seven unique parts, the item-profiles for structural integrity, floor panel made from CFRP. In addition, tinted backwall for mounting electronical components made form plexiglass which also encloses the system. Furthermore, applicable hinges, corner fasteners, angle bracket and handles for transportation. The frames main objectives are providing a stable base for the robot within an enclosed environment.

#### Interfaces:

##### *Rail assembly:*

Assembly rails features its own two 440mm Item profiles, positioned perpendicular to front view. These are fastened by four corner brackets in total, two at the front and two in rear.

### **Design and development:**

The Senior frame development resembles the Junior model, with some minor changes. Conservation of mass being the primary reason for the design change. Shifting from 40x40mm profiles to 20x20mm downscals from 10.6kg to 3.3 kg, as well as the fasteners all decreasing in size. Corner cubes was unavailable for the 20x20mm Item profiles. This implied either a design that could be produced at TE or create a frame without utilizing these. In which the latter alternative was chosen due to cost-efficiency and development time. The back features a plexiglass door with IGUS hinges and magnetic lock. From this door, the white tinted backwall can be accessed. Electronics and microcontrollers are fixed to the wall using superglued Velcro, while the heavier components are supported by the floor. Carbon fiber flooring is made at USN Kongsberg Composite lab. The carbon fiber and epoxy is mostly a design factor and is made with a plywood core for structural integrity. For safety reasons during EXPO, the frame is surrounded by clear plexiglass at all sides exempt bottom.

### **Connections:**

The frame is constructed by utilizing a longer beam in front and rear at 480mm and for the sides a 440mm profile. In addition, four vertically placed at 560mm separating the top and bottom. The profiles are connected at all corners using corner brackets. The floor and backwall are mounted by screwing the 90-degree angle brackets into premade laser cut holes. Front and side panels made from plexiglass are secured by eight t-slot nuts, three at vertical profile and two at horizontal center. All fasteners utilize M5x07 coarse threaded machine bolt. Lastly, two handles are mounted at the top of the 440 profiles. All fasteners utilize M5x07 coarse threaded machine bolt. Lastly, two handles are mounted at the top of the 440 profiles.

### **Exploded view and Bill of materials:**

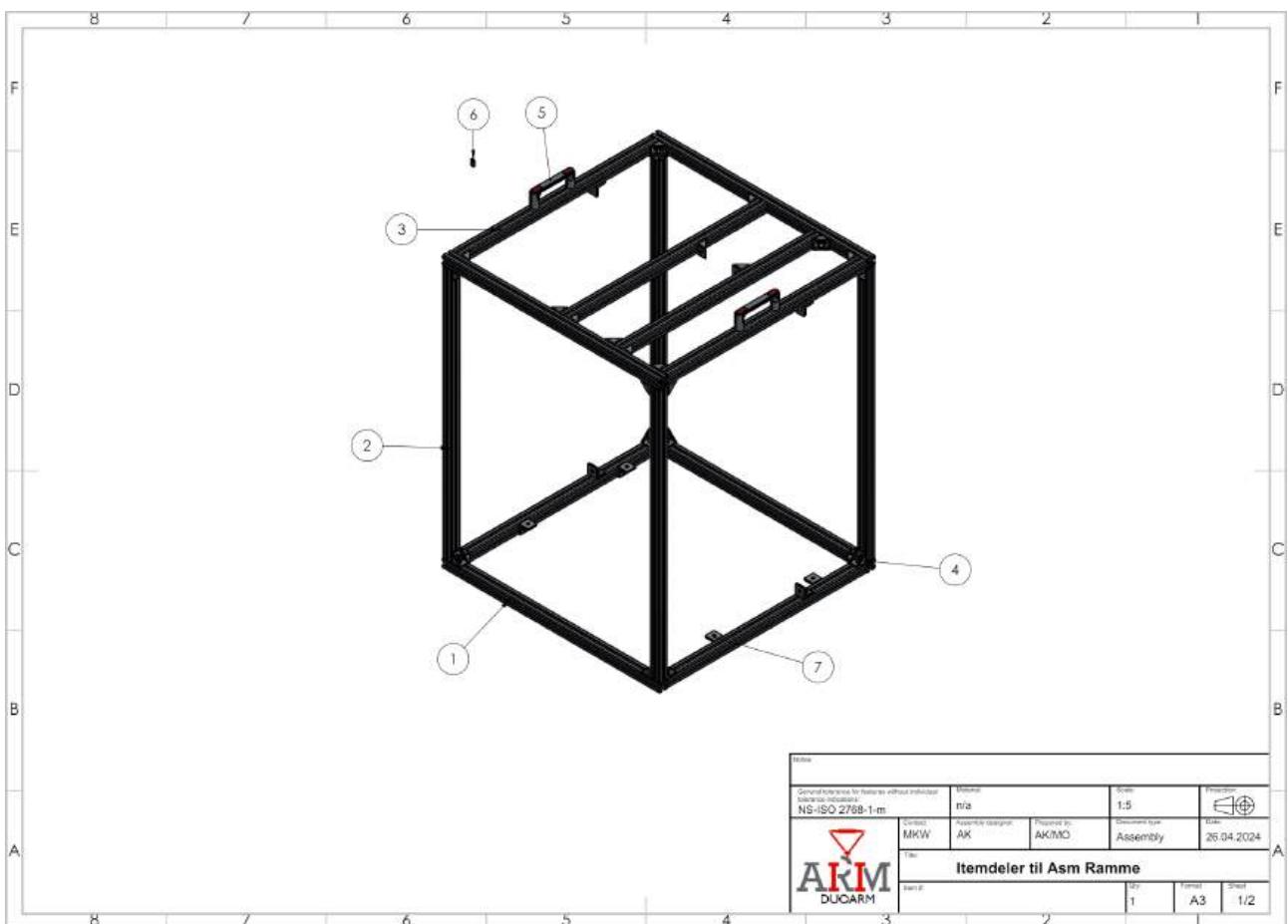


Figure 33: Frame

Shown above is an exploded view of the assembly with all its components. These are mated together as it is intended to be assembled. It has six unique parts for production at TE and five unique parts being ordered from either MiSUMi or IGUS.

## 6. SENIOR DEVELOPMENT

F	Pos.	Title	Item no.	Material	Manufactured	Supplier	Article name	Article no.	Size	QTY.			Link website
	1	Item Profile 20x20x480	310001	6061-T6 (SS)	No	Item 24	Profile 5 20x20	0.0.448.04	20x20	4			<a href="https://www.item24.com/en-it/profile-5-20x20-natural-44804?supplyUnit=STUECK&amp;nominalLength=3000&amp;nuten=-1&amp;category=profile-technology">https://www.item24.com/en-it/profile-5-20x20-natural-44804?supplyUnit=STUECK&amp;nominalLength=3000&amp;nuten=-1&amp;category=profile-technology</a>
E	2	Item Profile 20x20x560	310002	6061-T6 (SS)	No	Item 24	Profile 5 20x20	0.0.448.04	20x02	4			<a href="https://www.item24.com/en-it/profile-5-20x20-natural-44804?supplyUnit=STUECK&amp;nominalLength=3000&amp;nuten=-1&amp;category=profile-technology">https://www.item24.com/en-it/profile-5-20x20-natural-44804?supplyUnit=STUECK&amp;nominalLength=3000&amp;nuten=-1&amp;category=profile-technology</a>
D	3	Item Profile 20x20x440	310003	6061-T6 (SS)	No	Item 24	Profile 5 20x20	0.0.448.04	20x20	6			<a href="https://www.item24.com/en-de/profile-5-20x20-natural-44804?supplyUnit=STUECK&amp;category=profile-technology%2Fconstruction-profiles">https://www.item24.com/en-de/profile-5-20x20-natural-44804?supplyUnit=STUECK&amp;category=profile-technology%2Fconstruction-profiles</a>
C	4	CornerBracket	310004	6061-T6 (SS)	No	Item 24	Angle Bracket Zn	0.0.425.03	20x20	32			<a href="https://www.item24.com/en-de/angle-bracket-5-20x20-zn-white-aluminium-similar-to-ral-9006-42503?type=BR5&amp;outerDimensions=20x20&amp;category=profile-technology%2Ffastening-technology#technical-data">https://www.item24.com/en-de/angle-bracket-5-20x20-zn-white-aluminium-similar-to-ral-9006-42503?type=BR5&amp;outerDimensions=20x20&amp;category=profile-technology%2Ffastening-technology#technical-data</a>
B	5	HandlePi	310005	PA Type 6	No	Item 24	Handle Pi	0.0.679.07		2			<a href="https://www.item24.com/en-de/handle-pi-80-m5-pa-grey-67907?thread=5&amp;color=7&amp;category=profile-technology%2Fgrips-locks-and-catches">https://www.item24.com/en-de/handle-pi-80-m5-pa-grey-67907?thread=5&amp;color=7&amp;category=profile-technology%2Fgrips-locks-and-catches</a>
A	6	T Slot nut	310006	6061-T6 (SS)	No	Item 24	T-Slot Nut St	0.0.370.01	5mm	150			<a href="https://www.item24.com/en-de/t-slot-nut-5-st-m5-bright-zinc-plated-37001?type=BR5&amp;material=verz&amp;thread=5&amp;mountingForm=single&amp;model=3&amp;antifission=true&amp;category=profile-technology%2Fslot-nuts#technical-data">https://www.item24.com/en-de/t-slot-nut-5-st-m5-bright-zinc-plated-37001?type=BR5&amp;material=verz&amp;thread=5&amp;mountingForm=single&amp;model=3&amp;antifission=true&amp;category=profile-technology%2Fslot-nuts#technical-data</a>
	7	Anglebracket-2	310007	6061-T6 (SS)	No	Item 24	Angle Bracket	0.0.677.77	20x20	8			<a href="https://www.item24.com/en-de/angle-bracket-5-20-right-angled-white-aluminium-similar-to-ral-9006-67777?type=BR5&amp;material=5int&amp;category=profile-technology%2Ffastening-technology">https://www.item24.com/en-de/angle-bracket-5-20-right-angled-white-aluminium-similar-to-ral-9006-67777?type=BR5&amp;material=5int&amp;category=profile-technology%2Ffastening-technology</a>

Figure 34: Frame bill of materials

The bill of materials is showcasing each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

Parts ordered:

Bolts: These are ordered from MiSUMi. Further details are found in Bill of Materials. [50]

Item 20x20x6000: [75]

Item 20x20x3000: [76]

Corner bracket: [77]

Angle bracket: [78] Hinge: [79]

Handle: [80]

### 6.1.6 Rails

MO | SG

#### Introduction:

The purpose of the rails is to make it able for the robot to operate along the y-axis in a linear motion. This operation is needed when there is a change in the size of chip packages, and the estimated need for this change is a couple of times during a day. The rail mechanism have to be visible from above and fixed to the frame, it is required to move the robot a minimum of 100mm, connect the robot, and hold its weight.

Initially in the process when we decided to develop the concept Duopod on rails, we were not sure how we wanted to distribute our time. If we would develop only the Duopod and purchase a fitting rail system, or if we had the time and effort to develop both. This was reviewed with TE, and we were encouraged to develop the rail system as well, as this would be an extra challenge for us. Having three group members with mechanical engineering as their discipline, we considered that we had enough work capacity, and one person was assigned to develop the railing system.

#### Interface:

This subassembly has a direct interface to the subassembly frame and the assembly fastening hub. The rails will also have an interface to the rest of the subassemblies that makes up the robot in regards to allowable work and design space, and weight capacity of the rail assembly.

#### Research and design:

There are a lot of existing linear motions systems, and the development of our system started with doing research of the mechanisms and components applied in some of these.

To create movement in a direction, a linear motion system have a drive mechanism which generates the motion, and guides that primarily carries the load of the object, ensuring a straight path. The figure below shows an example of a linear motion system and components needed to carry a load and create linear motion from a rotating motor. When the motor is rotating and translates rotation to the lead screw, the connected ball screw (which is a threaded nut) will move along the screw. This results in rotating motion being converted into linear motion, and the linear distance traveled each screw rotation is determined by the lead of the screw. The lead of a screw has correlation with the screws pitch, which is the distance between the screw threads. The relation between the threaded screws lead and pitch is determined by the number of starts in the screw. To calculate the lead, the number of starts is multiplied with the pitch, which means that for a single start threaded screw the lead equals the pitch.[81]

It is desirable for the lead screw to rotate freely, ensuring a smooth motion and reducing the rotational power input required. To allow for the lead screw to rotate freely, it is connected to radial ball bearings and the external loads are distributed on linear guides.

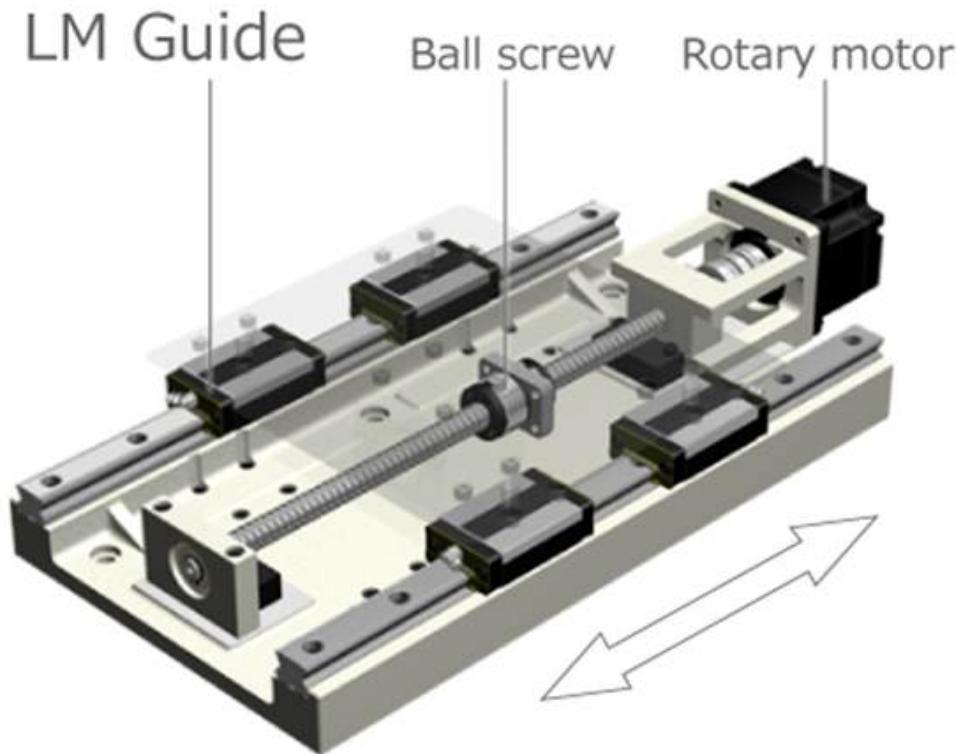


Figure 35: Linear motion system using linear motion guide units with a lead screw and ball screw, driven by a rotary motor.

[82]

There are different options for drive mechanism in a linear motions systems, and two of these are lead screw driven as shown in the figure above, and belt driven mechanism shown in the figure below. The belt drive converts rotating motion to linear motion using belts with teeth that are connected between two pulleys in each end, this resulting in the belt transferring motion to the load.[83]

There are different pros and cons when it comes to these two drive mechanisms, and different parameters were taken into account. The lead screw driven mechanism were considered to be the most suitable for our needs and requirements, because it has lower cost, requires less input torque, and have a higher accuracy and positional repeatability compared to the belt driven mechanism. Belt driven mechanism is favoured when there is a need for high travel speed, efficiency, and a need for longer travel distances, which are not of importance in our case.[84]



Figure 36: Example from Igus of a belt driven linear motion system.  
[85]

The decision of what type of guides were most suitable for our purpose, the requirement of having the mechanism visible from above and that the Rails assembly needed to be fixed to the Frame assembly were leading points to be considered. The two examples seen in Fig. ?? and Fig. 36, would fit our purpose if it was turned upside down, as we need the load hanging below the linear motion system. This would on the other hand result in hiding the mechanism and not meet our requirement of visibility from above. There was not a lot of research invested in different types of guides, when discovering the solution of having round shafts with linear bearings would make the mechanism more visible from above. Seen in the figure below an example from Igus, where a design like this have been to great inspiration for the final design of the Rails assembly.

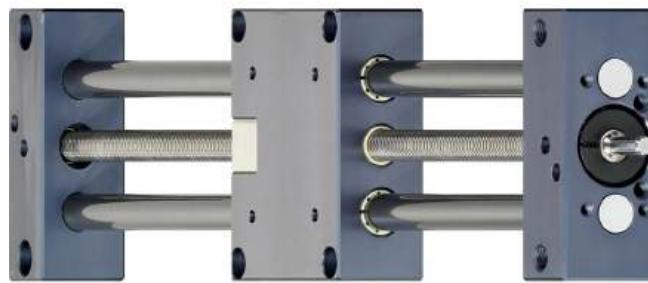


Figure 37: Example from Igus of a linear motion guide with shafts[2]

As seen in the figure above there are parts that are typically off-the-shelf products, like the lead screw and nut, shafts and linear bearings and clamping rings. Parts like these were decided early on in the process to be ordered from Igus, and other components like parts for the middle block, end caps and other parts needed could be designed by us and produced by TE.

In a linear motion system it is very common to implement a stepper motor. When the decision to design a linear motion system was made, an extra servo motor was already purchased to the project. In the two first iterations the motor is connected with an aluminium adapter and shaft coupling, and later an aluminium adapter with gear.

Different design iterations were done through the project to optimize the design for production and due to changing interfaces.

*Design iteration 1:*

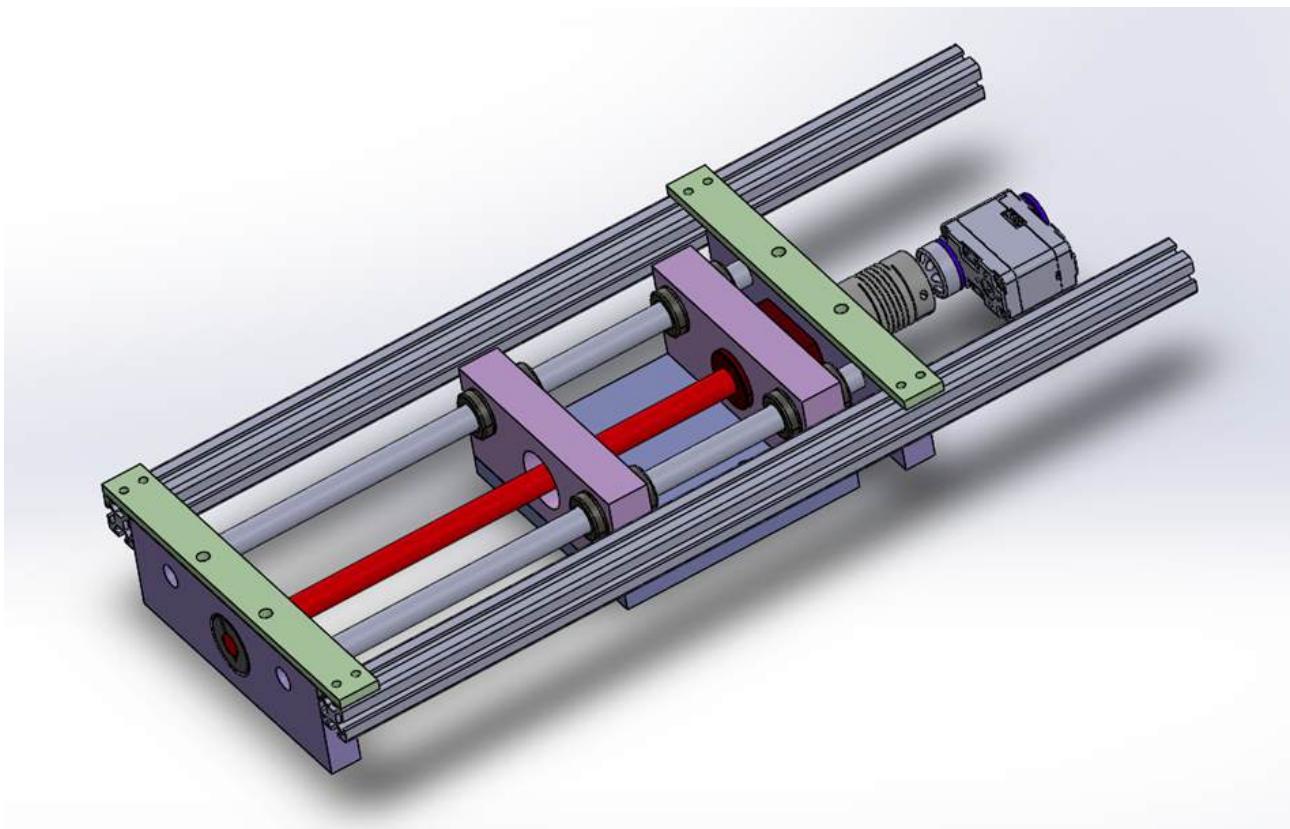


Figure 38: Rails assembly iteration 1

In the figure above is the first iteration of the rail assembly. The bright red part is the lead screw, and the dark red part is the lead screw nut connected to a shaft coupling, connecting the motor and the lead screw. At this first iteration it was not decided how to fix the motor to the and the interface between the fastening hub and bottom plate in the middle block was not decided. Two end caps in aluminium are connected to the Item profiles with an aluminium plate on top. Four linear bearings are connected with two retaining rings each, and are distributing the load on to two aluminium shafts. The distance between the aluminium shafts were aligned with the distance between the motor center holes in the fastening hub.

*Design iteration 2:*

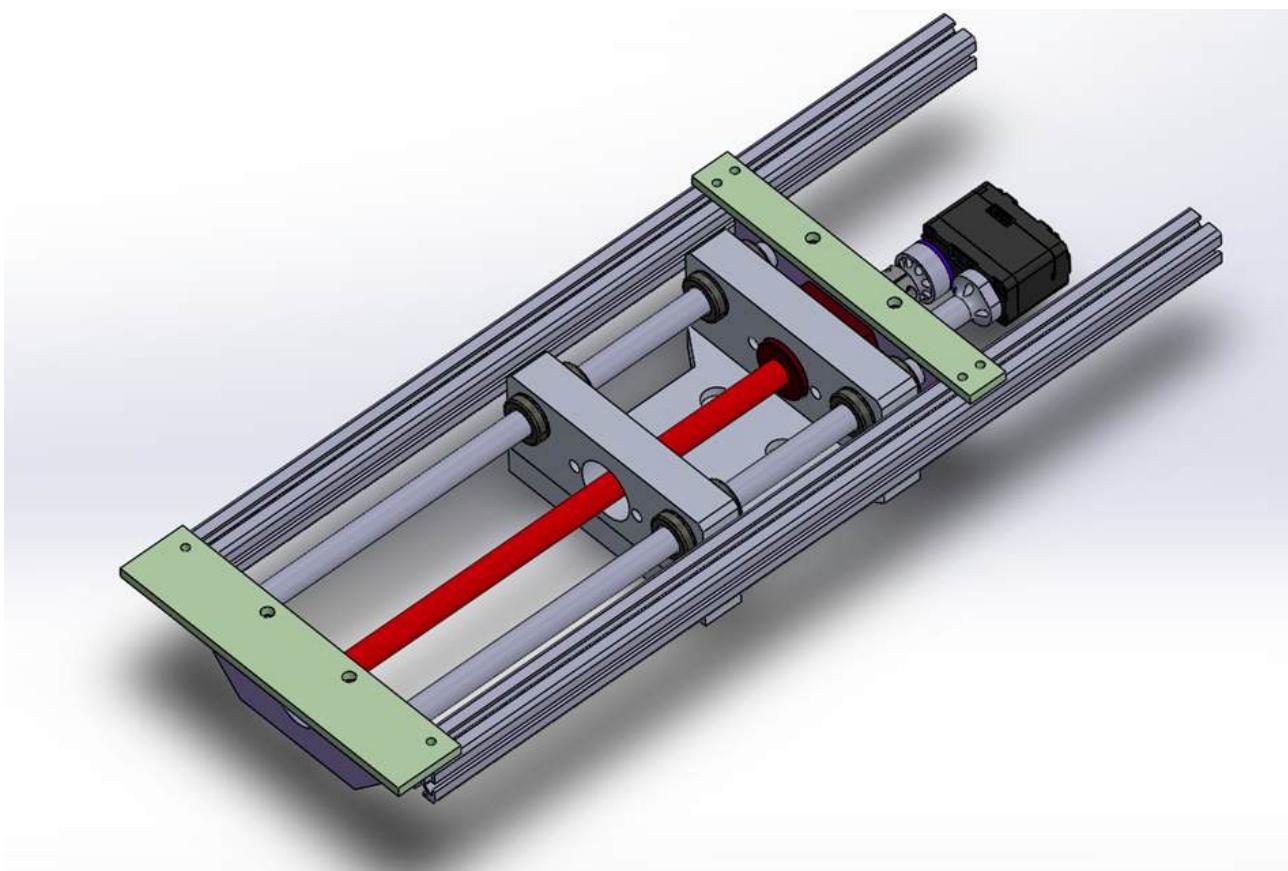


Figure 39: Rails assembly iteration 2

The figure above shows the second iteration of the rail assembly. The lead screw was designed to be machined on both ends, and a smaller shaft coupling was utilized.

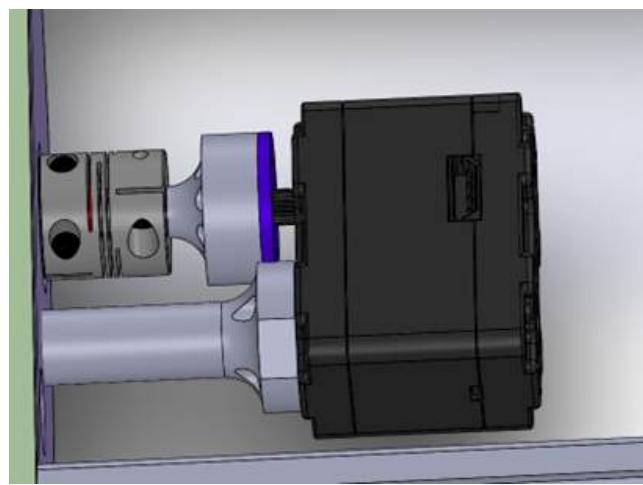


Figure 40: Motor connection rails assembly iteration 2

An adapter connected to the end cap was designed to fix the motor. The adapter allowed free access to the two power inputs in the motor.

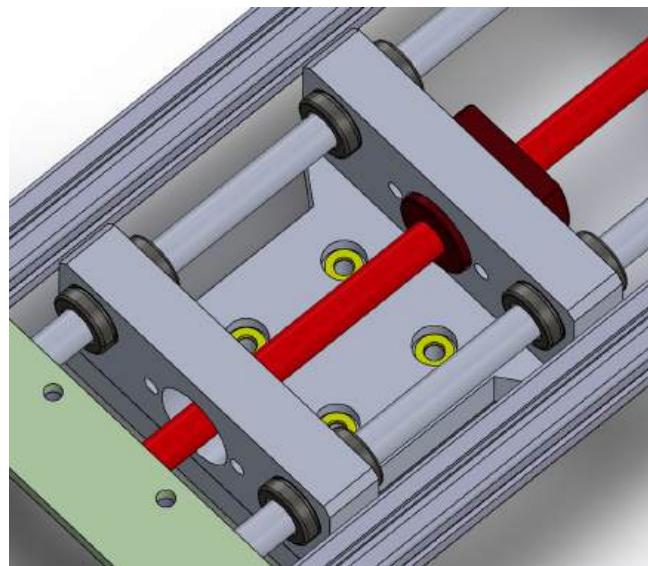


Figure 41: Interface rails assembly iteration 2

Interface to the fastening hub was decided, implementing four clearance holes to M6 bolts in the bottom plate. The sides on the bottom plate was reduced to not interfere with the arm assembly.

*Design iteration 3:*

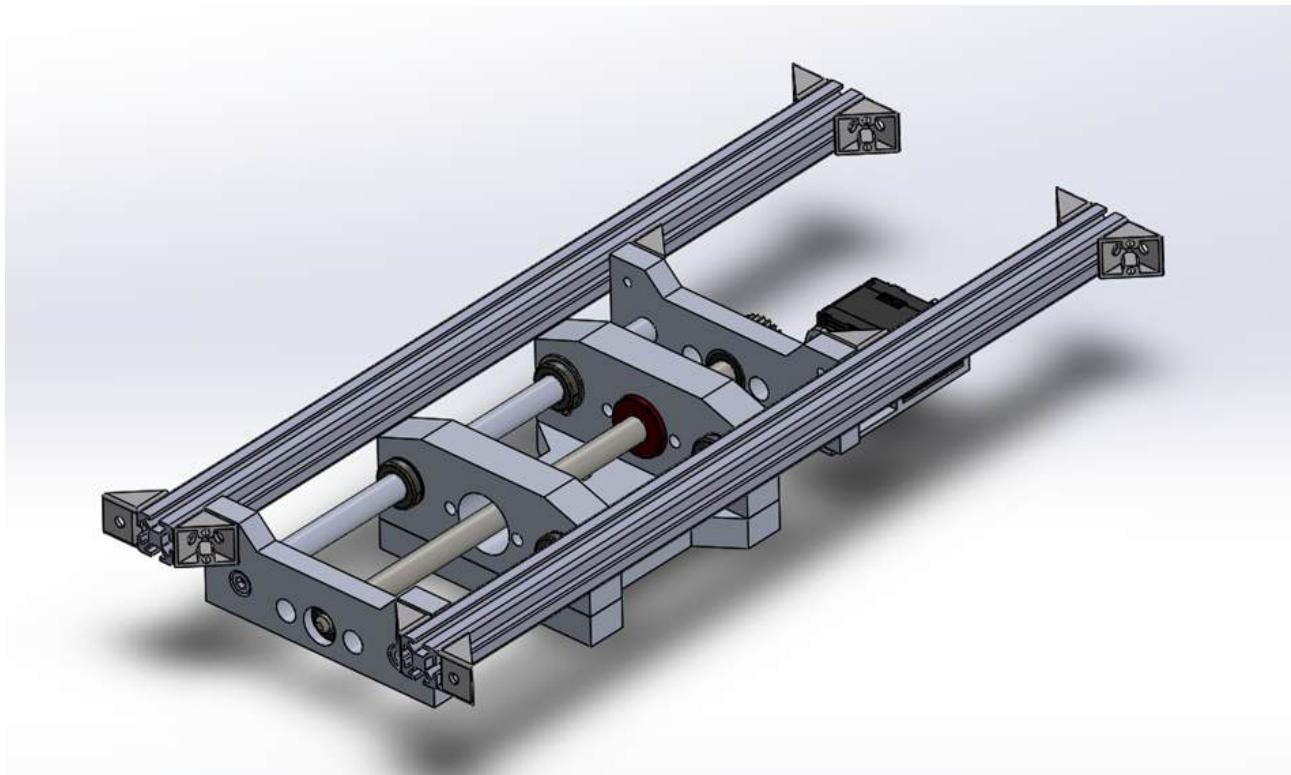


Figure 42: Rails assembly iteration 3

In the third iteration gears were implemented and a new motor bracket was designed. The motor adapter and gears are described further in the chapter fastening-hub assembly. The connection between the end caps and the Item profiles were changed to be connected with angle brackets, not having anything building over the Item profiles. In the figure below there is

a top view of the assembly and how the motor was moved off center of the lead screw to make room for gears.

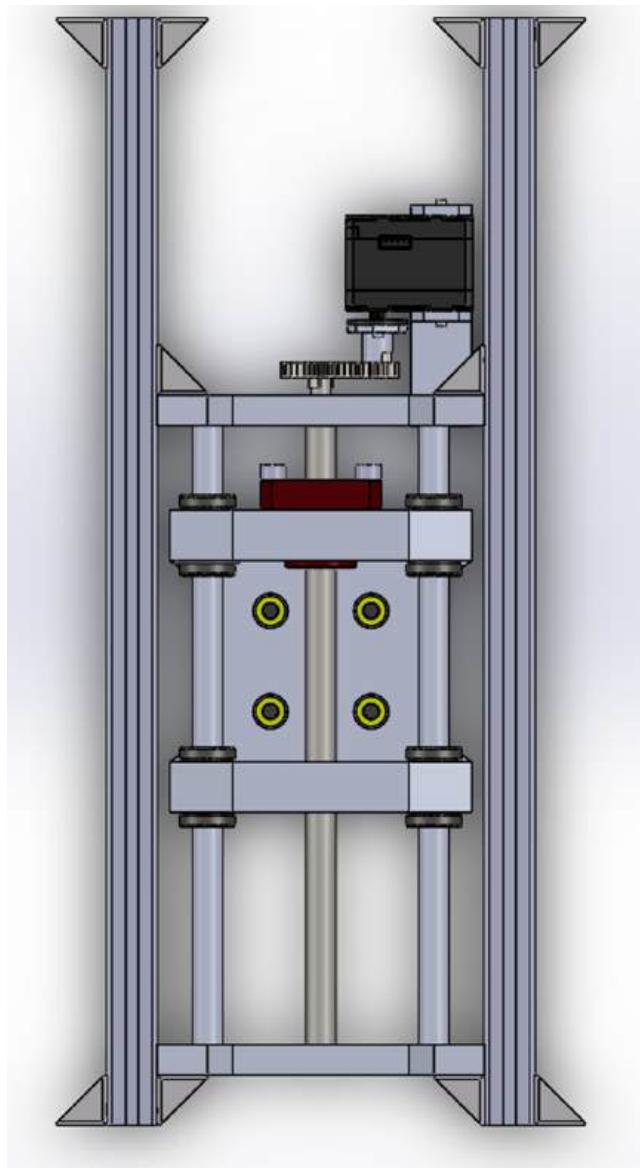


Figure 43: Top view of the rails assembly iteration 3

*Design iteration 4:*

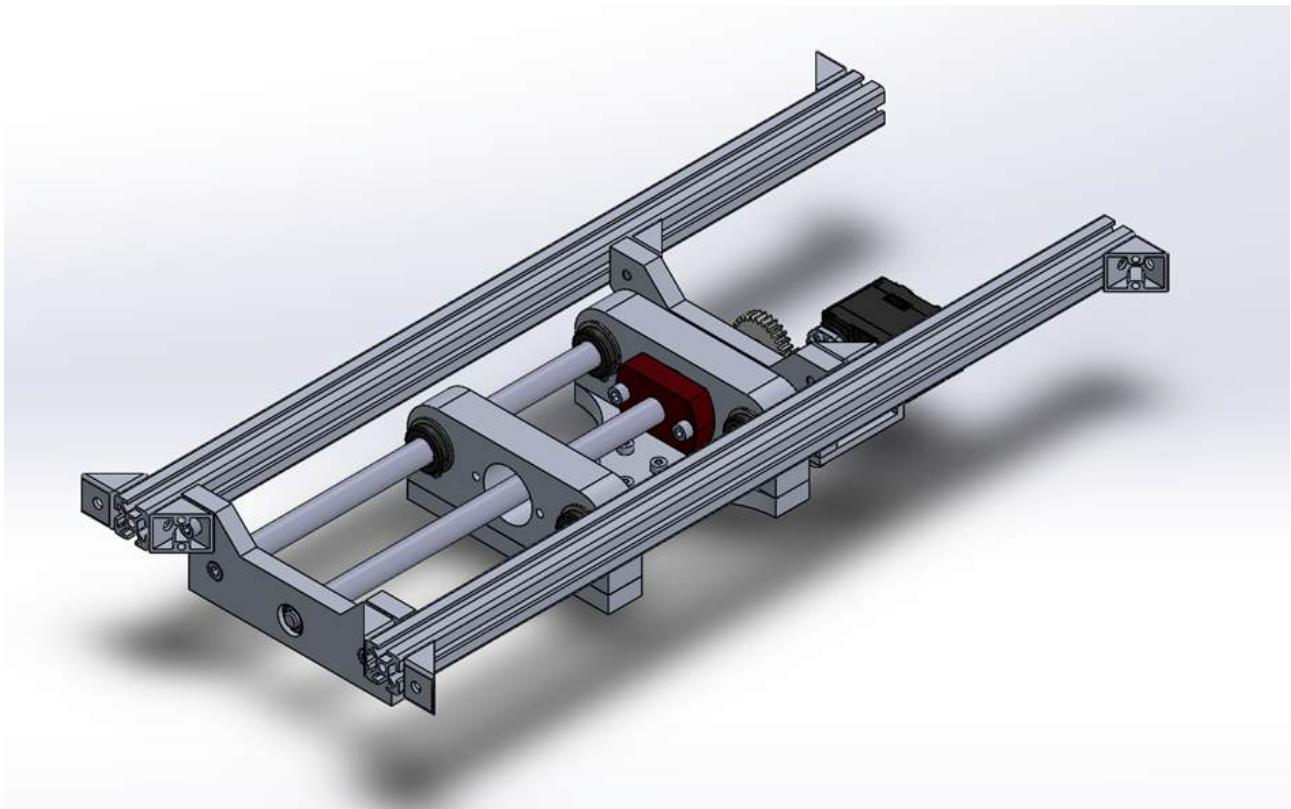


Figure 44: Rails assembly iteration 4

In the figure above is the fourth and final iteration where end caps and middle block were modified to reduce weight. The interface between the assembly and the fastening hub was changed from four M6 bolts to six M4 bolts.

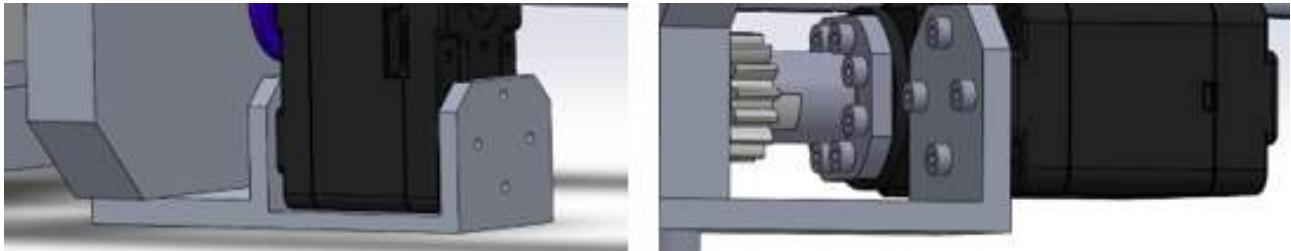


Figure 45: To the left previous motor bracket, to the right new motor bracket

Motor brackets were changed, since the previous design would have a more extensive production method, having a aluminium block milling desired geometry, wasting material and have higher cost. A standard aluminium angle modified with clearance holes and chamfers was chosen to be the new bracket to fasten the motor.

In the chapter below you find the full overview of the parts in the final rails assembly, and if it is ordered or manufactured.

## Bill of Materials

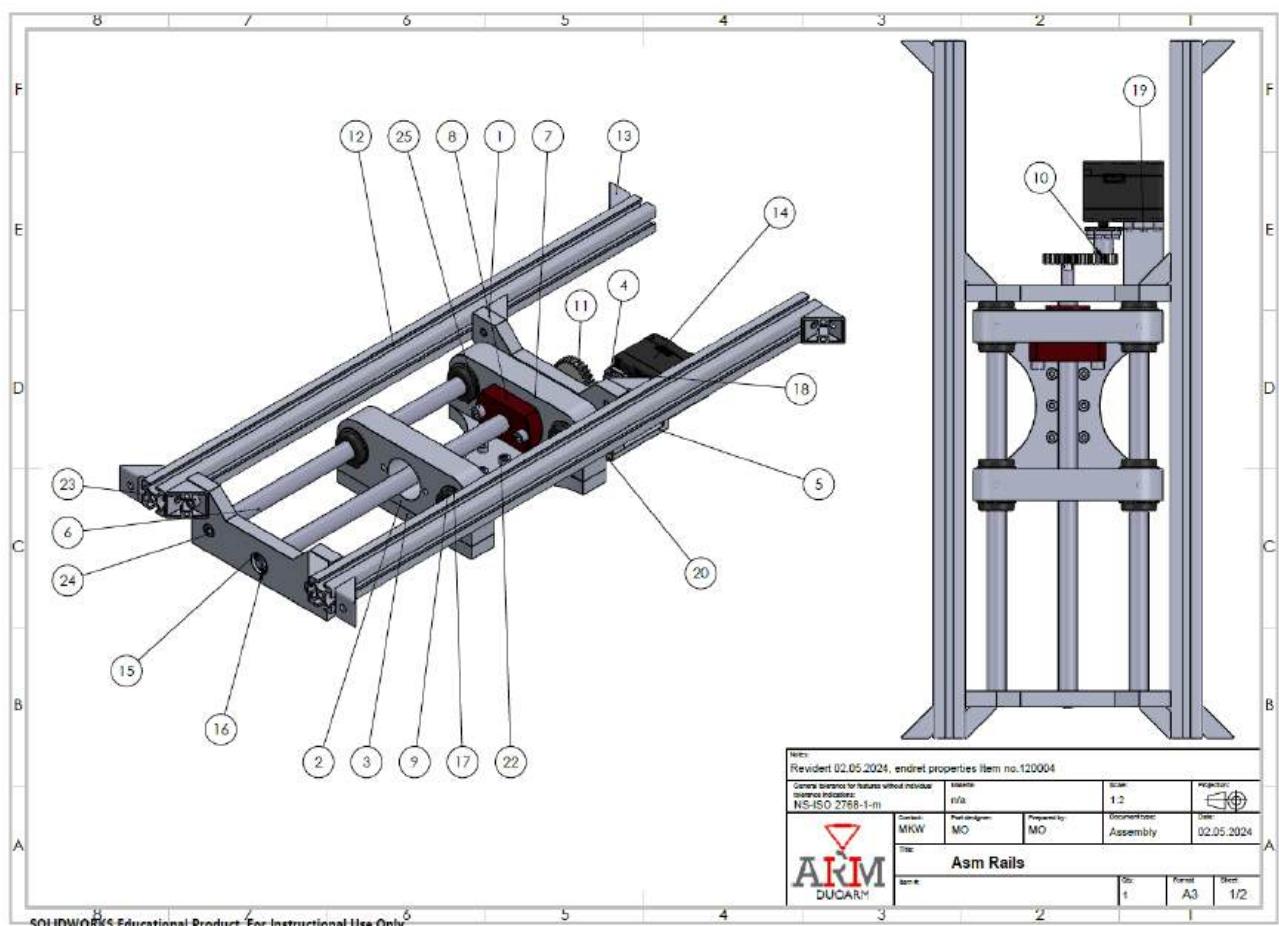


Figure 46: Parts in the Rails assembly

Shown above is a view of the Rails assembly with all its components.

Table 11: Bill of Materials for the rails assembly

Pos.	Title	Item no.	Material	Manufactured	Supplier	Article name	Article no.	Size	QTY.	
1	Endfeste	120001	6063-T6	Yes					2	
2	Feste Linear Bearings eg LS Nut	120002	6063-T6	Yes					2	
3	Bunnplate Rail	120003	6063-T6	Yes					1	
4	AdapterDrivhjul	100000	6063-T6	Yes					1	
5	MotorBraket Rails	120004	6063-T6	Yes	Astrup	Alu vinkler like/ulikebent	0010340141	50x30x4	1	
E	6	Aluminium shaft	120005	6063-T6	Yes	IGUS	drylin® R hard-anodized, aluminum shaft, AWM	AWM-12	Dia: 12mm, L: 250mm.	
7	Lead Screw 12-3	120006	Plain Carbon Steel	Yes	IGUS	dryspin® trapezoidal lead screw, right-hand thread, stainless steel	PTGSG-12X3-01-R-E5	12-3 L(300)	1	
8	Lead Screw NUT 12-3	320001	n/a	No	IGUS	dryspin® flange lead screw nut with spanner flat, trapezoidal thread, RFRM	RFRM-282835TR12X3	12-3	1	
9	Linear Bearing 12	320003	6063-T6	No	IGUS	drylin® R linear slide bearing RJUM-01	RJUM-01-12	12	4	
D	10	Grip15T	100001	ABS	No	DuoArm			1	
11	Grip30T	100002	ABS	No	DuoArm				1	
12	Item Profile 20x20x440	310003	6061-T6 (SS)	No	Item 24	Profile 5 20x20	0.0.448.04	20x20	2	
13	CornerBracket	310004	6061-T6 (SS)	No	Item 24	Angle Bracket 2n	0.0.425.03	20x20	8	
14	Motor HS1	300000	n/a	No	Robotshop	Lynxmotion SES-V2 High Speed Smart Servo (LSS-HS1)	RB-Lyn-988	HS1	1	
C	15	Ball Bearing 608	320002	Plain Carbon Steel	No	Misumi	Deep groove ball bearings	608-2RS	608 (8/22)	2
16	Retaining Ring C8	320004	Spring Steel	No	Misumi	Retaining Rings / External / C-Type	STWN8	8	2	
17	Retaining Ring C22	320005	Spring Steel	No	Misumi	Retaining Rings / External / C-Type	STWN22	22	8	
18	M2x0.4 L5	300001	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M2-5	M2x0.4, L5	8	
19	M2x0.4 L10	300002	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M2-10	M2x0.4 L10	4	
20	M4x0.7 L12	300003	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M4-12	M4x0.7 L12	4	
21	M4x0.7 L20	300005	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M4-20	M4x0.7 L20	8	
22	M4x0.7 L25	300006	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M4-25	M4x0.7 L25	6	
23	M5x0.8 L12	300010	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M5-12	M5x0.8 L12	4	
24	M5x0.8 L15	300011	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M5-15	M5x0.8 L15	4	
B	25	M5x0.8 L32	300012	Steel	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M5-32	M5x0.8 L32	2
A						General Drawing for feature without individual dimension information NS-ISO 2768-1-m	Rev:	1.2		
						Part designer: MRW	Prepared by: MO	Document type: Assembly	Date: 02.05.2024	
						ITEM:	Asm Rails			
						Part #:	1	Format:	Sheet	
						QC:	A3	Page:	2/2	

SOLIDWORKS Educational Product. For Instructional Use Only.

The bill of materials is showcasing each part used in the assembly. This includes its unique item number, material, if its manufactured or ordered as well as supplier, article name, article number, size, and the total quantity of each part.

### 6.1.7 Fastening-hub assembly

SG | AK

#### Introduction:

The fastening hub main assembly is composed of three different sub-assemblies. These create the fastening hub main assembly that includes all components mentioned in the sub-assemblies bellow.

#### Sub-assemblies:

##### Fastening-hub sub-assembly:

Introduction:

The fastening hub subassembly is consist of five different components including ball bearings press fitted into a milled circular space and M6x1,0 coarse threaded bolts. [86] The remaining three parts makes up the hub itself and creates a housing for the axle subassembly and mounting point for fastening bracket subassembly.

*Design and development:*

This chapter is about the design throughout the development process.

*Design iteration 1:*

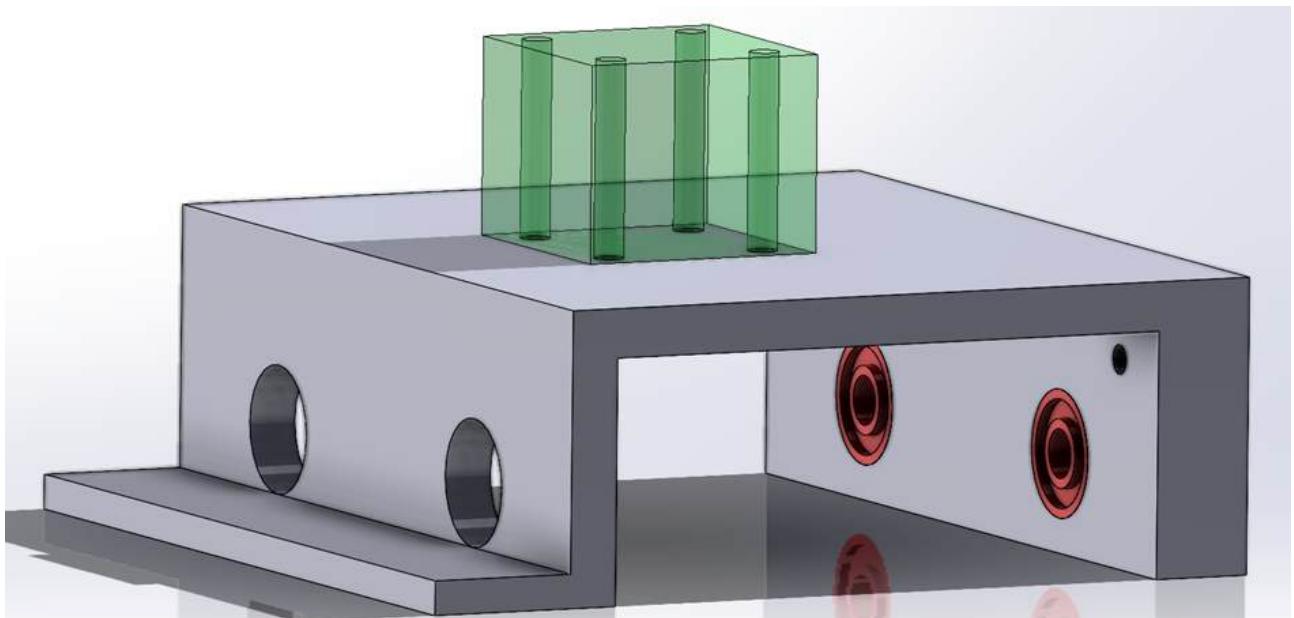


Figure 47: Fastening hub iteration 1

Shown above is the initial fastening hub design which is connected to the rail system. It was planned to have two ball bearings assuring axle rotation shown to the right in red. Additionally, a mounting hole for the upper stabilizer arm on the top right side. The transparent green box is a simplified presentation of the rail system with the positioned threaded mounting holes. Furthermore, on the left side is a shelf where the motors are supposed to rest, with the engine drive wheel connected to the axle inside the open holes. This presented an option to retain the shaft from axial movement, either by mounting the LSS-HS1s directly to the back plate or by using retaining rings on either side of the axle inside the open gap.

*Design iteration 2:*

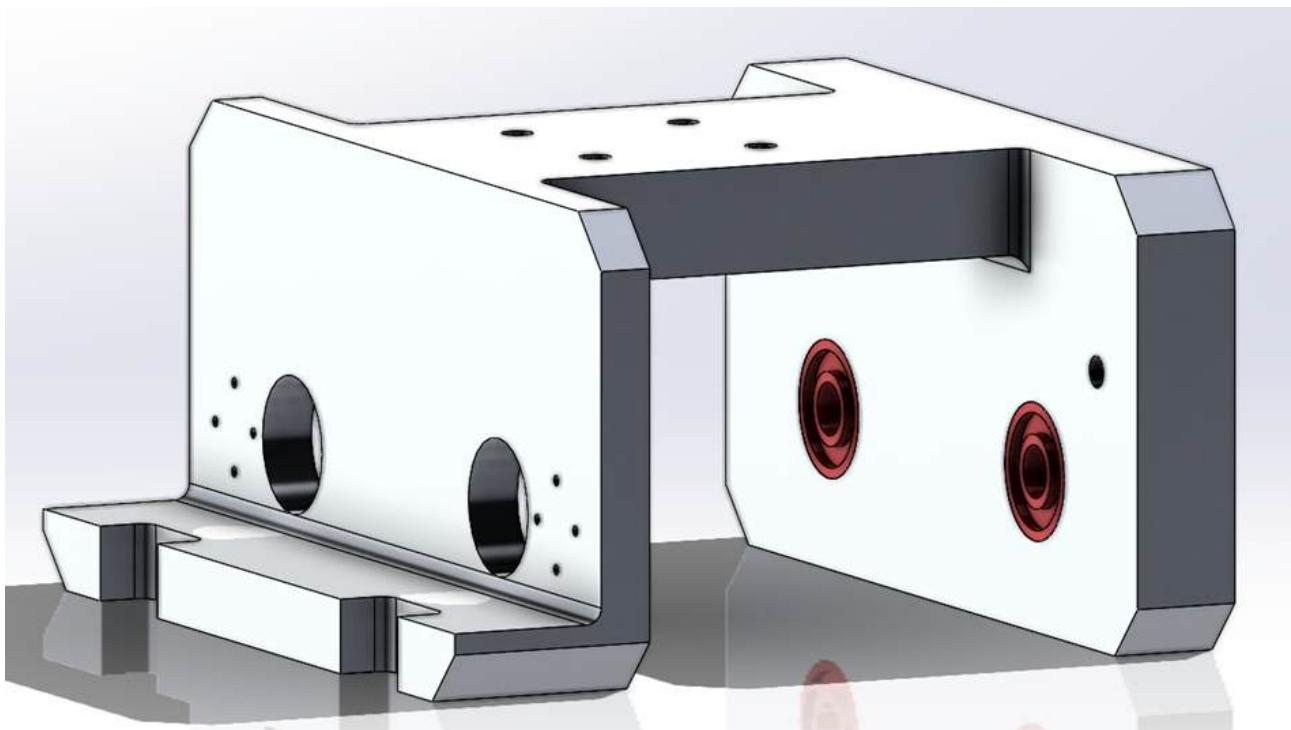


Figure 48: Fastening hub iteration 2

The next iteration included M2x0,4 coarse threaded mounting holes for the motors. In addition, an area of the shelf removed pertaining a potential serial connection between the engines. Resulting in less cables connected to the control board. Improving the cable management within the system to improve the overall tidiness and reduce the risk for tangling. Furthermore, parts of the top plate were removed to further decrease mass, additional chamfers implemented on sharp edges and fillets on inside corners. The fillets are derived the milling tool radius and desired due to a reduction of stress concentration in these areas.

*Design iteration 3:*

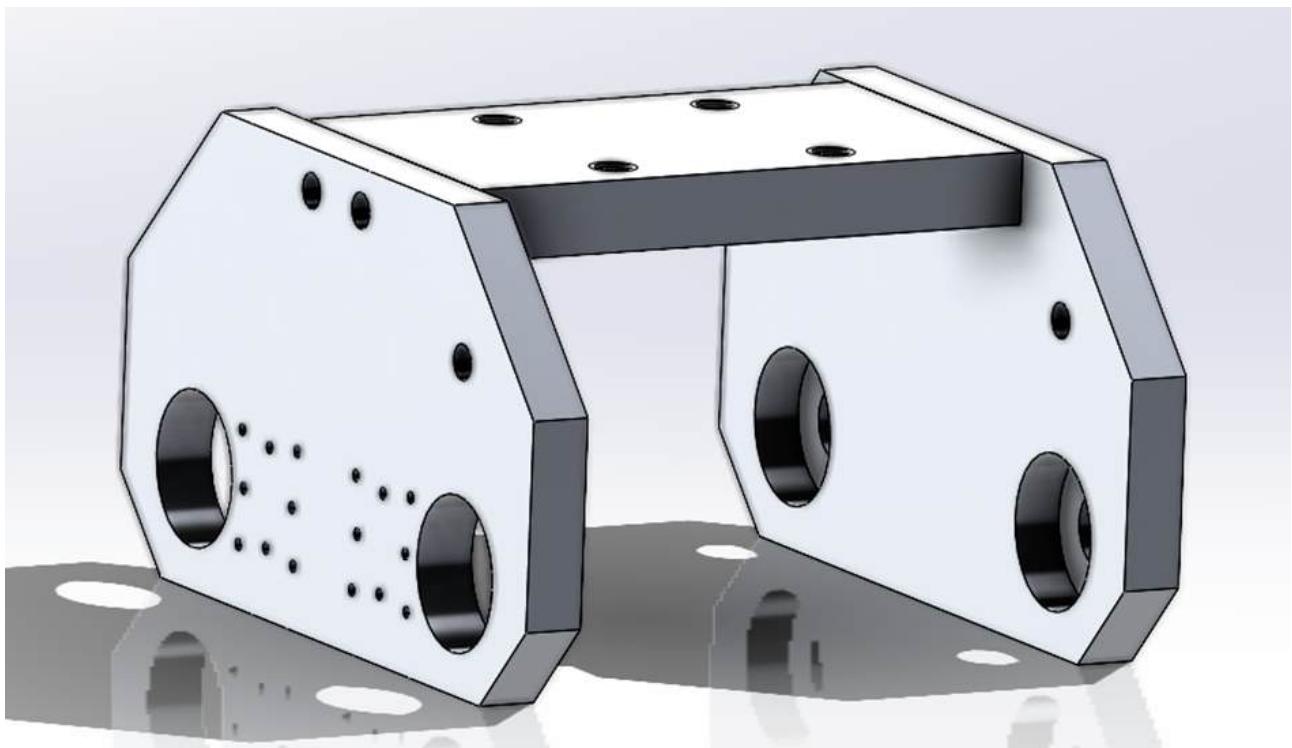


Figure 49: Fastening hub iteration 3

In the following iteration, the hub was split into three pieces to reduce waste material drastically, improving cost-efficiency and machine time. The previous iteration had to be machined out of a solid block of aluminum. This iteration simplified the process thru machining three separate sheets, bolting them together using M6x1,0 machine bolts screwed in the center parts coarse tapped holes. Additionally, the motor shelf was removed and extra mounting holes for the LSS-HS1 added to the back-plate. Resulting in a reduced bolt shearing probability under load. Furthermore, an increase in chamfers for additional mass reduction, removal of fillets made by the milling tool positioning largest stress concentrations at the bolts. Lastly, an extra mount for stabilizer was added regarding stability considerations by utilizing two.

**Mass properties of Festehub5**  
**Configuration: Default**  
**Coordinate system: -- default --**

**Density = 0.00 grams per cubic millimeter**

**Mass = 661.77 grams**

**Volume = 245100.00 cubic millimeters**

**Surface area = 70294.43 square millimeters**

**Center of mass: ( millimeters )**  
X = 0.68  
Y = 9.14  
Z = -0.10

Figure 50: Fastening hub iteration 3.1

At this stage the systems mass had to be decreased as much as possible leading thinner walls. Thickness was reduced to a minimum at 10mm because of the ball bearings unchangeable thickness at 8mm. The mid plate remained a bit thicker, ensuring enough threads to tightly fasten the hub to the rail with a preload. Linking connected assemblies to the frame.

*Design iteration 4:*

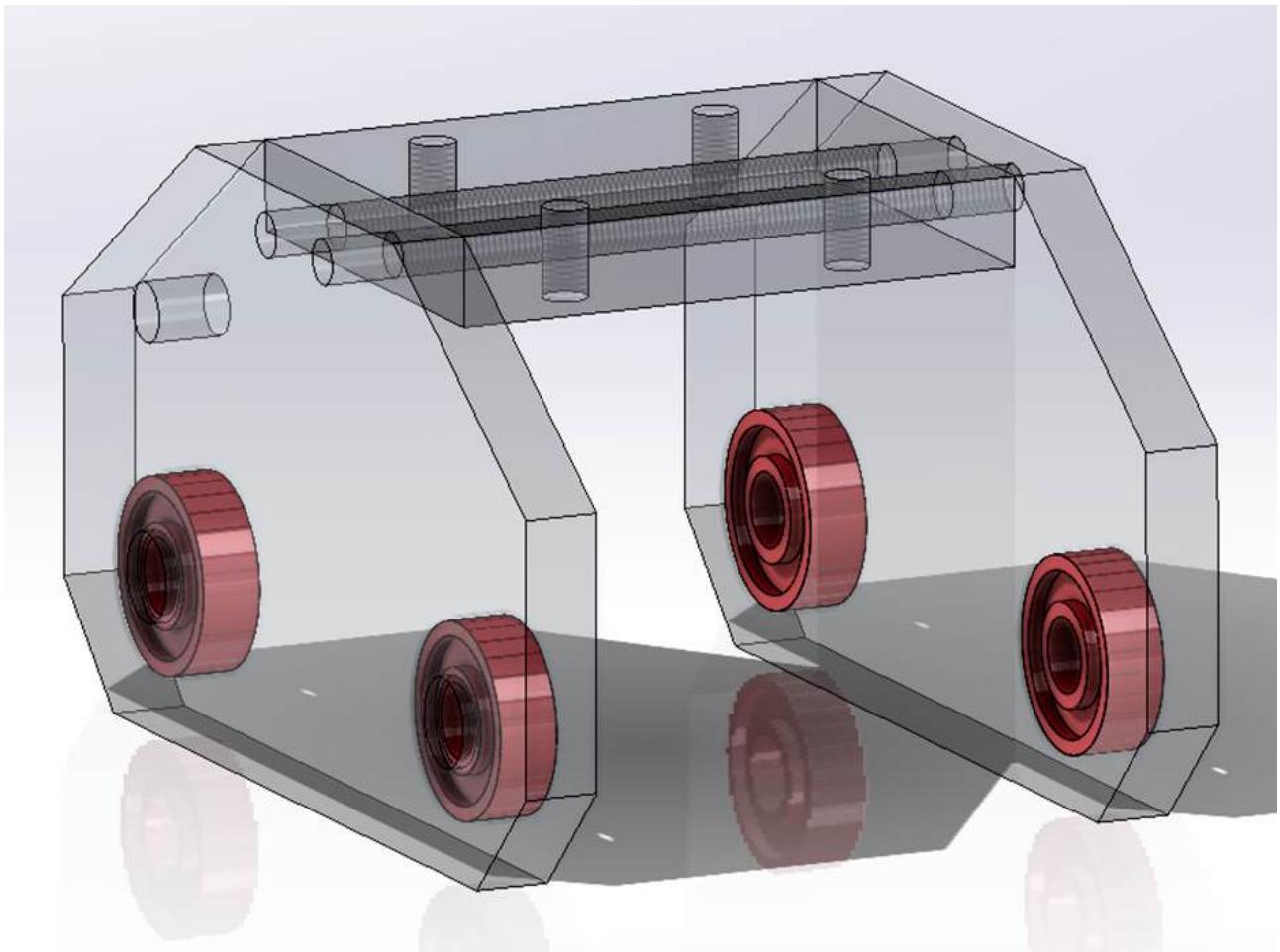


Figure 51: Fastening hub iteration 4

Despite mass reduction, two additional ball bearings were added on the opposite side assuring evenly distributed load across the axle. The LSS-HS1 drive wheel gear is a concerning weak point regarding stress concentrations. Adding these bearings would remove unnecessary stress on the gear linked to the engine internals. It could potentially lead to reduced component lifespan and increases the risk of fatigue failure. Furthermore, the threaded holes through mid-plate joining the three parts together are not being threaded all the way through. This was done as a simplification and visualization until the final width between front and back-plate was determined.

*Design iteration 5:*

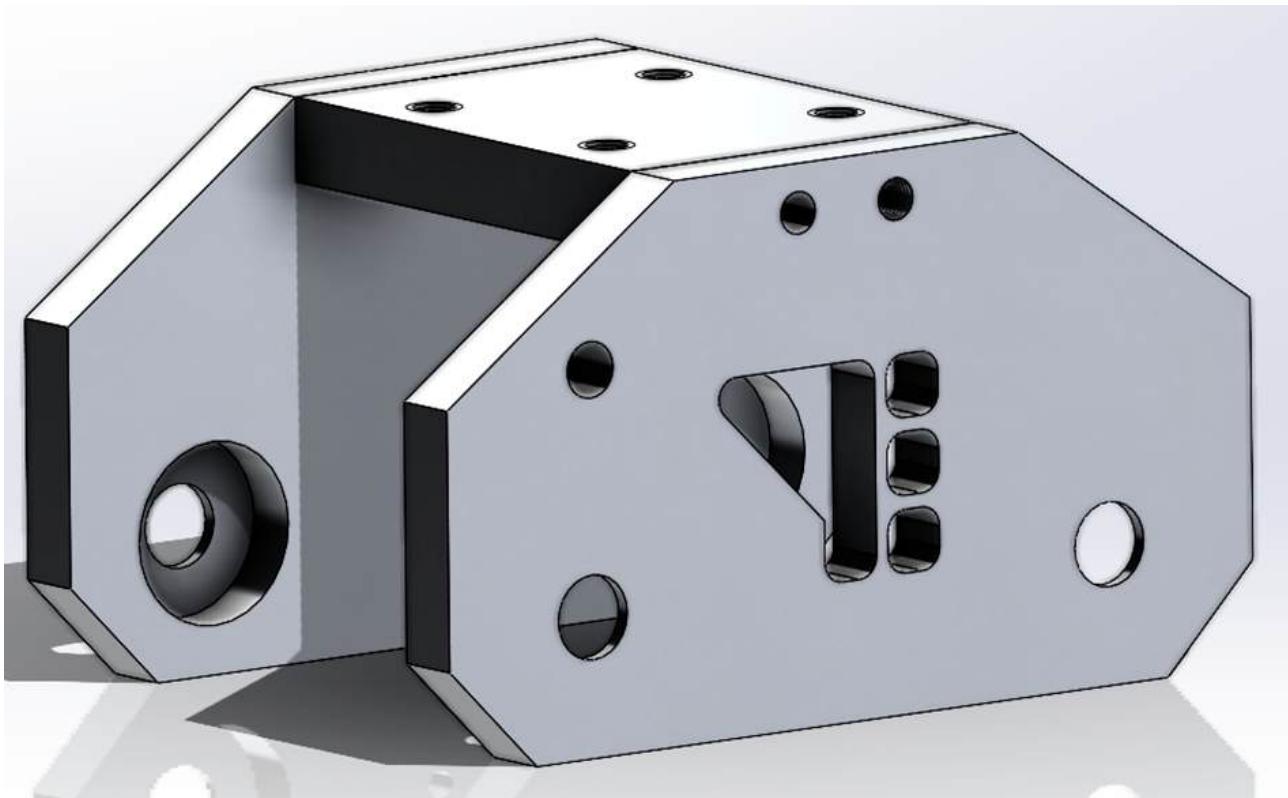


Figure 52: Fastening hub iteration 5

After bearing slots, a smaller hole at 14mm in diameter centered in the circular indent was added to avoid friction between the plates and axle ends. Assuring that the axle could rest and rotate safely without complications within the bearing. In addition, holding the beneath linked systems center of mass at axle midpoint. TE logo was then cut out on the front for display. It has a fillet radius of 3mm generated by the pin mill normally used. The reason behind not having the correct logo but it intended to be an advertisement. Furthermore, the stabilizer mounting position was also determined and changed back to being on one side. Since having two stabilizers would decrease the required lifting height. Caused by an interference between the active arm and stabilizer pin.

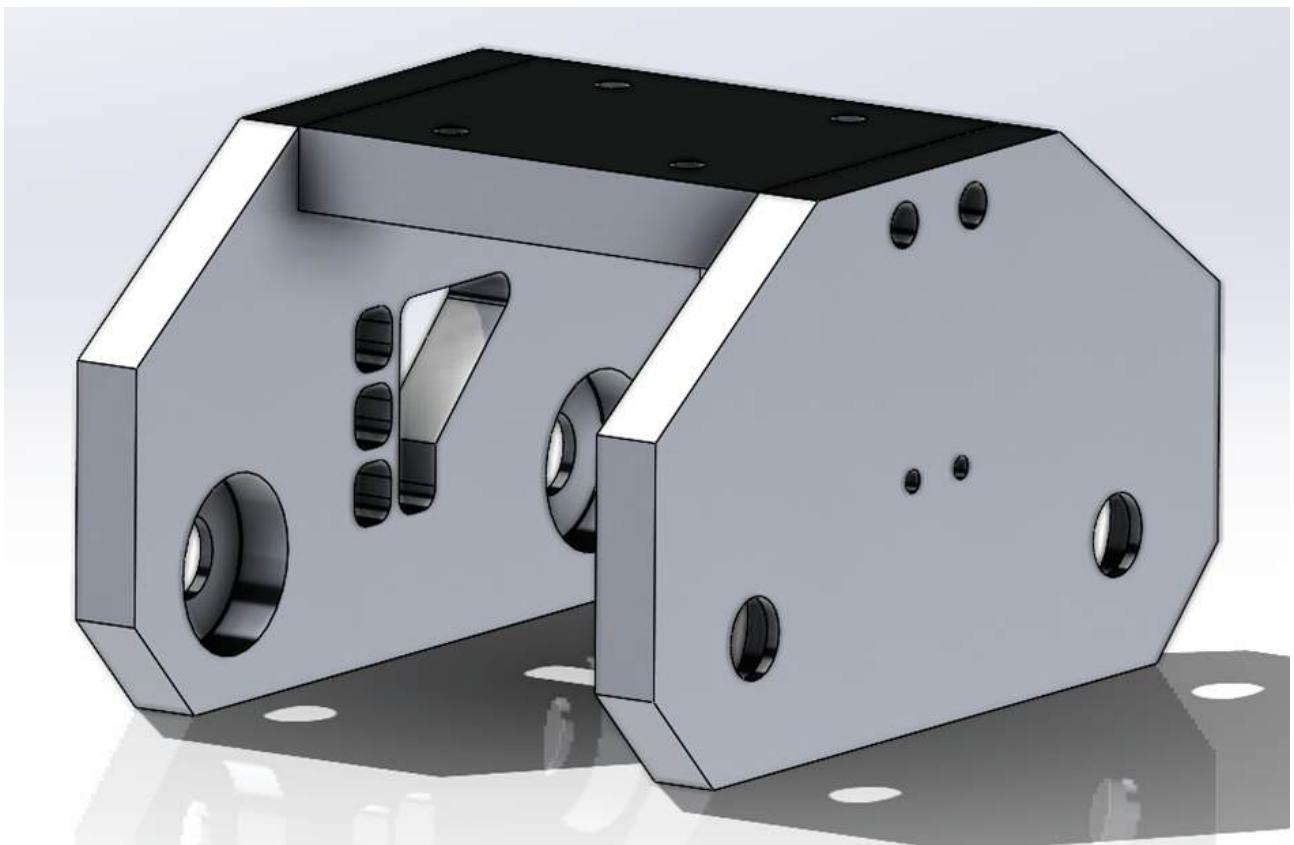


Figure 53: Fastening hub iteration 5.1

A new solution for mounting the motors added to the design. It was a necessity since we figured out at this point that the LSS-HS1 required gears. These had to be mounted on the axle and motor at a spot, fitting each others pitch diameter assuring efficient power transmission. The solution for engine mount turned out to be a bracket, bolted to the back plate in 2 x M4x07 coarse threaded holes. The bracket hub is a sub-assembly of its own.

*Design iteration 6:*

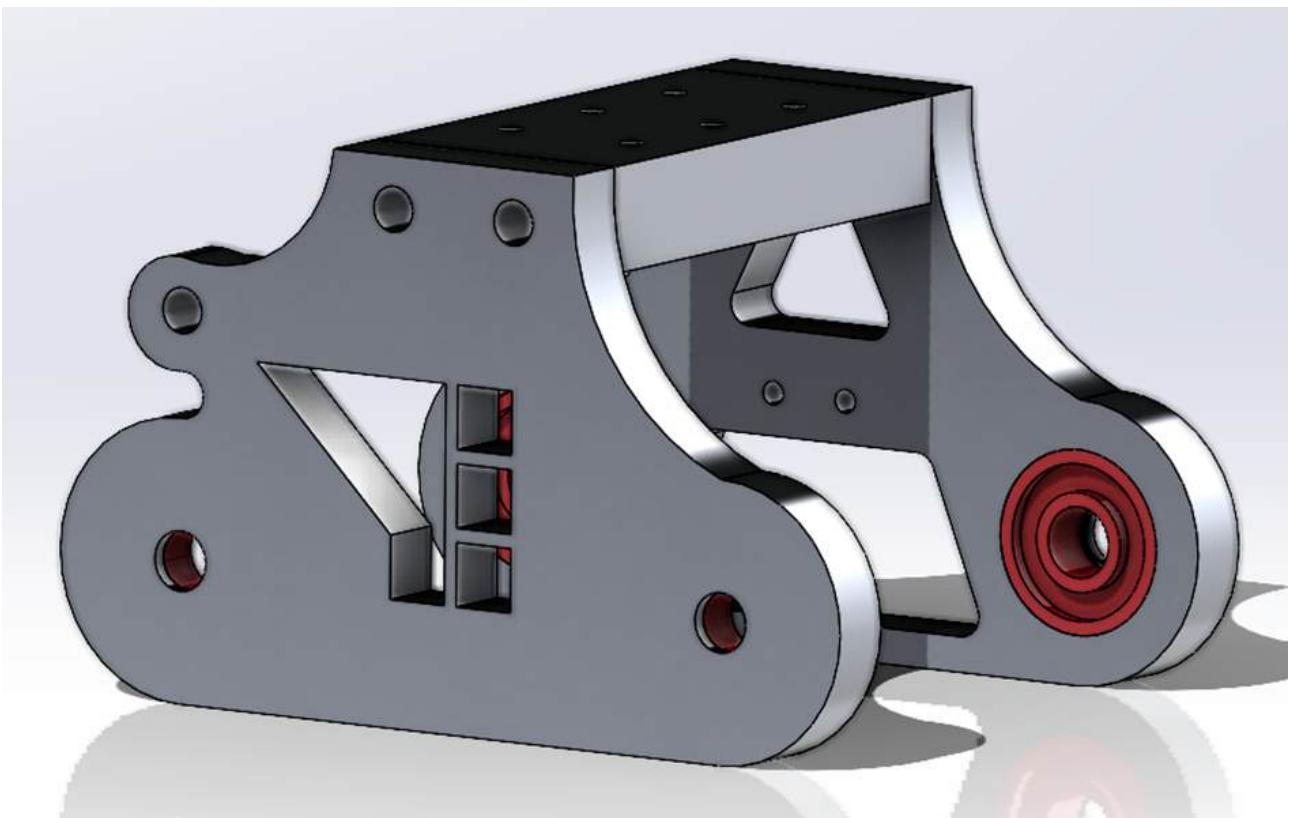


Figure 54: Fastening hub iteration 6

The last iteration initiated a new design from scratch. Acquiring more freedom since parts are cut utilizing a water-jet and bolted together. Providing the opportunity for more advanced shapes, accurate TE logo and mitigating production time and cost. It was made possible since all parameters for the dependent components was determined. Additionally, diminishing mass without compromising structural integrity.

<b>Mass properties of SubAzm Festehub</b>	
<b>Configuration:</b>	<b>Default</b>
<b>Coordinate system:</b>	<b>-- default --</b>
<b>Mass = 411.21 grams</b>	
<b>Volume = 161597.07 cubic millimeters</b>	
<b>Surface area = 63185.57 square millimeters</b>	
<b>Center of mass: ( millimeters )</b>	
<b>X =</b>	<b>31.11</b>
<b>Y =</b>	<b>12.35</b>
<b>Z =</b>	<b>12.74</b>

Figure 55: Fastening hub iteration 6.1

Furthermore, the mid plate thickness was increased ensuring enough grip and decreasing bolt

tearing risk. Additionally, added two more M4x07 coarse threaded holes to even out forces further shrinking shearing risk at the rail-fastening hub interface. In hindsight, thickness should be reduced by another 5mm, and the bolts secured utilizing nuts. M6x1,0 bolts had then been reduced to M4x0,7 additionally reducing mass.

*Exploded view and Bill of Materials:*

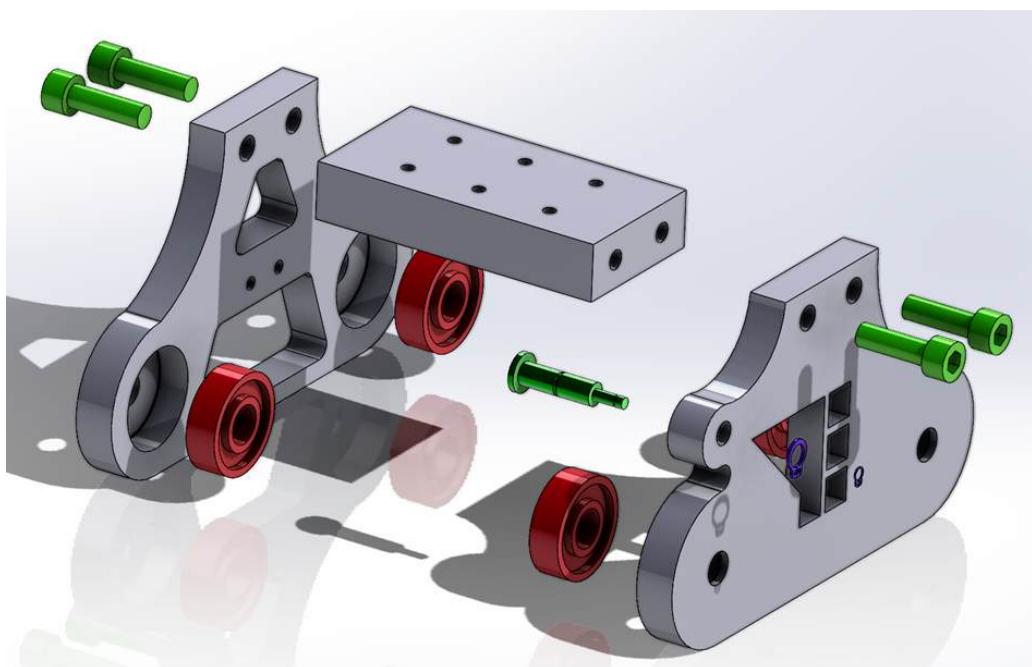


Figure 56: Fastening-hub exploded view

Shown above is an exploded view of the sub-assembly with all its components.

Table 12: Fastening-hub Bill of Materials

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Part name	Article no.	Size	QTY
1	Frontside	130006	6063-T6	154.08	Yes					1
2	Midt	130005	6063-T6	122.05	Yes					1
3	Bakside	130004	6063-T6	118.32	Yes					1
4	M6x1.0 L20	300007	Steel	0.97	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M6-20	M6x1.0 L20	4
5	BallbearingSteel	330002	Steel	2.71	No	Misumi	Deep groove ball bearings / single row	6000VV	OD = 26mm, ID = 10mm	4
6	RetainingringØ3	300031	Spring steel	0.00	No	Misumi	Retainingring C-Type	STWN3	Nominal Ø3mm	1
7	stabilisator adapter FESTEHUB	130010	6063-T6	1.99	Yes					1
8	RetainingringØ6	300032	Spring steel	0.02	No	Misumi	Retainingring C-Type	STWN6	Nominal Ø6mm	1

The bill of materials is showcasing each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

*Parts ordered:*

*Retaining rings:*

These are ordered from MiSUMi. Further details can be found in Bill of Materials. [87]

*Ball bearing:*

These are ordered from MiSUMi. Further details are found in Bill of Materials. [88]

*Bolts:*

These are ordered from MiSUMi. Further details are found in Bill of Materials. [50]

*Parts produced:*

*Fastening-hub sheets:*

The front, middle and back sheet contour is cut out utilizing a water-jet, also includes logo, material removal on the back sheet, threaded holes, excluding blind ones and milled areas.

Tapped holes linking sheets on mid plate must be drilled using a smaller drill bit than the final hole size and then threaded. Furthermore, the cavity for bearings is required to be milled utilizing a pin mill.

### *Stabilisator adapter FESTEHUB:*

The hinge pins are produced using a lathe an operation known as a turning. In this operation the work-piece is rotating while the tool is stationary mounted on the tool-holder. Tool is then fed lengthwise on along the work-piece at a predetermined depth and feed rate. Material is then removed at a rate known as MMR which is dependent on depth of cut, feed, and the rotational surface speed of the work-piece. The process is repeated until desired diameter is achieved. Furthermore, the groove for retaining ring is cut and the component is parted away from its cylindrical start piece at the hinge pin head. Additional, beginning cuts are often rough cuts, typically at high feed rates and larger depths followed by finish cuts at lower feed rates and depths to produce a good surface finish with acceptable tolerances.

### *Fastening-bracket sub-assembly:*

#### *Introduction:*

The fastening bracket sub-assembly consists of bracket with two LSS-HS1. In addition, an adapter with the minor-sized gear fixed to the motors. This sub-assembly mounted on the hub back plate and adjusted to make sure that the gear pitches are tangent.

#### *Design and development:*

This chapter is about the design throughout the development process.

#### *Bracket design iteration 1:*

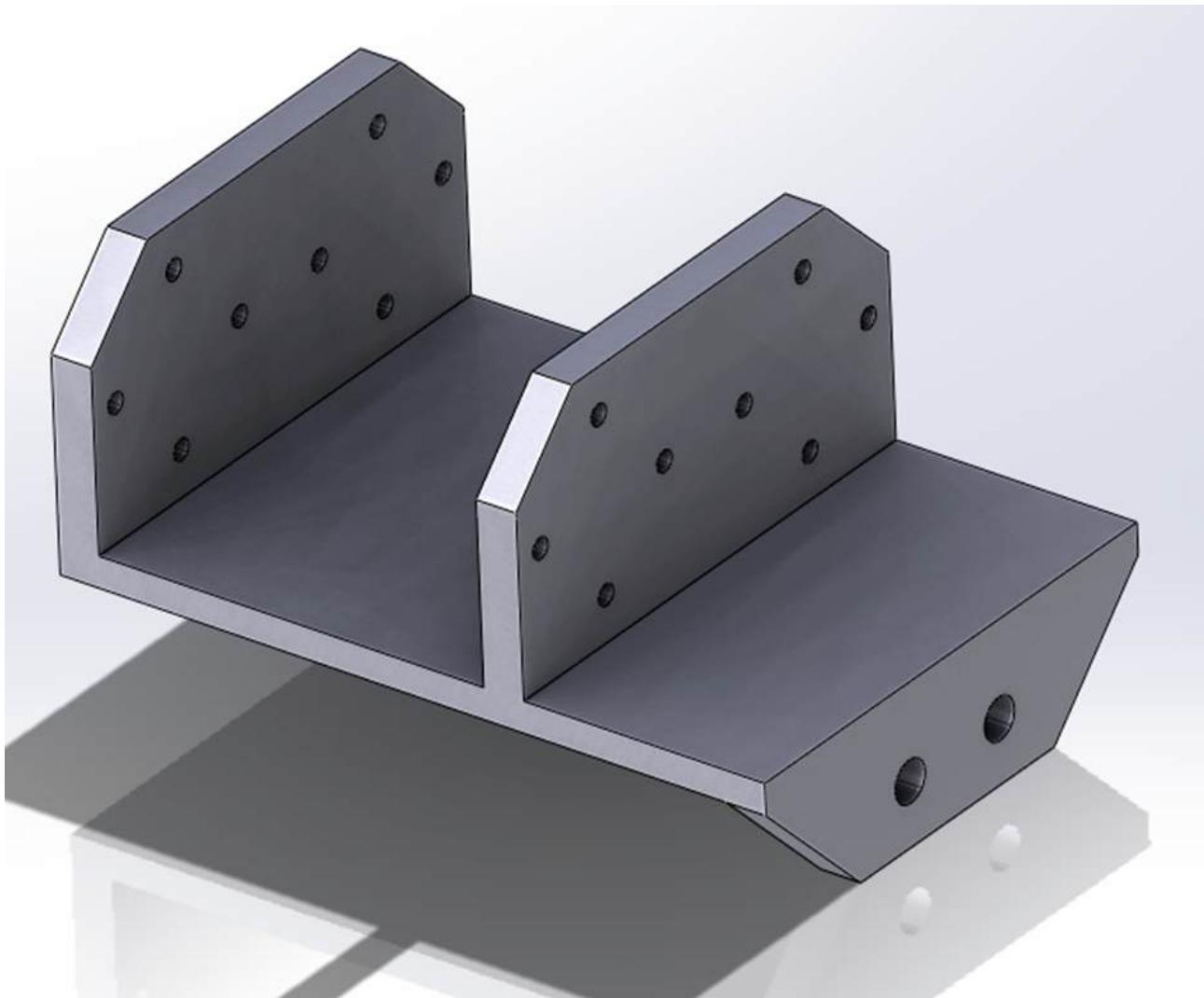


Figure 57: Bracket iteration 1

The initial idea was to mount the engines on a shelf, but this had to be altered due to various challenges resulting in the creation of a fastening bracket. In the first iteration, two LSS-HS1s was planned to be screwed in place using 16 x M2x0,4 coarse threaded bolts. The bolts had to be coarse since the motors had coarse tapped fittings. In addition, the circular profile for the holes had to be Ø16,66248mm, since LSS-HS1 is designed using the imperial measurement system. The hole linking bracket and hub is Ø4 with clearance hole utilizing the hole-wizard function in SW. This function allows for easy adjustment to clearance where threaded bolts are going through. It also has other clever options, like drilled and threaded holes. After positioning mounting points, all sharp edges were chamfered bringing about decreased mass and enhancing esthetics. Furthermore, this design produces a fair amount of waste material when machined, resulting in an alternative iteration.

*Bracket design iteration 2:*

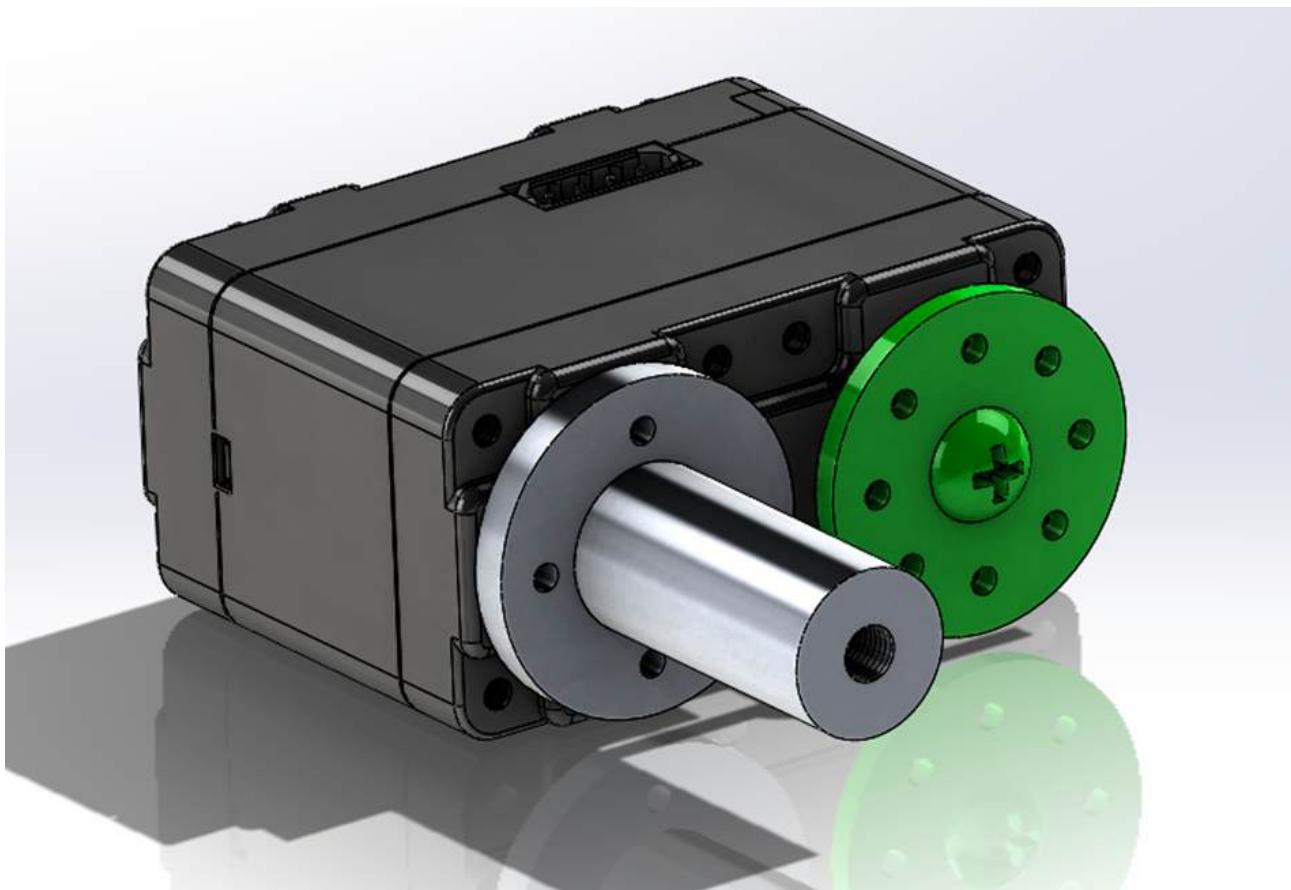


Figure 58: Bracket iteration 2

The alternative design was initially thought to fasten the LSS-HS1 to the back plate by screwing an M4x07 coarse bolt into the tapped adapter hole. The adapter intended to be machined from a cylindrical shaft by using a lathe and then screwed to the motor. This design does not prevent the adapter bolt from radial motion which would eventually be problematic. The M4x0,7 bolt could unscrew itself, even if it was fitted with a preload. A solution to increase grip between internal adapter threads and bolt could have been red Loctite to elevate resistance. This is not a viable solution led to another iteration.

*Bracket design iteration 3:*



Figure 59: Bracket design iteration 3

As the first iteration, a classic bracket was designed. In comparison to iteration 1, the mid mounting wall got removed since it is not necessary to use more than four bolts per motor. The rightmost mounting wall turned upside down, decreasing amount of waste material. The plate is 4mm thick and could be produced using different processes discussed further down.

*Adapter design iteration 1:*

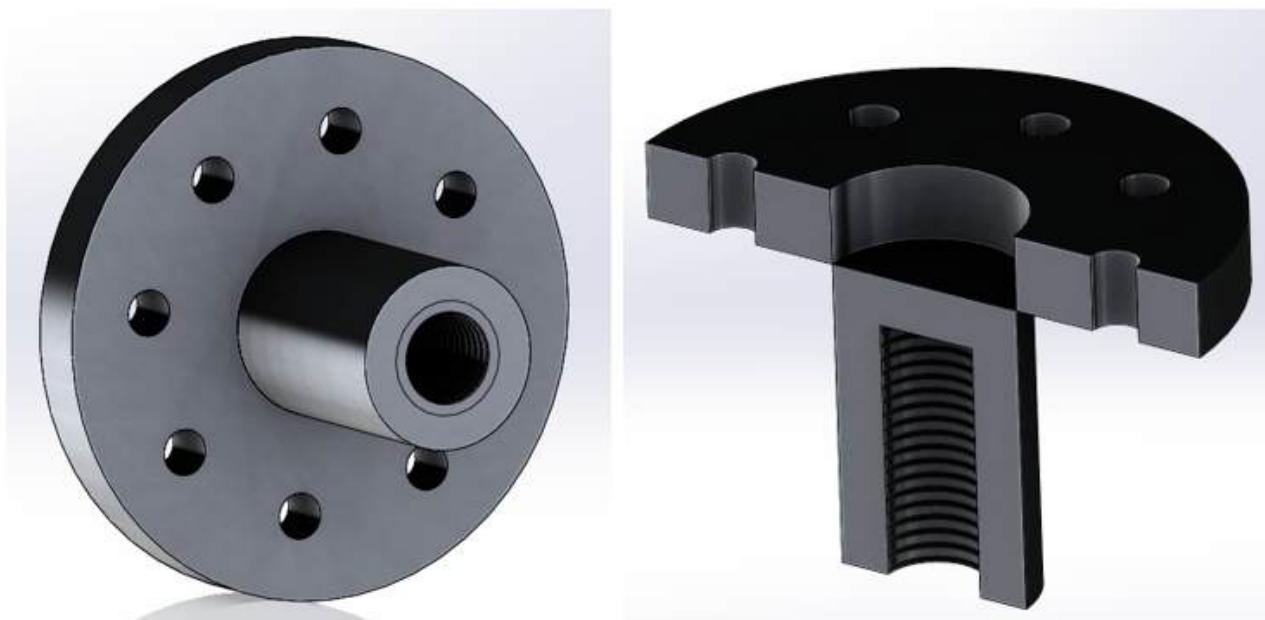


Figure 60: Adapter design iteration 1

As seen in the figure above, an adapter to the drive wheel was required for the small gear. Be-

cause the gear is too tiny for screw holes that would fit LSS-HS1 drive wheel profile. Initially the 3D printed gear was thought to be mounted directly on the adapter cylinder by screwing it tight with an M4x0,7 coarse bolt. This could potentially initiate crack formation in the material. In addition, the radial rotation after being fastened would only be held by the bolts preload and friction between adapter and gear. This is not ideal, bringing about a new design.

*Adapter design iteration 2:*

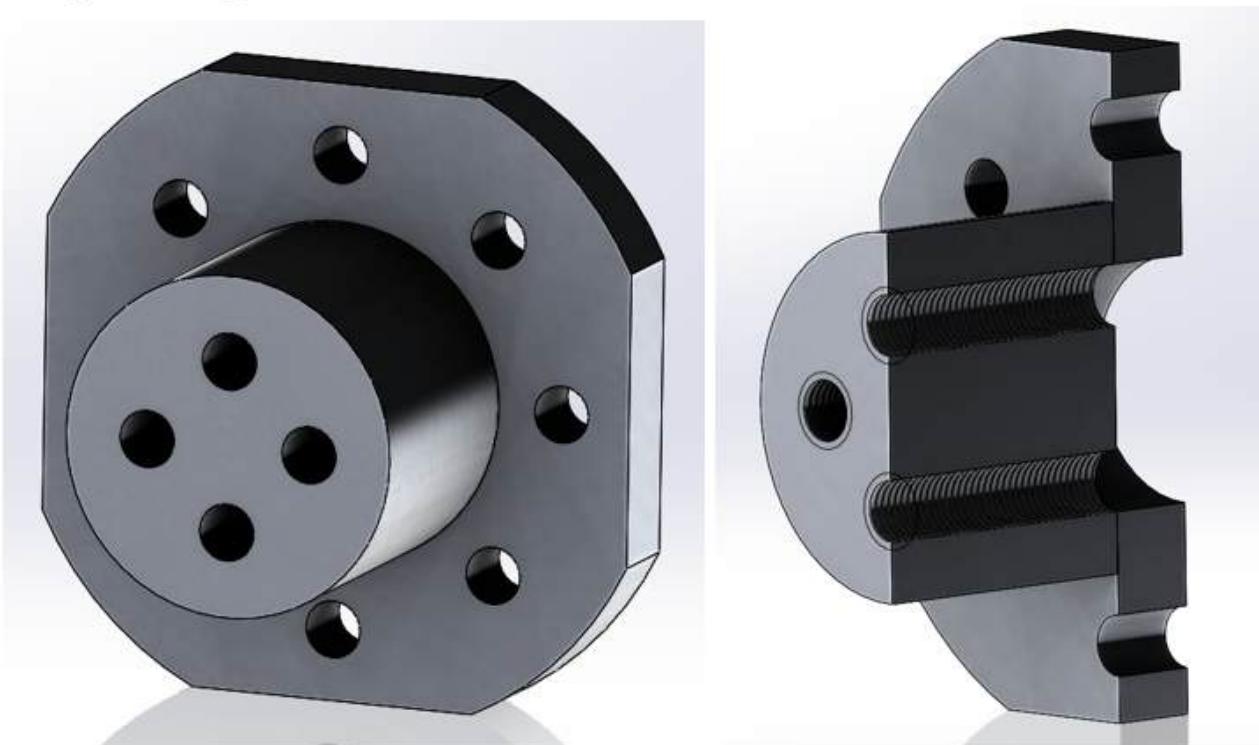


Figure 61: Adapter design iteration 2

This design is rooted in the first iteration with modified cylinder size to maximum diameter. This could not be larger due to the 8 x M2x0,4 coarse machine bolts being used to fasten the adapter. The four mounting holes intended to be threaded preventing the gear from unscrewing itself by utilizing 4 x M2x0,4 coarse machine bolts. But the gear had to be designed as small as possible, rendering this alternative impossible. Because the circular profile for the four holes was too close to the gear pitch diameter. There was also an attempt to decreasing the number of holes from four to three and still turned out to not work initiating another iteration.

*Adapter design iteration 3:*

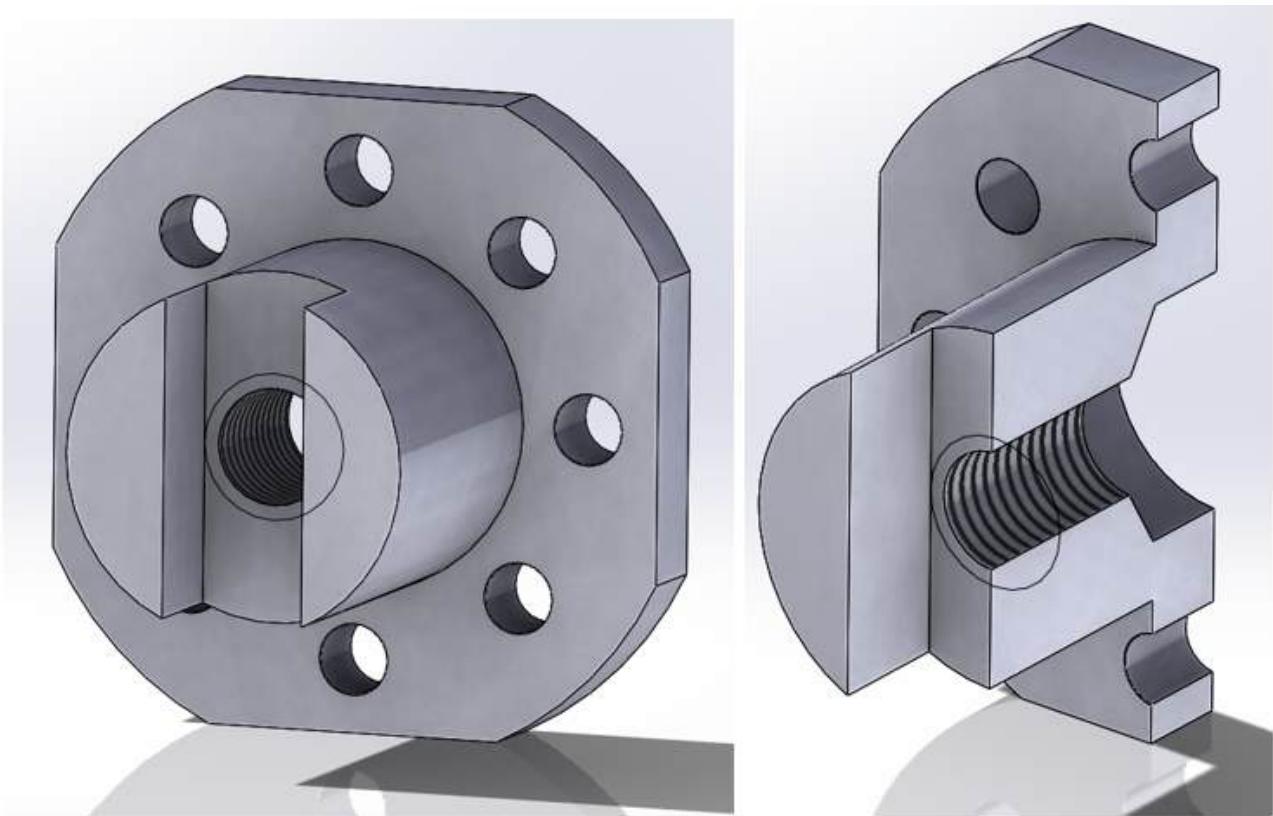


Figure 62: Adapter design iteration 3

Last iteration is built upon the first with a bolt secured through the adapter retaining the gear from axial movement. In addition, a milled path of the positive design on the gear. Preventing the linked gear from unscrewing risk and radial rotation. In hindsight, a machine key should have been milled out axially on the cylindrical tap.

*Exploded view and Bill of Materials:*

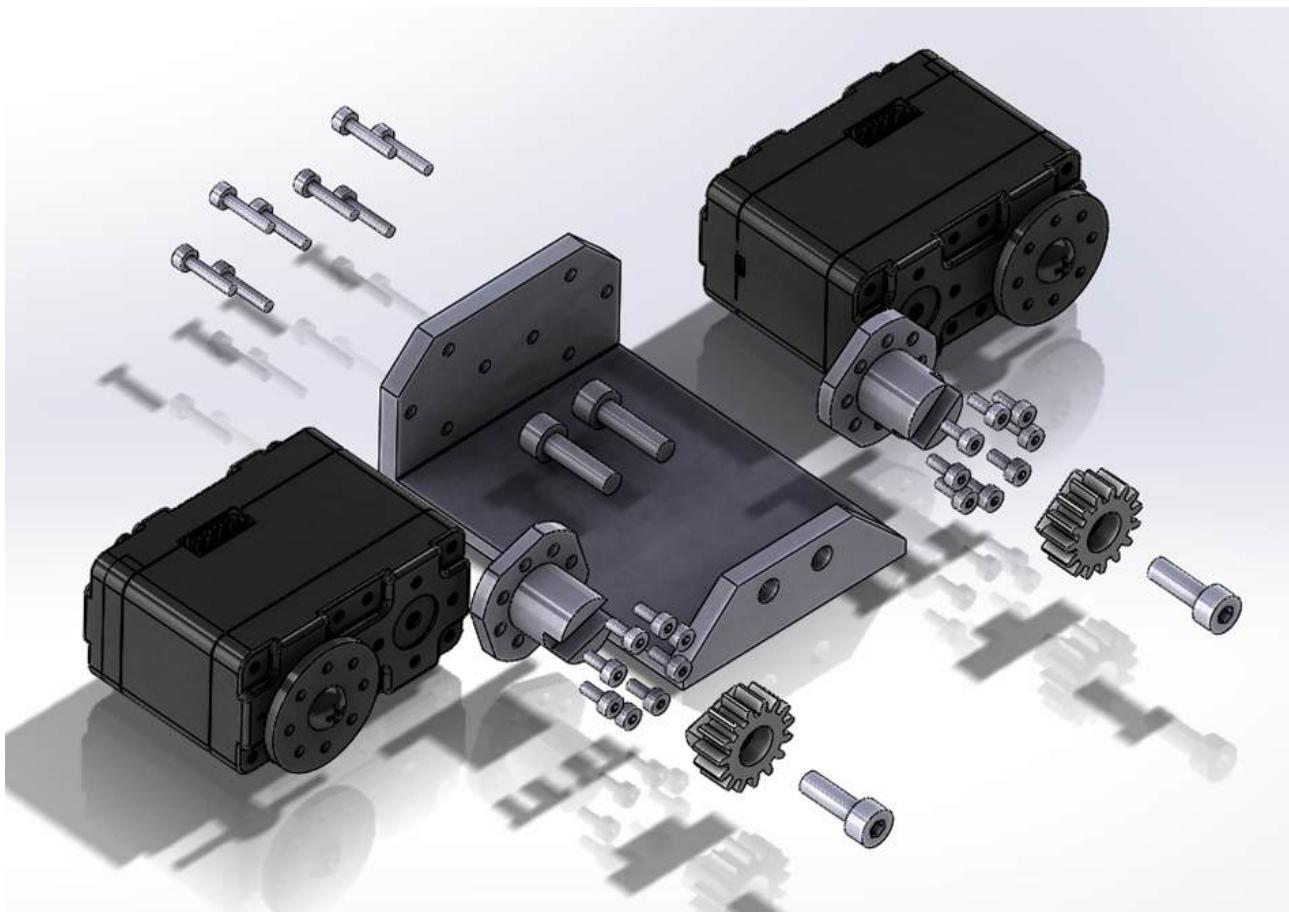


Figure 63: Fastening-bracket exploded view

Shown above is an exploded view of the sub-assembly with all its components.

Table 13: Fastening-bracket Bill of Materials

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Part name	Article no.	Size	QTY
1	BraketMotor	130009	6063-T6	45.81	Yes					1
2	AdapterDrivhjul	100000	6063-T6	5.41	Yes					2
3	GirP15T	100001	ABS	1.01	No	DuoArm	N/A			2
4	Motor HSS1	300000	N/A	46.81	No	Robotshop	Lynxmotion SES-V2 High Speed Smart Servo (LSS-HSS1)	RB-Lyn-986	HSS1	2
5	M2x0,4 L5	300001	Steel	0.04	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M2-5	M2x0,4, L5	16
6	M2x0,4 L10	300002	Steel	0.05	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M2-10	M2x0,4 L10	8
7	M4x0,7 L14	300004	Steel	0.31	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M4-14	M4x0,7 L14	2
8	M4x0,7 L12	300006	Steel	0.26	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M4-12	M4x0,7 L12	2

The bill of materials is showcasing each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

*Parts ordered:*

*Bolts:*

These are ordered from MiSUMi. Further details can be found in Bill of Materials. [50]

*LSS-HSS1:*

These are ordered form Roboshop. Further details are found in Bill of Materials. [?]

*Parts produced:*

*Bracket:*

This part could be produced utilizing a variety of methods, only two is mentioned. Firstly, the contour could be cut with a water-jet and then the holes made using a drill. The holes are not blind, nor threaded and should be machined in a three or five axis CNC machine. Secondly, the whole contour, including holes can be cut in a water-jet and then bent upright utilizing a

sheet bender. The latter option is more cost-effective and feasible, but due to limited time left and knowing that TE had to send the water cut part to their department in Moss for bending. It was not a viable solution in this case, and the part ought to be machined by a five-axis mill.

### *Adapter drivhjul:*

Producing the adapter could begin with a Ø30mm cylindrical aluminum shaft cut about 80mm long, assuring enough grip in the lathe spindle. Then turned the outer contour down to its required measures. Furthermore, fastened in a CNC mill and milled the flat edges for improved mounting option for drilling the M4x0,7 holes. At the same locked position, sticking off the path utilizing a stick steel tool. Then the part would be flipped around and mounted using the flat edges to drill the M4 center hole. Additionally in the same mount, drilling the Ø8mm spacing slot for LSS-HS1 drive wheel center screw. Lastly, tapping the required 0,7 pitch thread through the part. [86]

### *Gear:*

Produced utilizing a 3D printer able to print with different types of polymers. A step file is sent to the printer, printing parameters are manually defined and then the printer will do the rest. The finished part is then examined to verify an acceptable part.

### ***Axle sub-assembly:***

#### *Introduction:*

This sub-assembly consists of five unique components. Firstly, the retaining rings are mounted on the produced axle to keep it axially locked. Additionally, a M4 bolt screwed into the end, holding the gear in place. Initially the idea was to create a spline shaft to transfer radial force to the arms, but later the axle was modified by utilizing a keyway instead.

#### *Design and development:*

This chapter is about the design throughout the development process.

#### *Axle design iteration 1:*



Figure 64: Axle design iteration 1

Initially the axle was designed with splines that would cohere with the negative identity on the arm adapter. This solution would assure effective transmission of radial force between the axle and adapter. Additionally, the arms would be locked in place using retaining ring grooves turned out during production. The axle length was yet to be determined since all dependent variables had to be resolved. Furthermore, the diameter on both ends could at max be 10mm, due to adapter integration seen in the next iteration.

*Axle design iteration 2:*

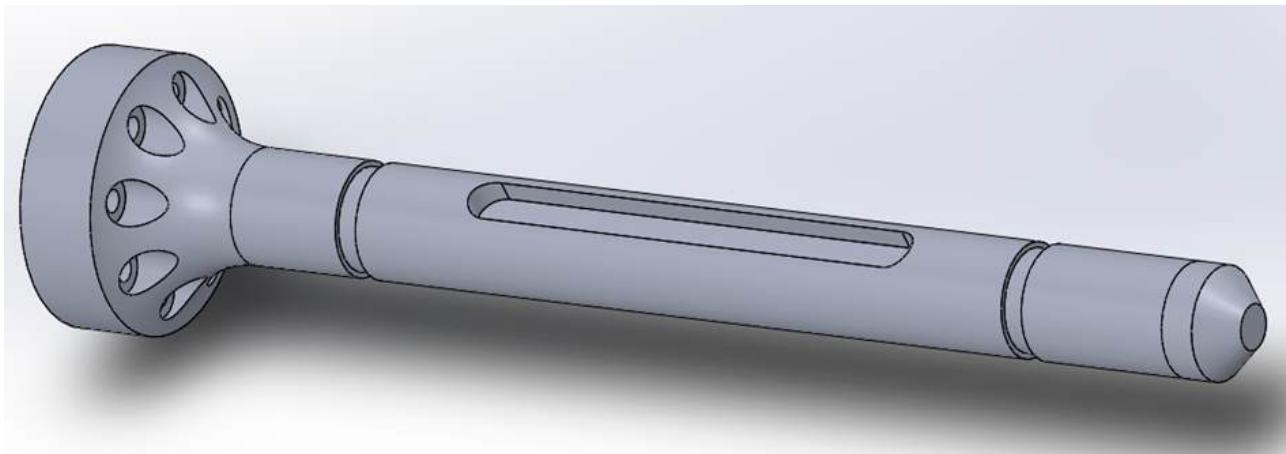


Figure 65: Axle design iteration 2

The outside adapter diameter to left is 24mm, the same as LSS-HS1s drive wheel and is directly connect to the motor. Mounted in place by 8 x M2x0,4 coarse machine bolts. At axle center, the splines were removed and a keyway for machine key was implemented. This is a simpler, time-saving and a more cost-effective solution. It is easier to produce a keyway than splines due to the time demanding and costly machine operation. Furthermore, a chamfer on the right side was added at 45 degrees. Making it easier to press fit the rightmost end into a ball bearing

shown on the model as a split line. Initially, one ball bearing having a set screw was thought to be used. The press fitted bearing would then support the motor mount in retaining axially movement. This was later dismissed due to the extreme price difference of plain bearings and set screw bearings at MiSUMi. Additionally, bearings made from polymer was investigated, but no suitable solution was found to meet the requirements. [89]

*Axle design iteration 3:*



Figure 66: Axle design iteration 3

In this iteration, the fillet on the adapter was removed to reduce machine time and improve cost-efficiency. The fillet itself could easily be machined by using a lathe. The decrease is gained from the sunken hole for bolt heads. These required to be machined out using a pin mill from a disadvantageous direction. The keyway must be milled out, but this requires another fixture within the CNC machine. Additionally, flat sides added to the adapter for versatile fixtures. Moreover, the adapter solution was modified to accommodate gearing necessity and increase output torque received from LSS-HS1.

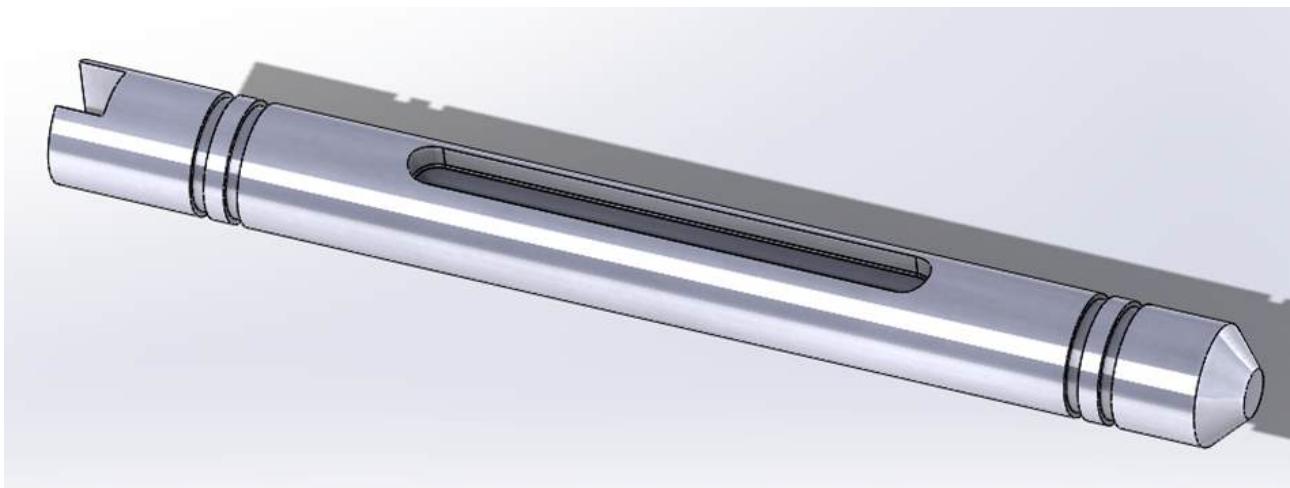


Figure 67: Axle design iteration 4

Resulting in the last iteration which includes the addition of two new grooves, securing the axle longitudinally. These are placed right next to the bearings. At this stage, all dependent parameters were determined and made it possible to place everything accurately. In hindsight, the axle diameter could have been greater since it no longer was dependent on the engine drive wheel adapter. Then the turning operation would easier, since Al 6063-T6 is a soft material and could flex during production. Resulting in difficulties for the machine operator regarding sufficient surface tolerance along the work-piece.

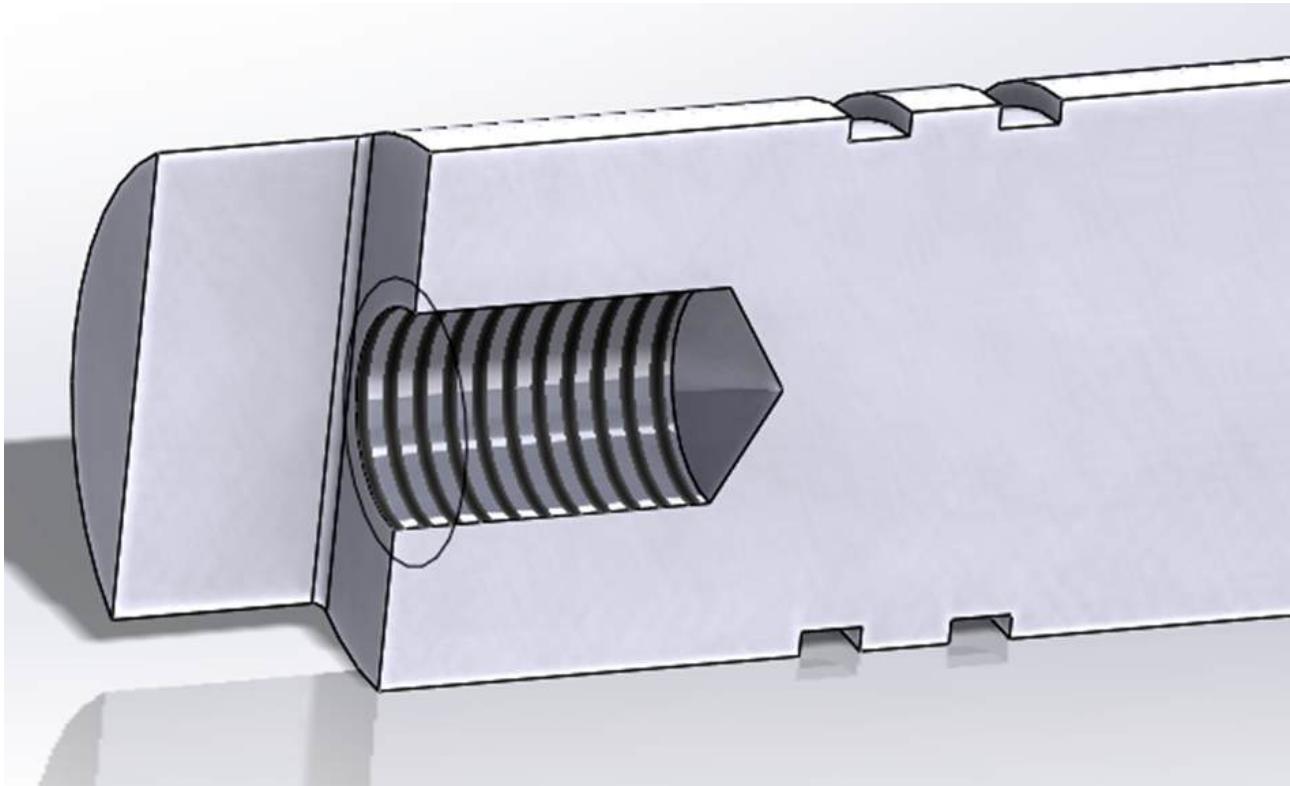


Figure 68: Axle design iteration 4.1

Moreover, the new solution for transmitting rotational force using gears was implemented to the axle. The principal utilized was mentioned above in the fastening bracket sub-assembly under adapter. It only has one additional factor considered. The threading process, which in this case is blind because the hole does not go all the way through the work-piece. Thread rods does not usually create threads at the end of the rod, which means that the predrilled hole must be deeper than the threads required by the bolt. Additionally, common practice is to add the diameter of the thread rod including one or two millimeters to the drilled hole guaranteeing that the required thread depth is met. As mentioned in the chapter above, a machine key could have been the better option securing the gear from rotational movement. It would spare the machine operator for one fixture during milling since the same fixture for both keyways could be used.

*Machine key:*

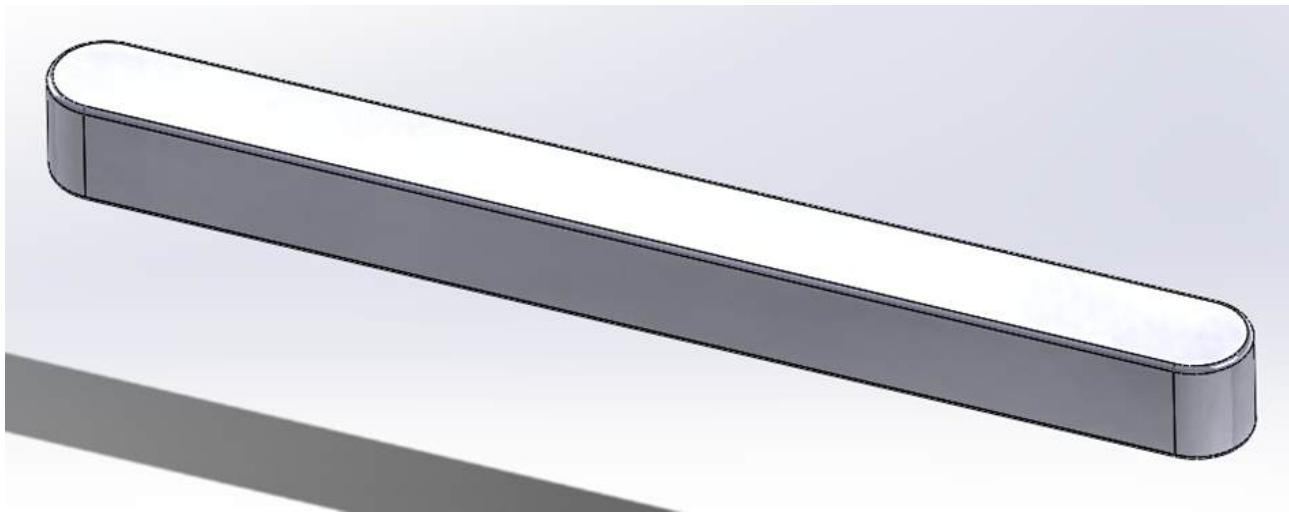


Figure 69: Machine key

The machine key is put into the keyway machined into the axle, guaranteeing transfer of radial force. Design parameters were determined by utilizing the metric standards for machine keys. [89] Apart from the length that was determined by the arm adapter. Additionally, axle keyway also had to be designed by utilizing these standards to assure perfect fit with required tolerances. In hindsight should machine key length been reduced from 44mm to >40mm and bought from MiSUMi. This is more cost-effective since the part is small and more expensive to produce. It could also be ordered in larger quantities at a discount like most parts form MiSUMi.

*Exploded view and Bill of Materials:*

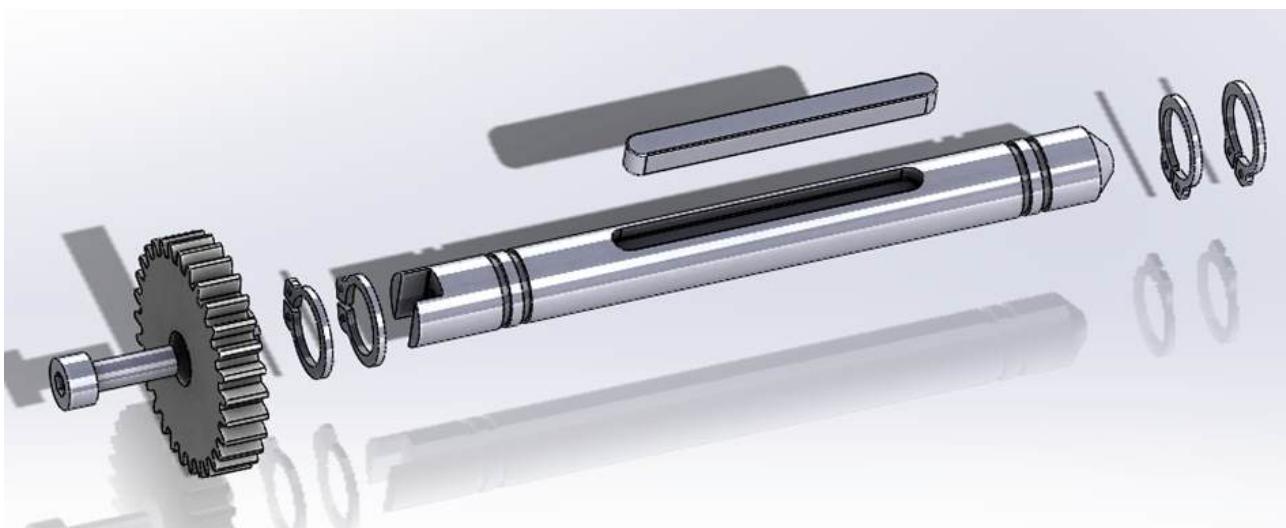


Figure 70: Axle sub-assembly exploded view

Shown above is an exploded view of the sub-assembly with all its components.

Table 14: Axle sub-assembly Bill of Materials

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Part name	Article no.	Size	QTY
1	KileL44	130003	6063-T6	1.86	Yes					1
2	Aksel	130001	6063-T6	18.90	Yes					1
3	GirP30T	100002	ABS	4.16	No	DuoArm				1
4	Retainingring10mm	330001	Steel	0.06	No	Misumi	Retaining ring/ External / E-Type	LSRE-64TIGSC-NO.9	Shaft, OD = 18mm, ID = 9,1mm	4
5	M4x0,7 L12	300006	Steel	0.26	No	Misumi	Hexagonal Socket Head Bolt	CSH-ST-M4-12	M4x0,7 L12	1

The bill of materials is showcasing each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

*Parts ordered:*

*Retaining ring:*

These are ordered from MiSUMi. Further details can be found in Bill of Materials. [87]

*Bolts:*

These are ordered from MiSUMi. Further details are found in Bill of Materials. [50]

*Parts produced:*

*Gear:*

Produced utilizing a 3D printer able to print with different types of polymers. A step file is sent to the printer, printing parameters are manually defined and then the printer will do the rest. The finished part is then examined to verify an acceptable part.

*Machine key:*

This part should have been ordered instead of machined to save production costs and time. To produce it, an oblong rectangle is cut out of aluminum then mounted in a multi axial CNC machine, setting the zero point. Then milling the radial key end with an end mill tool in addition to the fillet for the available top surface. Further, the key is flipped, mounted, zero-point set, and the other side is milled with same procedures.

*Axle:*

The part is supposed to be Ø10mm with a H7 surface tolerance that allows for rotation. In hindsight, should this be adjusted to a press fitted tolerance. The part ought to be press fitted into the ball bearings on each side, even though it is easier to dismantle the assembly utilizing H7. The tolerance could be problematic producing using a lathe since the shaft is made from a material considered as soft. The better option would be to order a shaft that already has been grinded to the required diameter. In this case, it is sensible to cut the premade shaft with a few extra mm in length and cut off the end attaining a good surface. Then cut out grooves for retaining rings on this side by utilizing a stick steel tool. Furthermore, flipping the shaft and repeating the same procedure. Then drilling the end hole with a Ø3,5mm drill and after that tapping the hole with M4 coarse threads at a pitch of 0,7. Subsequently clamping the shaft in a five axial CNC mill lengthwise, produce the keyway utilizing an end mill tool for keyways. Then rotate the component 90 degrees, milling the last pathway at the tapped hole end.

### 6.1.8 Arms

AK | SG

## ***Arm assemblies***

### ***Introducion:***

The arm assembly is a crucial part of the Duopod design. The arm configuration has a high impact on the stability, reach, dimensions of workspace, as well as the end effector. Regarding to the arm lengths, these are fixed from the Junior design-space process, as this fulfilled the required workspace. The active arms are powered by a motorized shaft utilizing two Lynx motion HS-1 smart servos. Passive arms receive no power but are manipulated by the active arms and the shared end effector.

### ***Interfaces:***

#### *The axle sub assembly:*

The axle rests inside the IGUS bearings plugged into the arms and adapter. Radial constraint is provided by a keyway and machine key, which are used to fix axle rotation to other elements. In addition, the axial movement is fixed by placing retaining rings on both sides of the arm assembly.

#### *The stabilizer assembly:*

To merge the stabilizer assembly to the arm there is used a slightly longer hinge pin at the right arm elbow. The mount for the stabilizer is spaced out using the IGUS polymer bearings. Hence, the stabilizer is mounted on the elbow pin and locked in place utilizing a retaining ring.

#### *The tool hub assembly:*

This interface uses the same principles for separation of carbon fiber and aluminum as explained in Connections below, and are further elucidated in the tool hub chapter.

### ***Connections:***

The assembly is composed of two active arms, connected by an axle adapter. The carbon fiber arms are separated from aluminum parts using the IGUS high glide polymer spacers. In the axle end of active arm, there are found two different collared sleeve bearing variations. These bearings separate both the hinge pin and the axle adapter from direct contact with the carbon fiber. In the Axle hole there are four 14mm diameter bearings. One inserted into the arm and the second into the adapter, at both sides of adapter. At the 6mm adapter hinge pin, there are four 6mm diameter bearings assembled in the same way. At the elbow we have two

different hinge pin configurations in regards to the right or left arm. The right arm (Frontal view), utilizes the same method for spacing with the 6mm collared sleeve bearings. Instead of an adapter there are two extra grooves with retaining rings, securing the arms axially. The two passive arms are then mounted on an elbow pin with the carbon fiber bracing in between, locked in place by the inner retaining rings. The left arm (Frontal view), features a longer pin working as an interface with the stabilizer assembly. In addition to the bearings from right arm, there are three additional bearings. One sleeve bearing for spacing, one thruster bearing for axially locking the stabilizer, and one collared sleeve bearing which penetrates the stabilizer triangle.

### ***Design and development***

#### *Active arm iteration 1:*



Figure 71: Active arm iteration 1

The original arms from the Junior model was first integrated into the senior model, as they already had the correct dimensions, and a new interface with axle was undetermined as of now. Therefore, we had the old axle interface, but the holes for adjustment was removed during this iteration.

#### *Active arm iteration 2:*



Figure 72: Active arm iteration 2

Iteration two was formed by the new axle interface with the initial spline coupling and aluminum adapter. Hence there are found two new holes at the axle end. One at Ø12mm for the

IGUS bearing holding the axle, and a smaller at Ø6mm for an additional fixture with a 6mm pin with an equivalent bearing as a interface spacer.

*Active arm iteration 3:*



Figure 73: Active arm iteration 3

From iteration two there had been some changes to the axle design, changing the spline coupling with a machine key design. Fitting the machine key, an expansion of the axle hole from Ø12mm to Ø16mm was necessary to implement a larger IGUS bearing in which increased the hole dimension and made axle insertion possible. Additionally, implying a modification of the utilized retaining ring to a greater outer diameter. Furthermore, a small slit centered on the part was added for aesthetics and additional mass reduction.

*Passive arm iteration 1:*



Figure 74: Passive arm iteration 1

Similarly to the active arm, the passive had fixed dimensions from Junior and was directly transferred to the senior model. Further, the modular holes for adjustments including the skeletal design was momentarily suppressed.

*Passive arm iteration 2:*

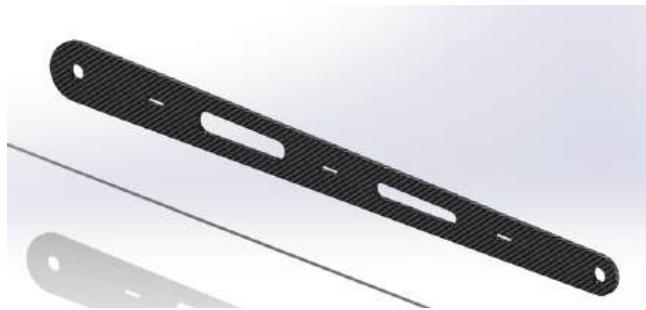


Figure 75: Passive arm iteration 2

Lastly, this iteration contained a variety of modifications. This includes a modification to the arm width. As a result, the original cylindrical spacers were switched for a more optimal carbon fiber bracing. The carbon fiber bracing is placed into the new rectangular slots and glued in place utilizing suitable epoxy for the given matrix. Furthermore, there was added skeletal slits intended to improve design aesthetics.

*Bracing:*



Figure 76: Bracing

The bracing design was ultimately done freehanded, since the forces in Y-Axis is almost negligible and there were limited restrictions to address. In addition, the bracing only aids structural integrity and rigidity. The mounting slots was predetermined in the arm design; therefore, the design is modeled after these points of interface with passive arm. Due to the water-jet production process and the high stiffness of carbon fiber this design could be designed mostly for esthetics. Since CFRP stiffness is high, the design may be modeled freely focusing on the esthetics. Because of the waterjet production method, there are few constraints when modeling due to the process which. Subsequently, there was made multiple attempts at creating this bracing with a topology study in the SolidWorks simulations feature. They are not featured in

the report given the poor results, but the essence of the simulation can be summed as follows: Using a solid piece of carbon fiber fixed at realistic points and using proportional forces to simulate inner stress in the material. Global goals for the relation between mass and stiffness are predetermined and the program removes mass where stress levels are low. *Axle Adapter iteration 1:*



Figure 77: Axle adapter iteration 1

First iteration of axle adapter featured two holes. One at Ø12mm for the axle interface which includes a spline coupling for the rotational fixture. Additionally, a hole at Ø8mm for the Ø6mm hinge pin to support the adapters rotational ability and preserves the arm rotation. Diameter dimensions is determined by the IGUS bearings inserted into the adapter. This initial design also featured rounded edges to minimize the mass and additionally improving the esthetic design.

*Axle Adapter iteration 2:*

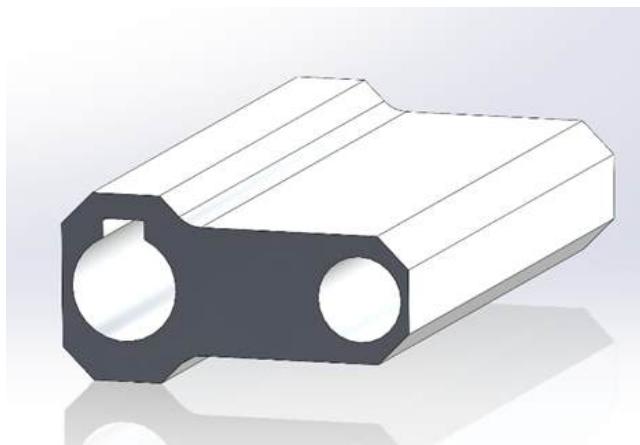


Figure 78: Axle adapter iteration 2

In iteration two there was a modification with axle interface, transitioning from a spline coupling to a machine key. The new axle hole with suitable a keyway was implemented into the design. Ref [89] Furthermore, feedback on the second design review highlighted complications regarding

the machine process of the adapters surface contour, shown in iteration 1. This generated a design modification where a rectangular design was implemented with additional 3mm chamfers to the outer corners. Furthermore, additional mass was reduced due to the complications that arose regarding LSS-HS1s lifting capacity. This caused the adapters T-shape appearance. *Axle Adapter iteration 3:*

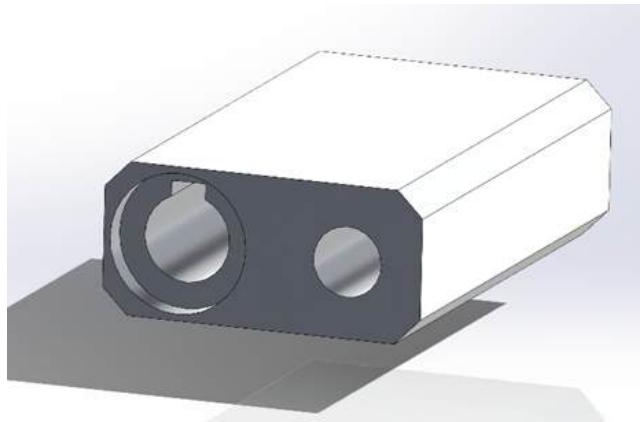


Figure 79: Axle adapter iteration 3

Hence, this iteration was simplified to one rectangular shape and chamfered at outer corners. Another challenge came to light when reviewing the total assembly. The machine key would not fit through the current 12mm inner diameter bearing. However, this was corrected by switching to a 14mm bearing, giving the machine key enough space to enter. As a result, a matching indent for the mentioned bearing was integrated at the hole entrance

***Exploded view and bill of materials:***

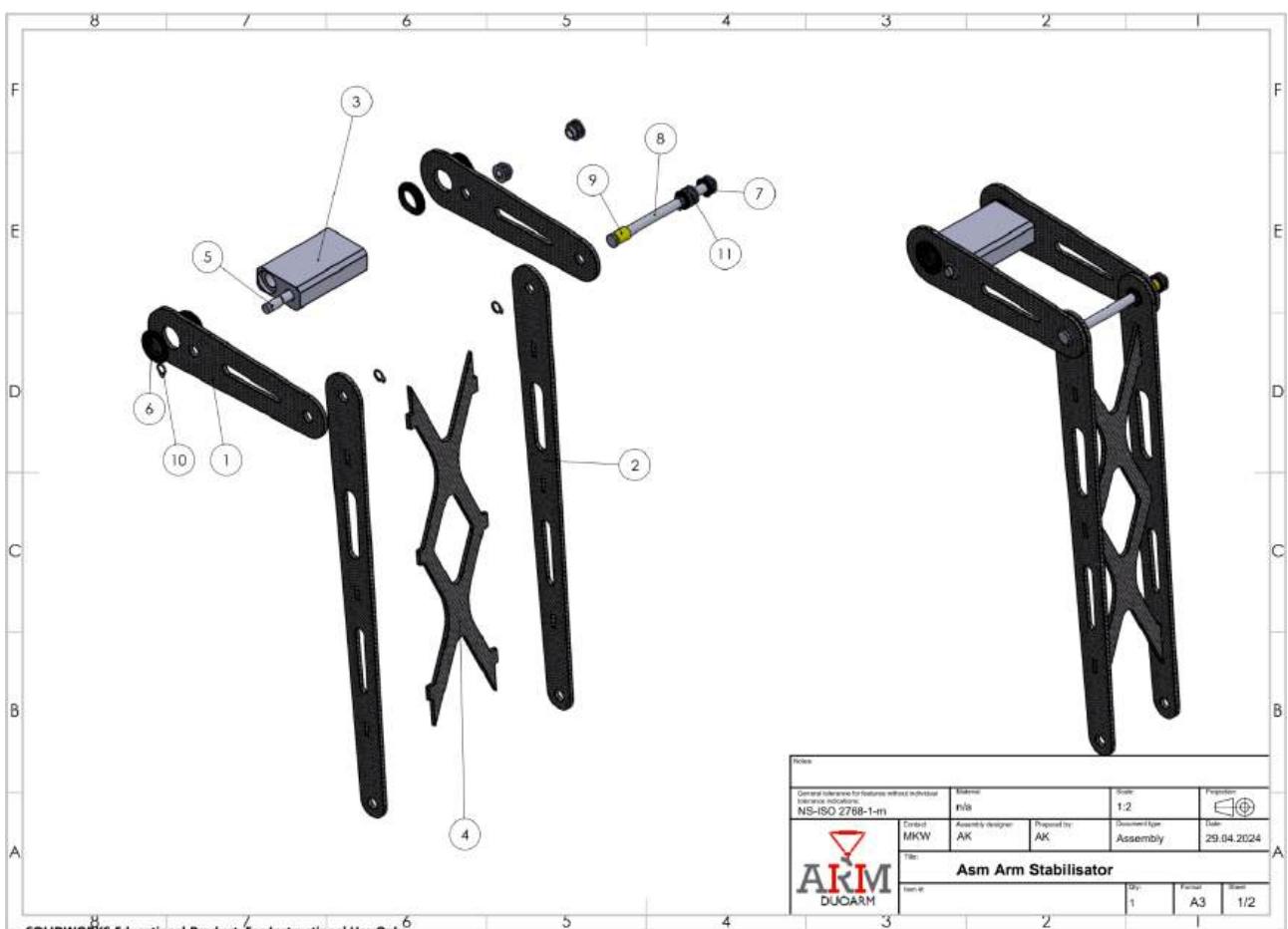


Figure 80: Stabilizer arm assembly

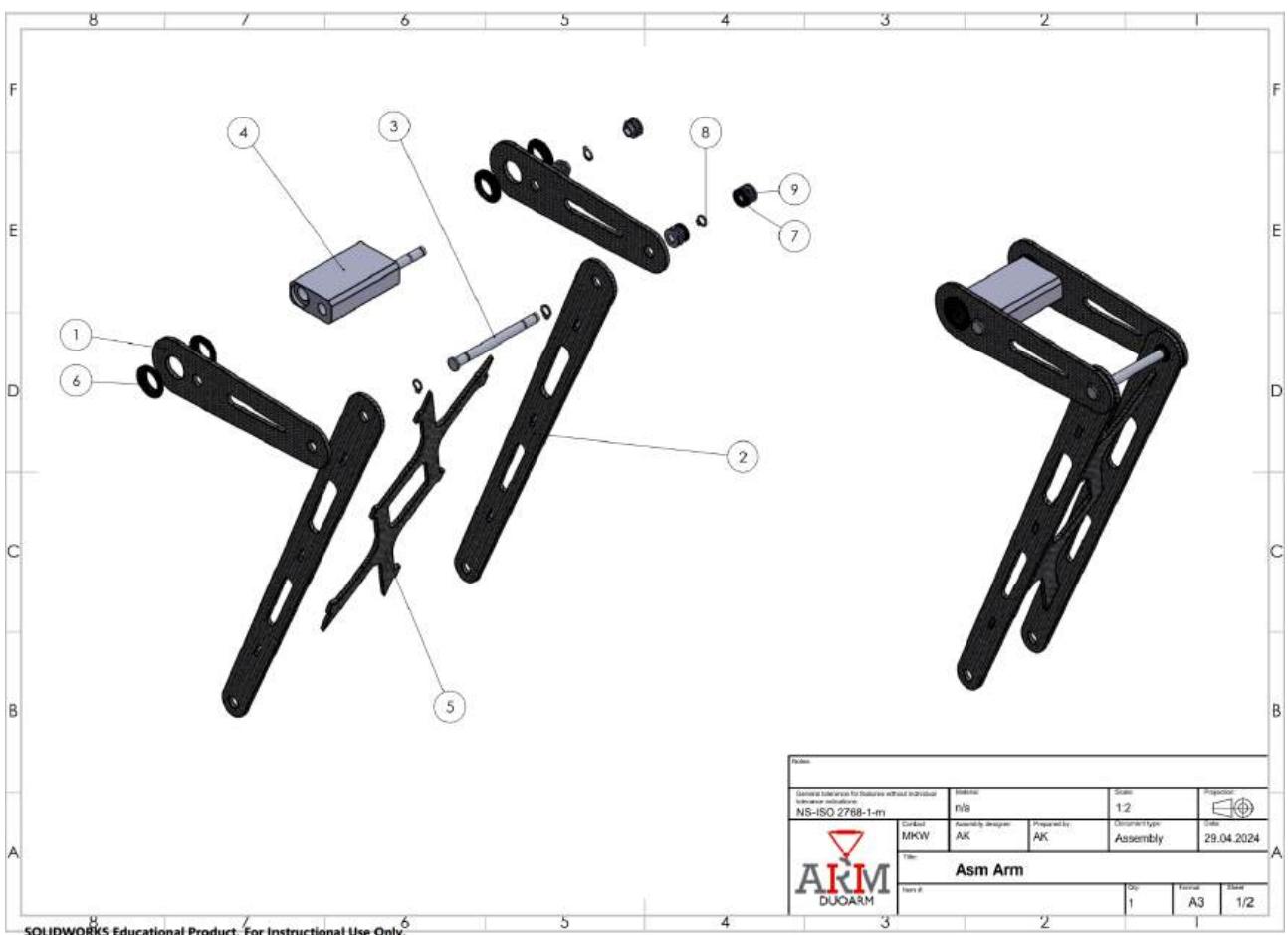


Figure 81: Arm assembly

Shown above is an exploded view of the assembly with all its components. These are mated together as it is intended to be assembled. It has six unique parts for production at TE and five unique parts being ordered from either MiSUMi or IGUS.

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Part name	Article no.	Size	QT
1	Aktiv Arm	140001	CFRP	27.64	Yes	Easy Composites	High Strength Carbon Fibre Sheet	CFS-RI-3-0225	500x470	2
2	Passiv Arm	140002	CFRP	41.76	Yes	Easy Composites	High Strength Carbon Fibre Sheet	CFS-RI-3-0225	500x470	2
3	Aksel-Arm Kobling	140003	6063-T6	79.66	Yes					1
4	Avstiving	140004	CFRP	30.37	Yes	Easy Composites	High Strength Carbon Fibre Sheet	CFS-RI-3-0225	500x470	1
5	HingePin AksellKobling	140006	6063-T6	5.80	Yes				6x73.25mm	1
6	FlangeBearing 14x4mm	340007	n/a	0.36	No	IGUS	iglide® G300, sleeve bearing with flange	GFM-1416-04	14x4mm	4
7	ThrusterBearing 6x1.5	300050	n/a	0.13	No	IGUS	iglide® G300, thrust washer	GTM-0612-015	6x1.5mm	3
8	HingePin ArmStab	140009	6063-T6	6.98	Yes				6x88.25mm	1
9	SleeveBearing 6x10	340010	n/a	0.22	No	IGUS	iglide® J sleeve bearing	JSM-0608-10	d <sub>1</sub> = 6mm, d <sub>2</sub> = 8mm, b <sub>1</sub> = 10	1
10	RetainingringØ6	300032	6063-T6	0.05	No	Misumi	Retainingring C-Type	STWN6	Nominal Ø6mm	4
11	Flange bearindD6	300022	n/a	0.15	No	lgus	Sleeve bearing with flange	JFM-0608-04	d <sub>1</sub> = 6mm, d <sub>2</sub> = 8mm, b <sub>1</sub> = 4mm	9

Figure 82: Bill of materials: stabilizer arm assembly

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Part name	Article no.	Size	QTY
1	Aktiv Arm	140001	CFRP	27.64	Yes	Easy Composites	High Strength Carbon Fibre Sheet	CFS-RI-3-0225	500x470	2
2	Passiv Arm	140002	CFRP	41.76	Yes	Easy Composites	High Strength Carbon Fibre Sheet	CFS-RI-3-0225	500x470	2
3	HingePin AkselKobling	140006	6063-T6	5.80	Yes				6x73.25mm	2
4	Aksel-Arm Kobling	140003	6063-T6	79.66	Yes					1
5	Avstiving	140004	CFRP	30.37	Yes	Easy Composites	High Strength Carbon Fibre Sheet	CFS-RI-3-0225	500x470	1
6	FlangeBearing 14x4mm	340007	n/a	0.36	No	IGUS	iglide® G300, sleeve bearing with flange	GFM-1416-04	14x4mm	4
7	ThrusterBearing 6x1.5	300050	n/a	0.13	No	IGUS	iglide® G300, thrust washer	GTM-0612-015	6x1.5mm	2
8	RetainingringØ6	300032	Spring steel	0.05	No	Misumi	Retainingring C-type	STWN6	Noninel Ø6mm	4
9	Flange bearingØ6	300022	n/a	0.15	No	IGUS	Sleeve bearing with flange	JFM-0608-04	$d1 = 6\text{mm}$ , $d2 = 8\text{mm}$ , $b1 = 4\text{mm}$	8

Figure 83: Bill of materials: Arm assembly

The bill of materials showcases each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

*Parts ordered:*

Retaining ring 6mm:

These are ordered from MiSUMi. Further details are found in Bill of Materials. [87]

IGUS bearings:

Flange bearing 14x4mm:

These are ordered from IGUS. Further details are found in Bill of Materials. [90]

Flange bearing D6 (6X3mm):

These are ordered from IGUS. Further details are found in Bill of Materials.

Sleeve bearing 6x10mm:

These are ordered from IGUS. Further details are found in Bill of Materials. [91]

Thruster bearing 6x1.5mm:

These are ordered from IGUS. Further details are found in Bill of Materials. [92]

*Parts produced:*

Composite parts:

These parts are a composition of carbon fiber and epoxy resin. Components included in the assembly consists of the bracing, active and passive arm. These are ordered in sheets at 500x470x3mm from Easy composites. Designated shapes are produced by utilizing the cutting force received from high pressure water in a water-jet. In order to produce these shapes, it is placed an extra outline from a relevant 2D view in the drawing delivered to TE for production. This contour is used when programming the water-jet cutting path. [93]

Adapter:

The adapter is produced using a CNC milling machine. Fastening at the long sides, half the depth of the adapters rectangular shape will be generated by an end mill tool. Before turning the workpiece and repeating process. Secondly, the holes will be predrilled to a smaller diameter and reamed to correct dimension with . The internal keyway is made using a broaching tool at the desired width.

Hinge pins:

The hinge pins are produced using a lathe an operation known as a turning. In this operation the work-piece is rotating while the tool is stationary mounted on the tool-holder. Tool is then fed lengthwise on along the work-piece at a predetermined depth and feed rate. Material is then removed at a rate known as MMR which is dependent on depth of cut, feed, and the rotational surface speed of the work-piece. The process is repeated until desired diameter is achieved. Furthermore, the groove for retaining ring is cut and the component is parted away from its cylindrical start piece at the hinge pin head. Additional, beginning cuts are often rough cuts, typically at high feed rates and larger depths followed by finish cuts at lower feed rates and depths to produce a good surface finish with acceptable tolerances.

### 6.1.9 Arm stabilizer

AK | SG

#### Stabilizer Assembly

##### ***Introduction:***

The stabilizer's task is to maintain the tools horizontal position over the conveyor belt and in stacker at the bottom position. This field of study is called parallel kinematics which is used to calculate the correct length for arms and stabilizing rods by employing the correct relations between the components. This includes the stabilizer triangle which commonly has to have the same triangular proportions as the connections at the fastening hub and tool hub. [51] While this is the case, our model is based primarily on what was suitable for our use during the Junior design process. Given that the stabilizer performed the desired movement, only minuscule changes has been implemented. Given more time, this would be one of the primary points for further research.[49]

##### ***Interfaces:***

*Fastening hub assembly:*

The short connector rod features a ball and socket joint at both ends. This joint is mounted on the hinge pin at the left side of the fastening hub front, and is secured by a retaining ring.

### *Arm assembly:*

As mentioned in previous chapter Arm assembly, sub chapter: interfaces. The elbow hinge pin is longer and spaced out on the right arm. Thus, giving the possibility to connect to the stabilizer triangle. For connection the triangle is positioned on a polymer bearing at the end of the pin, again secured with a retaining ring.

### *Tool hub assembly:*

Bearing much resemblance to the fastening hub interface. The ball joint from the long connecting rod is mounted on a hinge pin. Positioned at the top left of the tool hub, the pin works as a spacer as well as the point of attachment. A retaining ring is locking the bearing in place.

### ***Connections:***

The assembly features three main components: stabilizer triangle as well as long and short connecting rods, followed by the essential polymer bearings. The connecting rods are threaded on both ends with IGUS ball and socket joints on both ends. Two 3mm flange bearing are positioned towards one another in both of the Ø4.5mm holes. The top one being spaced out additionally with a 9mm sleeve bearing. Securing the rods to the triangle, two pins are inserted from the front and locked with retaining rings.

### **Design and development:**

#### *Stabilizer triangle iteration 1:*



Figure 84: Stabilizer triangle

Iteration one utilized the model from Junior development, but material is replaced with 6063-T6 Aluminum. The connector arm holes are also increased from Ø3mm to Ø4.5mm to fit IGUS bearing inside.

*Stabilizer triangle iteration 2:*



Figure 85: Stabilizer triangle iteration 2

Modifications done on this iteration were done purely to simplify the 2D-drawing process. Changes addressed the inner design cut, modified to two equal diameters with connecting parallel lines. Therefore, the model is suitable for both milling processes as well as cutting with

waterjet.

*Stabilizer short and long arms iteration 1:*



Figure 86: Stabilizer short arm: iteration 1



Figure 87: Stabilizer long arm: iteration 1

Initially the stabilizer arms were made of CFRP ordered from Easy Composites. They are produced by cutting, utilizing a waterjet at TEs workshop, allowing for the same plate design as mentioned in the Junior section. The end holes shown in figures above would be directly attached to the hinge pins with a clearance assuring radial rotation. In addition, locked from axial movement using a retaining ring. They are fastened at the connection point mentioned above in one end and at the adjacent point to their respective hubs. Furthermore, the esthetic mass removal at center of the arms remained the same.

*Stabilizer short and long arms iteration 2:*



Figure 88: Stabilizer short arm: iteration 2



Figure 89: Stabilizer long arm: iteration 2

After hub and triangle modification, it was a necessity for additional adjustments. Therefore, other design alternatives were thoroughly investigated. This led to the first tapped rod design with diameter of 3mm.[86] A threaded rod ensured good adjustability as well as high affordability, but the thickness caused low stiffness. Mounting procedure changed from directly on hinge pin, to an IGUS ball and socket joint. These where attached to the ends by threads and

mounted with pins.

*Stabilizer short and long arms iteration 3:*

When challenges arose in regard to the high Senior mass, the focus changed to conservation of mass without compromising the rigidity. A variety of materials was examined with the best options being: Pultruded Carbon fiber rods and nylon rods. Transferring from stainless steel to general use carbon fiber and nylon the rods received  $\frac{1}{2}$  of the original weight. Difficulties included the threading of nylon and carbon fiber in regards to wear and general processing.

*Stabilizer short and long arms iteration 4:*



Figure 90: Stabilizer short arm: iteration 4



Figure 91: Stabilizer long arm: iteration 4

Iteration 1 provided the desired adjustability, but lacked the rigidity. Hence, the decision to modify a solid 5mm aluminum rod. Delivering both the adjustability of a threaded 3mm rod, while simultaneously providing the rigidity of a solid 5mm rod. The outcome is a polished aluminum which complements the high gloss carbon fiber, enhancing the esthetical contrast.

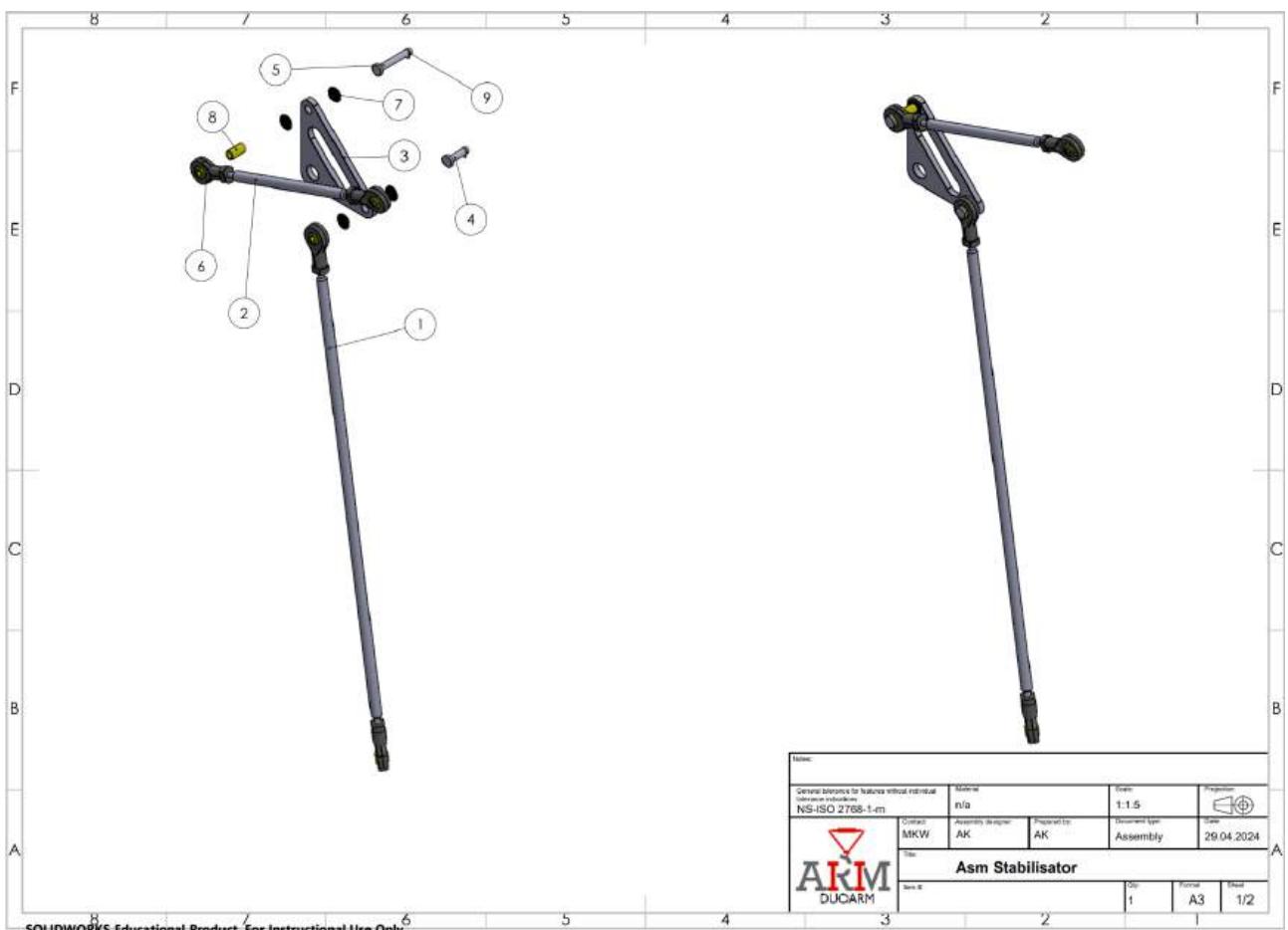


Figure 92: Stabilizer exploded view

### Exploded view and Bill of materials

Shown above is an exploded view of the assembly with all its components. These are mated together as it is intended to be assembled. It has six unique parts for production at TE and five unique parts being ordered from either MiSUMi or IGUS.

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Article name	Article no.	Size	QTY.
1	Stang 5x260mm	150001	6063-T6	13.00	Yes					1
2	Stang 5x102.5mm	150002	6063-T6	4.57	Yes					1
3	Stabilisator trekant	150008	6063-T6	8.12	Yes					1
4	HingePin stabilisator 12mm	150009	6063-T6	0.34	Yes				3x12mm	1
5	HingePin stabilisator 21mm	150010	6063-T6	0.51	Yes				3x21mm	1
6	Ball Joint 3mm	350003	N/A	1.05	No	IGUS	igubal® rod end bearing with female thread	KBLM-03	3mm	4
7	FlangeBearing 3x2	350004	N/A	0.03	No	IGUS	iglide® G300, sleeve bearing with flange	GFM-0304-02	d1 = 3mm, d2 = 4,5mm, d3 = 7,5mm, b1 = 2mm, b2 = 2,5mm	4
8	SleeveBearing 3x9	350006	N/A	0.08	No	IGUS	iglide® J, sleeve bearing	JSM-0304-09	d1 = 3mm, d2 = 4,5mm, b1 = 9mm	1
9	RetainingringØ3	300031	Spring steel	0.00	No	Misumi	Retainingring C-Type	STWN3	Nominal Ø3mm	2

Notes:  
General tolerance for features without individual  
dimensions: ±0.1 mm  
NS-ISO 2768-1-m

Material: n/a	Scale: 1:1.5	Projection:		
Contact: MKW	Assembly designer: AK	Prepared by: AK	Document type: Assembly	Date: 29.04.2024
Title: Asm Stabilisator				
Item #: 1	Dly: 1	Format: A3	Sheet: 2/2	

SOLIDWORKS Educational Product. For Instructional Use Only.

Figure 93: Stabilizer Bill of Materials

The bill of materials showcases each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

#### Parts ordered:

IGUS parts:

Ball Joint 3mm:

These are ordered from IGUS. Further details are found in Bill of Materials. [94]

Flange bearing 3x2mm:

These are ordered from IGUS. Further details are found in Bill of Materials. [95]

Sleeve bearing 3x9mm:

These are ordered from IGUS. Further details are found in Bill of Materials. [96]

Retaining ring 3mm:

These are ordered from MiSUMi. Further details are found in Bill of Materials. [87]

***Parts produced:***

Triangle:

The triangle contour and holes are envisioned for cutting by waterjet. Designated shapes are produced by utilizing the cutting force received from a high-pressure waterjet. In order to produce these shapes, it is placed an extra outline from a relevant 2D view in the drawing delivered to TE for production. This drawings contour is used when programming the machine cutting path. Secondly, the workpiece is mounted in a CNC mill and utilizes a reaming process for the H7 clearance tolerance.

Connection rods:

Rods composed of 6063 T6 Aluminum is cut to length, mounted in the chuck and machined down at both ends in a lathe. Secondly, these ends are threaded to 3mm coarse thread (3mm with 0.5mm pitch). Ref: iglidur, threads.

Hinge pins:

The hinge pins are produced using a lathe an operation known as a turning. In this operation the work-piece is rotating while the tool is stationary mounted on the tool-holder. Tool is then fed lengthwise on along the work-piece at a predetermined depth and feed rate. Material is then removed at a rate known as MMR which is dependent on depth of cut, feed, and the rotational surface speed of the work-piece. The process is repeated until desired diameter is achieved. Furthermore, the groove for retaining ring is cut and the component is parted away from its cylindrical start piece at the hinge pin head. Additional, beginning cuts are often rough cuts, typically at high feed rates and larger depths followed by finish cuts at lower feed rates and depths to produce a good surface finish with acceptable tolerances.

### 6.1.10 Tool hub assembly

SG | AK

#### Introduction

The tool-hub has an important role for the pick and place system. It is not only carrying

the fictional tool with a integrated quick release system. Additionally, connecting both arms and stabilizer. Linking the whole system together at the lowest position assuring that the mechanical movement of arms work together as intended. In addition, the stabilizer mounting point is critical which keeps the hub horizontal during a pick and place operation. Furthermore, the tool-hub is machined out of Aluminum at TE, Al 6063-T6 and the arms are made from CFRP. These materials cannot be in direct contact without creating a galvanic cell. A replaceable wear part must be implemented into the design, guaranteeing rotational movement while assuring no direct contact.

## Interface

The figure bellow shows an illustration of how it is intended to guarantee no direct contact between the CFRP passive arm and the aluminum tool hub. This will be achieved by using a washer with collar which has a cylindrical hole centered and utilizing a plain washer on the other side to hinder contact between the passive arm and fastening point on the hub. The initial idea was to employ a softer metal to split the materials from the creation of a galvanic cell, but even though using typical soft materials like copper or bronze creates a challenge. As a rule of thumb, the potential difference of materials in direct contact should not exceed a value of 0,25 volt. Furthermore, aluminum's value being -0,80-volt, untreated steel -0,70, copper -0,20 and bronze -0,25 from the voltage-series table. Resulting in a galvanic reaction where one metal acts as cathode and another anode. In which eventually ends in corrosion and drastically reduces the lifespan of the parts with increase possibility of fatigue failure.

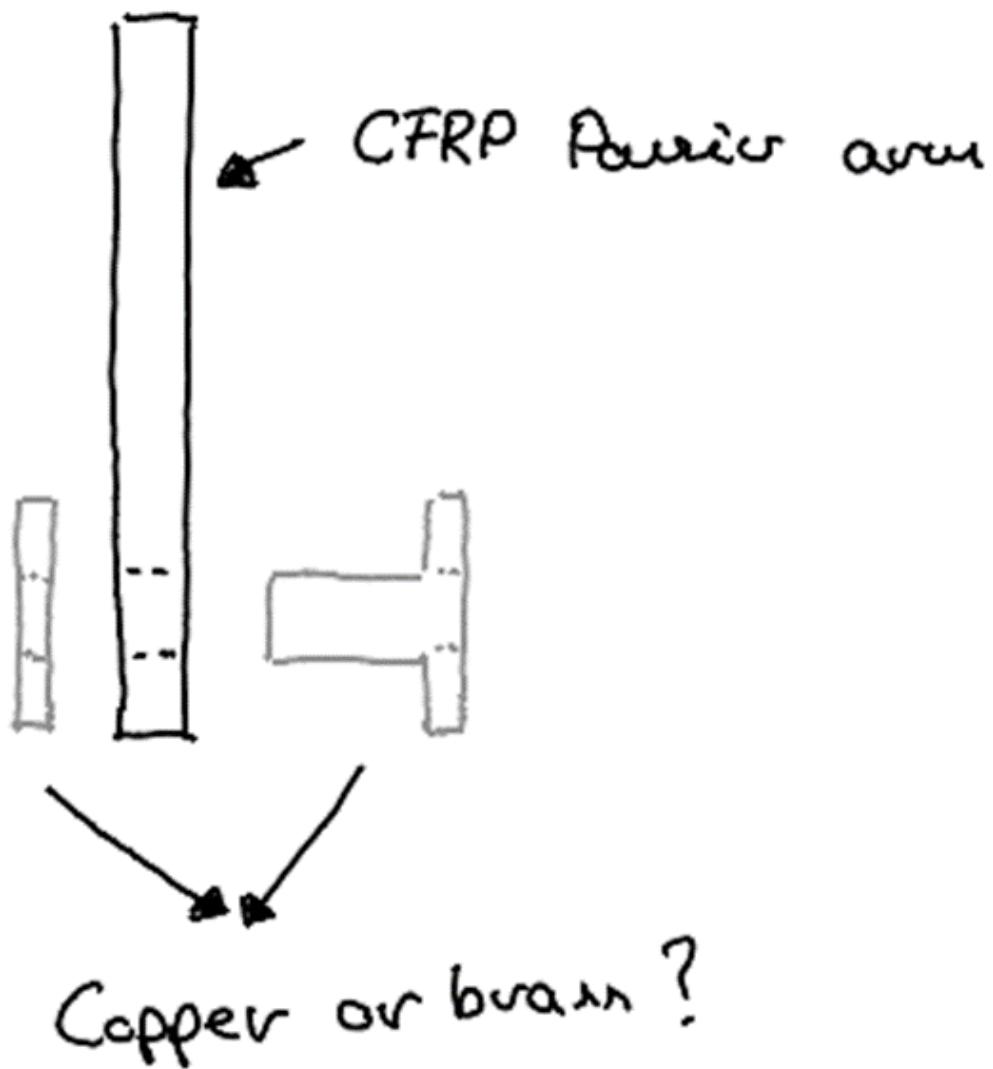


Figure 94: Interface connection

To solve these difficulties, a decision was made to change the material of replaceable parts to be polymer based. We found a web-page where shoulder washers made from nylon could be ordered. Nylon is a better option since it is less expensive, has less weight and is heat resistant which is ideal due to the potential heat generated from friction. Additionally, nylon is a durable and tough material that remains strong after exposure to different elements which makes it resistant to fading and more favorable than soft metals like copper and bronze. Furthermore, polymer acts as an insulator and works by being an insulating layer which does not react to metals and ensures no galvanic reaction between materials in direct contact. [97]

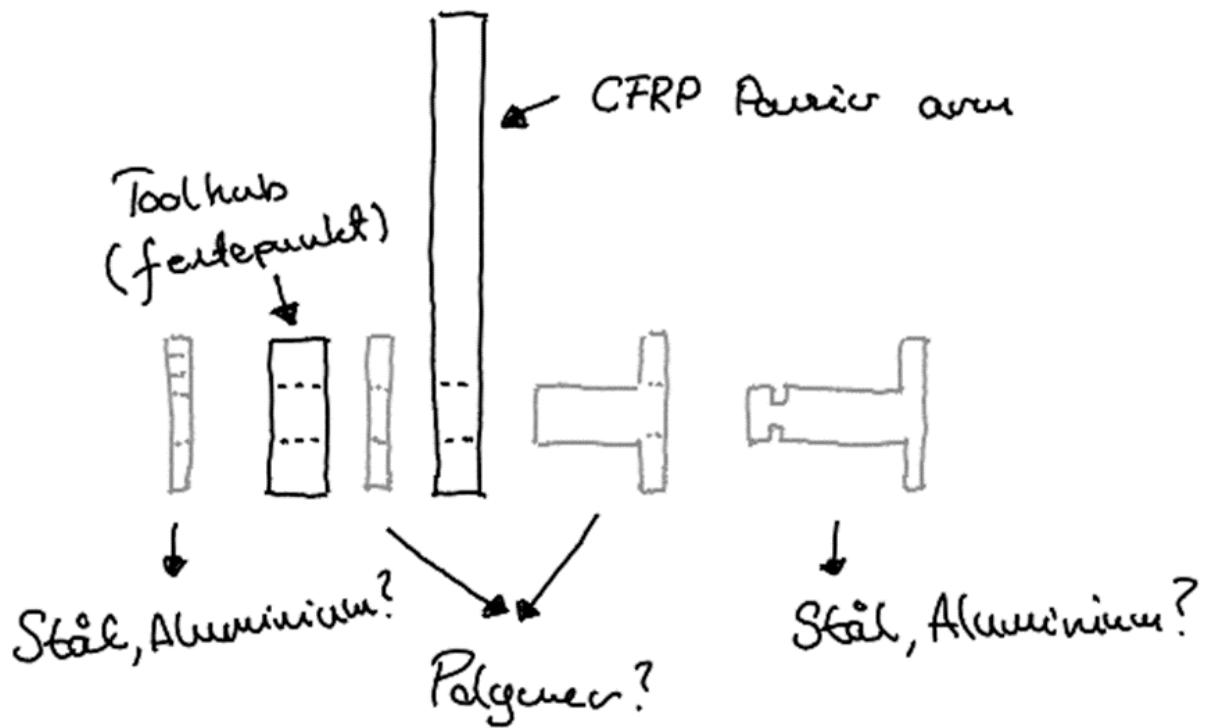


Figure 95: Tool hub - Arm interface

The figure above shows a draft of how the arm is thought to be mounted on the four corners of the hub. From the right side, a hinge pin carved with a groove at the end for the retaining ring is turned in a lathe from either steel or aluminum. This pin fits inside the collard washer made from polymer and fits snugly inside the CFRP passive arm. Furthermore, the pin is put into a washer and then through the hubs mounting hole and locked in place by using a retaining ring ordered from MiSUMi. Altogether, assuring radial rotation with low friction by using H7 tolerance on holes where involved parts is supposed to rotate. Additionally, avoiding a galvanic reaction and preventing axially movement.

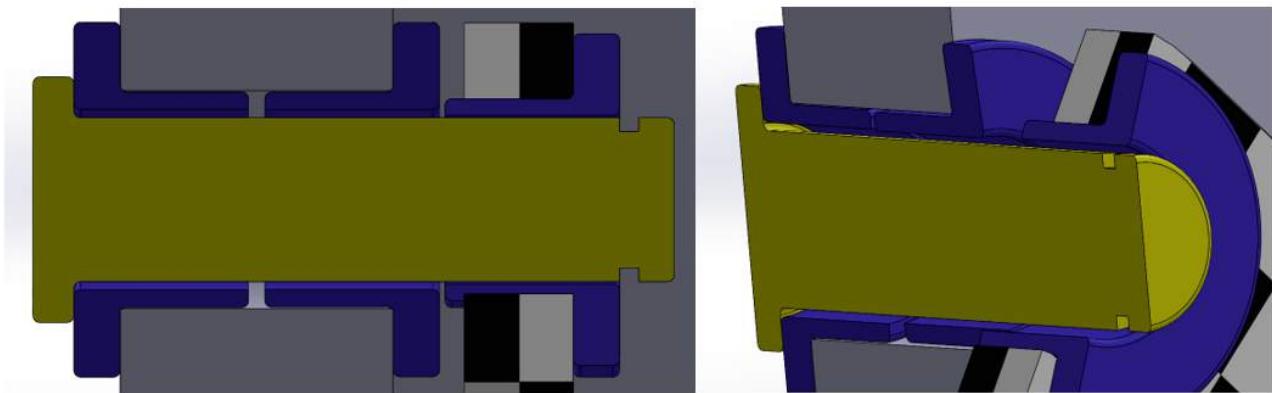


Figure 96: Section view - Connection

Instead of utilizing replaceable parts, a bearing solution was researched. The bearing could either be made from steel or polymer and implemented in the tool-hubs four mounting holes. But bearings would result in a larger tool-hub width which again accumulates to a chunky hub with unnecessary large mass that the LSS-HS1 motors must lift. The ball bearing idea was dismissed for the simpler solution by employing bushings made from a softer material than Al 6063-T6 and CFRP. Utilizing a bushing with direct contact provokes higher friction in the connection, but it was found that IGUS had bushings made from polymer that was highly resistant to wear with a relatively low friction coefficient. This was clearly the better option and a good substitute for ball bearings additionally maintaining low weight and preventing a galvanic reaction. [98]

### Tool quick release system

#### *Introduction:*

A quick release system for fastening different tools to the hub is essential for efficient and easy change when production parameters vary. In addition, it allows for spinoffs resulting in a robot for more versatile operations than just pick and place. This task is a B-requirement, should be implemented and a manual operation performed by the operator or service crew. Different solutions have been researched, which includes sliding and locking as well as snap locking. Furthermore, all parts should be cost-effective and easily maintainable by crew during scheduled maintenance. Components should also be less complex due to higher risk of component failure for more advanced subsystems. Hence, it should be time efficient to change tool, therefore quick release system.

Initially, the snap locking system used on hydraulic pressure hoses came to mind. This system can withstand high pressures but is a complicated sub system with multiple failure points. Parts needed would have been ordered and this could end up being costly. Even though there is no requirement for implementing vacuum tubes and a pressure gauge it cant be left out. By

using a snap locking system, the suction of vacuum could be integrated into the quick release system and would be the fastest option in addition to tool change. Since the packing machine is used for moving chip bags it would be logical to assume that the vacuum created is in the low to medium range, somewhere between 100kPa and 100MPa where a premade part could easily be found and implemented. But this is not a cost-effective solution and has more to do with the 1:1 model and not our 1:2 scale. So, another less complex solution shall be found and simultaneously maintaining the tool hubs structural integrity ensuring a extended lifespan of components used. [99]

Furthermore, the sliding and locking ideas includes the same principles used on mounting larger cameras to a tripod. Utilized by sliding the part being fastened in a machined track on the mounting hub. Then the parts will be fastened with a screwable plate pushing a part of the mounting hub into a machined path on the fastening part and locking it in place. Additionally, the part being fastened could slide into a machined lane and mounted on the hub by screwing it and locking the components together. Lastly the part being fastened could be placed within a machined pit on the tool hub, tracing the edges which keeps the part in place by either screwing it together or locking it place by using a screwable plate. There are many different mechanical methods for locking two components together, and within sliding and locking the focus is aimed at the most relevant ones with empathy on factors mentioned in the first paragraph.

*Alternative 1 (Camera mount):*



Figure 97: Camera mount [3]

In this alternative the opposite profiles of the ones in the figures are thought to be machined from aluminum and implemented into the tool. The locking principle seen in the figures would have been integrated into the tool-hub design. The tool itself should have a opposite trail and shape of the parts shown. It would then be slid into place on the tool-hub and locked in place with the locking tap on the side. This part shall then be locked in x, y and z direction ensuring a snug fit and low rattling during operation. Additionally, machining out the design on the tool-hub would be demanding and costly due to tolerances and multiple machining processes required. Furthermore, the locking force shown in the figure is mainly acting in the width direction on the adjustable threaded locking tap. It is the same principle used for locking a bicycle seat in the right height, using the friction force between materials to its advantage. Most of us know that this locking method under load may result in failure if not tightened enough. Additionally, threads receiving prematurely wear after several cycles. This could lead to wear on both the locking screw and tool-hub threads which eventually results in costly repairs where not only the locking screw and tap needs to be replaced, but the tool-hub itself, leading to the unnecessary costs of machining a new hub. Furthermore, thread wear will lead to tool vibrations and rattle, making it less accurate which could impact the packing process, robot accuracy and operability. Including the possibility of damaging MACF by not only being inaccurate but also potentially losing the grip on the tool, allowing it to slip out of its path in lengthwise direction and being dropped within the MACF, possibly resulting in severe damage to other components such as the assembly line which could start a domino effect of failures.

within the packing machine resulting in production stop and costly repairs. The solution is out of the picture even though it is an effective quick release option.

*Alternative 2 (Transducer knob):*



Figure 98: Transducer knob [4]

This is a similar method to the classical camera mount and known as a transducer knob. Normally used for transducer poles. Another use area is for mounting a rowing ore on the cylindrical half sphere allowing for rotational movement. The principle would have utilized a machined part from aluminum with a path, where the knob is pushed in and fastened by using threaded holes and countersunk bolts. This solution would ensure a very tight fit and minimize

the tool rattle possibility during operation. It would also lock the tool in all directions and keep it in place. If red Loctite was put on bolt threads, it is very unlikely that they would get untwisted during rapid movements. The tool could be swapped by simply unscrewing the four bolts and sliding it out from the machined path. It is an easy principle to implement on the hub for Senior since the part could be split into two. Additionally, the dimensions on the tool neck would depend on the width of the opening where the knob is seen in the figure. The locking plate could be redesigned with a larger opening but there is still a tearing risk of bolts or plate with the low tolerances required, due to limited space in x, y and z direction. Furthermore, the method is a safer option than the one above since this mechanical design will ensure a perfect fit and stronger force locking the tool in all directions.

*Alternative 3:*

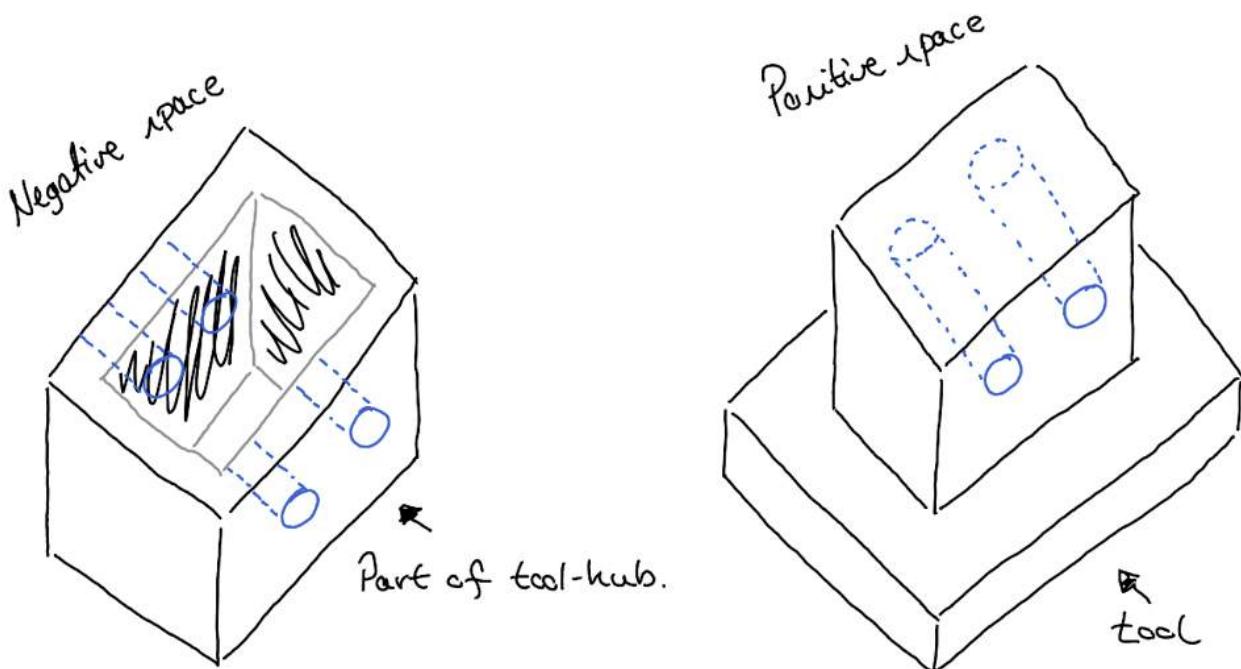


Figure 99: Alternative 3

Another sliding and locking idea were to machine out a negative space in the tool hub with drilled holes through, like the one shown on left side above. The drilled holes could be clean holes for the use of hinge pins and locking rings, or fine/coarse threaded holes on one side for the use of bolts with or without Loctite. Furthermore, the positive space to the right would be a design machined out on the tool itself, or a single part mounted to different tools with bolts and connected to the tool hub. Either the methods mentioned above or other quick release alternatives such as locking screws or ring bolts with a cotter pin. In this alternative, all the tool mass is resting on the hinge pin or bolt and acting in a downward direction. So, it is essential to choose the right dimension and material. Assuring that the forces acting on the circular cross-section is kept safely beneath the yield strength of the components. And by using steel, it would ensure minimal to no fatigue wear on the fastening component after a repeated

amount of operating cycles, potentially giving the fastening component infinite life. If failure were to occur, the tool could drop onto the conveyor belt or other parts inside the MACF potentially resulting in costly repairs and downtime. In conclusion, this method would be easy to machine and implement into both the tool hub and tool additionally being a cost-effective alternative if the forces acting on the cross-sectional area are thoroughly researched, tested, and documented. [Bib2]

*Alternative 4:*

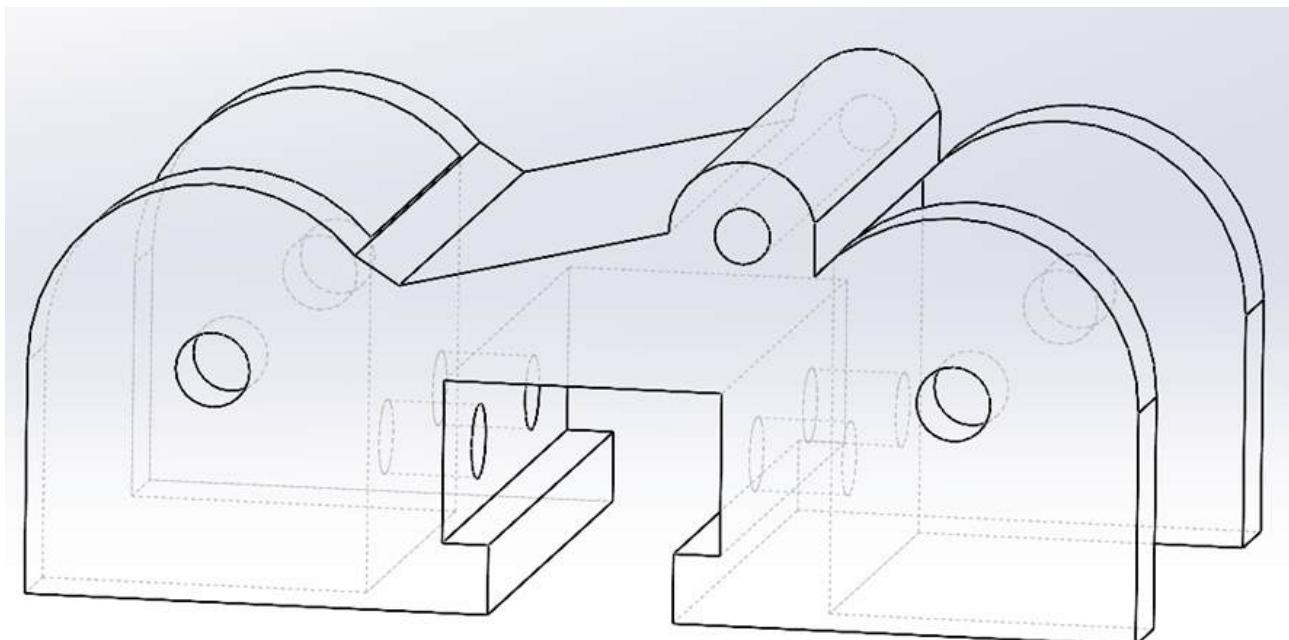


Figure 100: Alternative 4

Alternative four is a combination of the transducer knob and the one above. Initially it was meant to have a machined track through the square seen in the hub center shown above. This would relieve the bolts or hinge pins from loads in z-direction (downward) and the forces acting on the cross-section would only be in x-direction (width), and the tool would be locked from movement in y-direction (lengthwise) by the hub itself. This seems to be the more cost-effective and safe solution for a 1:1 scale.

*Alternative 5:*

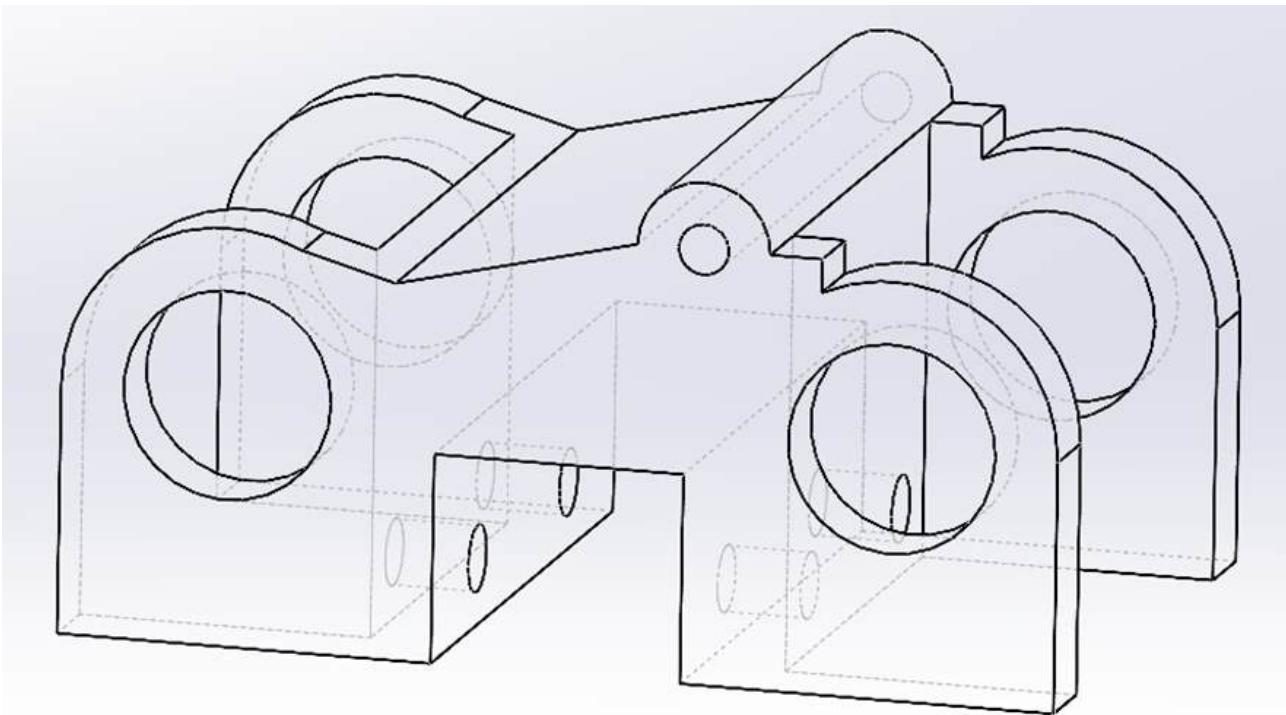


Figure 101: Alternative 5

In addition to the alternative above, the tracks must have a chamfer to realistically be machined cost-effectively. This depends on the available tools at TEs workshop since the pin mill must be long enough for the cut. It would also need additional mounting positions during the operation, which could lead to lower tolerances on the part, since the machines zero-point varies and must be set again for every new mount. Furthermore, for simplicitys sake the tracks could be removed on our 1:2 model since the tool is only fictional for the Senior model. Its purpose is to visualize the functionality of a pick and place movement without the pick and place operation of chip bags.

#### **Design and development:**

This chapter is about the design throughout the development process.

##### *Design iteration 1:*

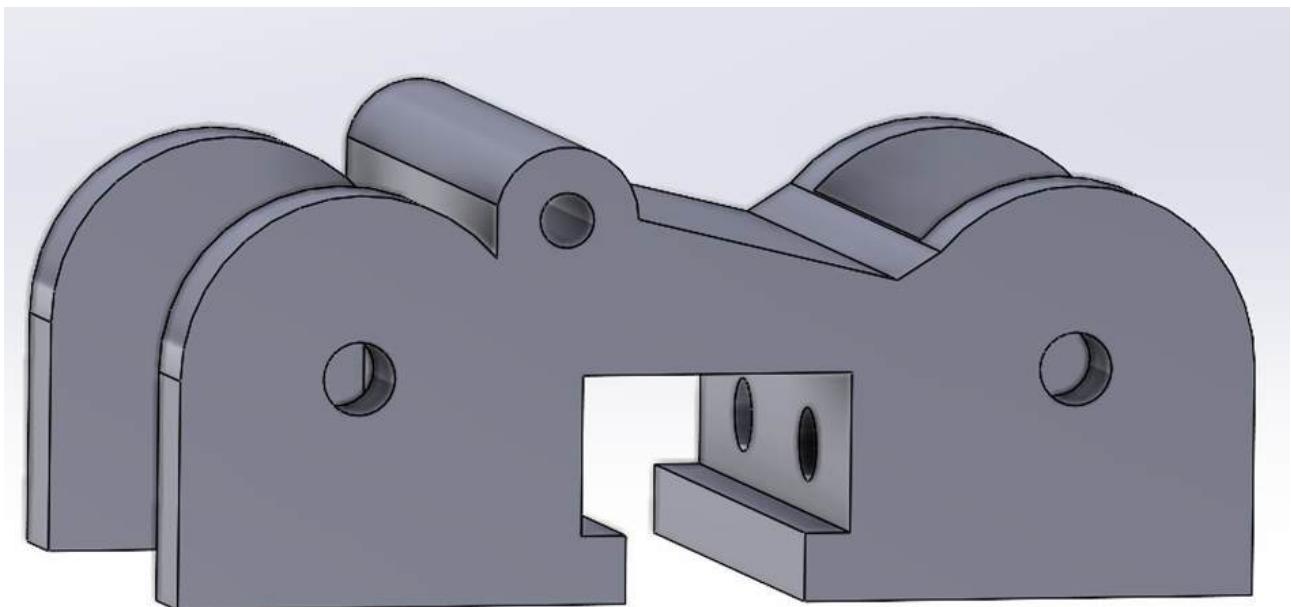


Figure 102: Iteration 1

In the initial hub design seen above, the passive arms are mounted either on the inside or outside holes at the length ends with a 50mm center distance. The mounting process is explained above in the interface chapter. In addition, the tool is slid into its negative space at hub center and mounted by using two hinge pins or bolts. Furthermore, the stabilizer arm positioning where not yet determined and was planned to be mounted in the upper hole. This design had a few complications due to impossible machining processes. It also required chamfers on all inside corners. Additionally, the sharp corner at the top left curve below the stabilizer hole is not possible to machine using the equipment available at TE, which lead to another iteration.

*Design iteration 2:*

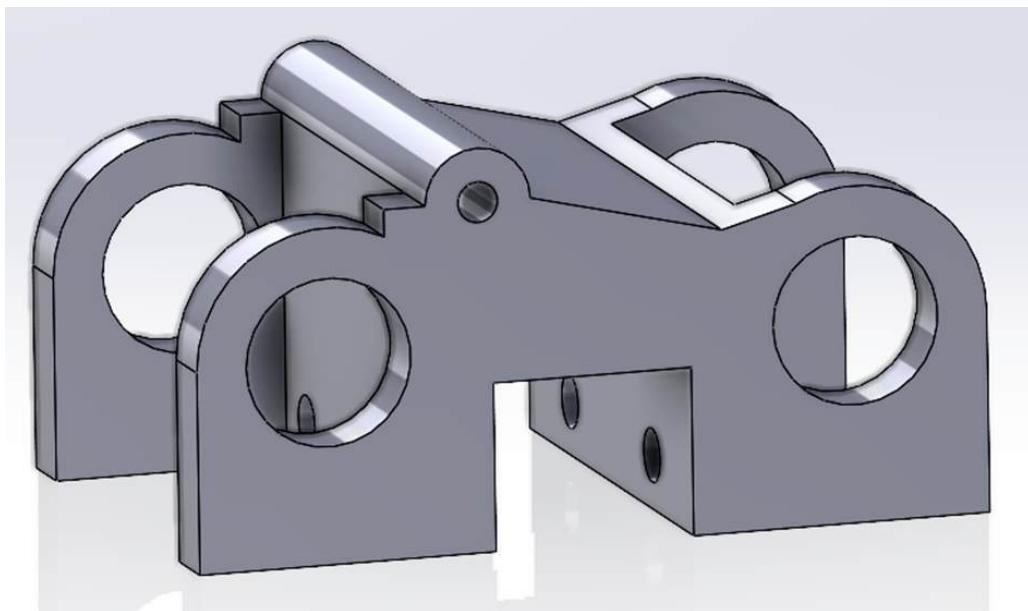


Figure 103: Iteration 2

The next iteration is based on the first one. Mounting holes for arms was played with, trying to figure out how large they must be. This aided the decision-making process of dimensions for the washers with and without a collar. The center shelf for a tool was also removed and it turned out to be difficult solving the sharp corner issue mentioned above. Which lead to a new design from scratch since it was easier and faster solution than trying to figure out how to solve the modeling issue at hand. Editing this part any further would-be time demanding due to the parts poor design foundation of features and sketches.

*Design iteration 3:*

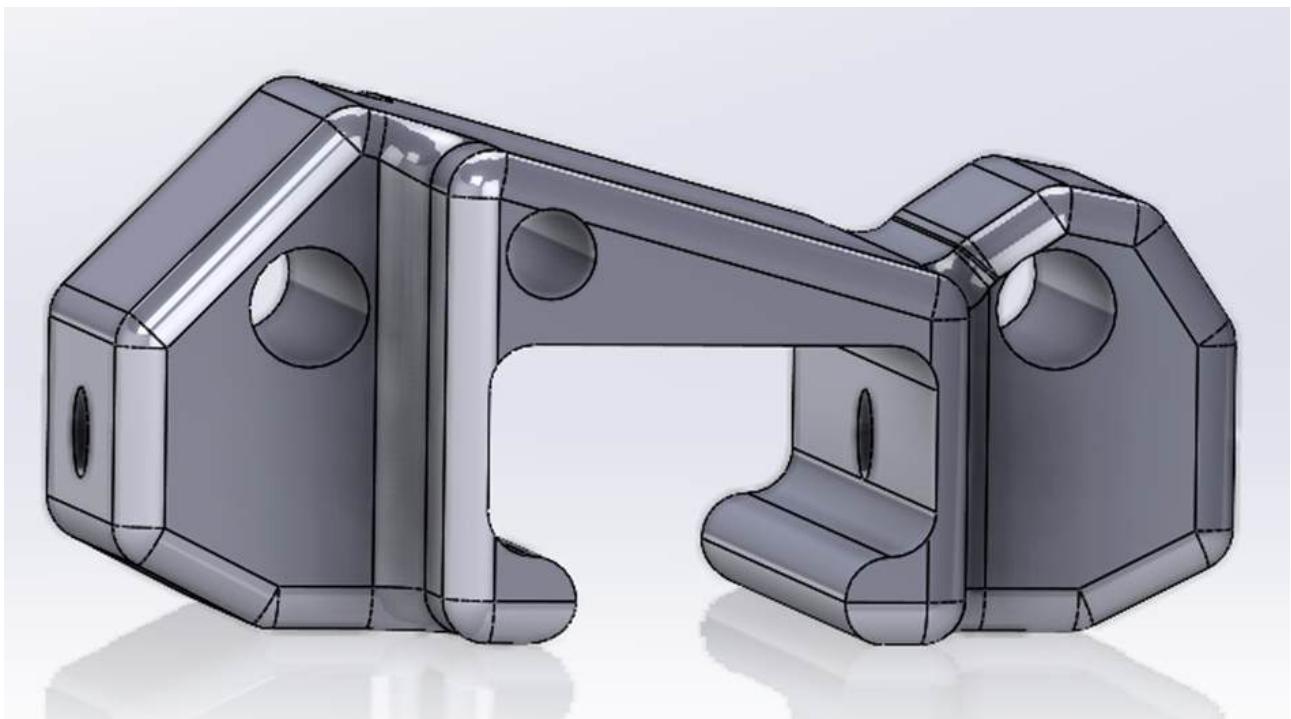


Figure 104: Iteration 3

A new design from scratch led to some few design modifications. Including, locking the mounting position for the arms on the outside and deciding the arm width. Additionally, adding the tool resting shelf into the design again and removing one of the quick release pins/bolts. Fillet was added to all hub edges to decrease stress concentrations on the part. Furthermore, the right most mounting hole for the arms could lead to tear since there is minimal material supporting it. This design turned out to be less favorable machining because there is no flat surface on the top holding the hub in place during the required milling operation. It would require a lot of different positions during the process, which reduces tolerances and increases expenses. Mostly due to new mounting positions and this is time demanding. Fewer mounting positions when machining is beneficial.

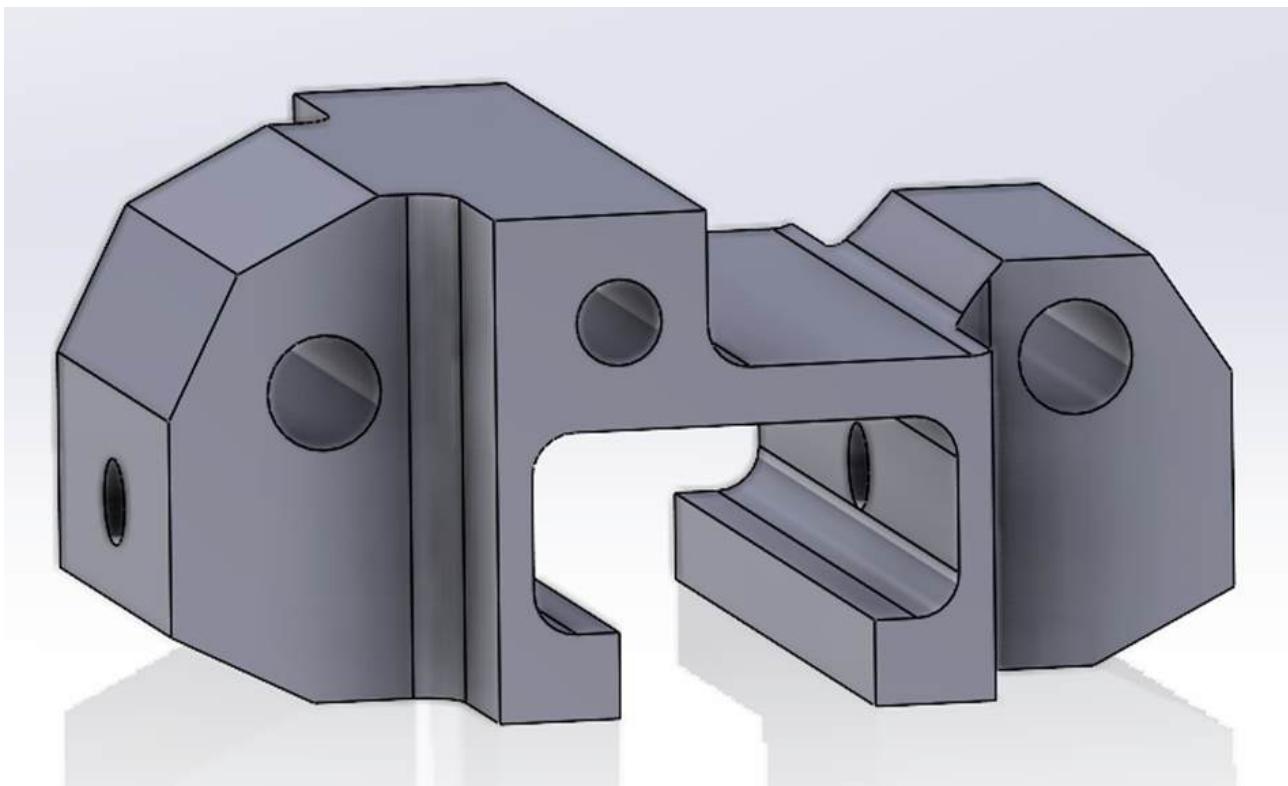


Figure 105: Iteration 3.1

The outside fillets ended up being removed since it reduces machine time. Inside chamfers had to remain due to the minimum pin mill radius allowed to use (R3). TE has smaller tools, but those are for more special situations and increases production time. There were also some difficulties with the chamfer furthest right at the plane surface to the right of stabilizer hole. This edge was sharp and facing upwards when it was intended to be a 90-degree angle. Furthermore, the stabilizer hole was not determined. This led to another iteration that made it easier to move it around trying to figure out the perfect mounting position.

*Design iteration 4:*

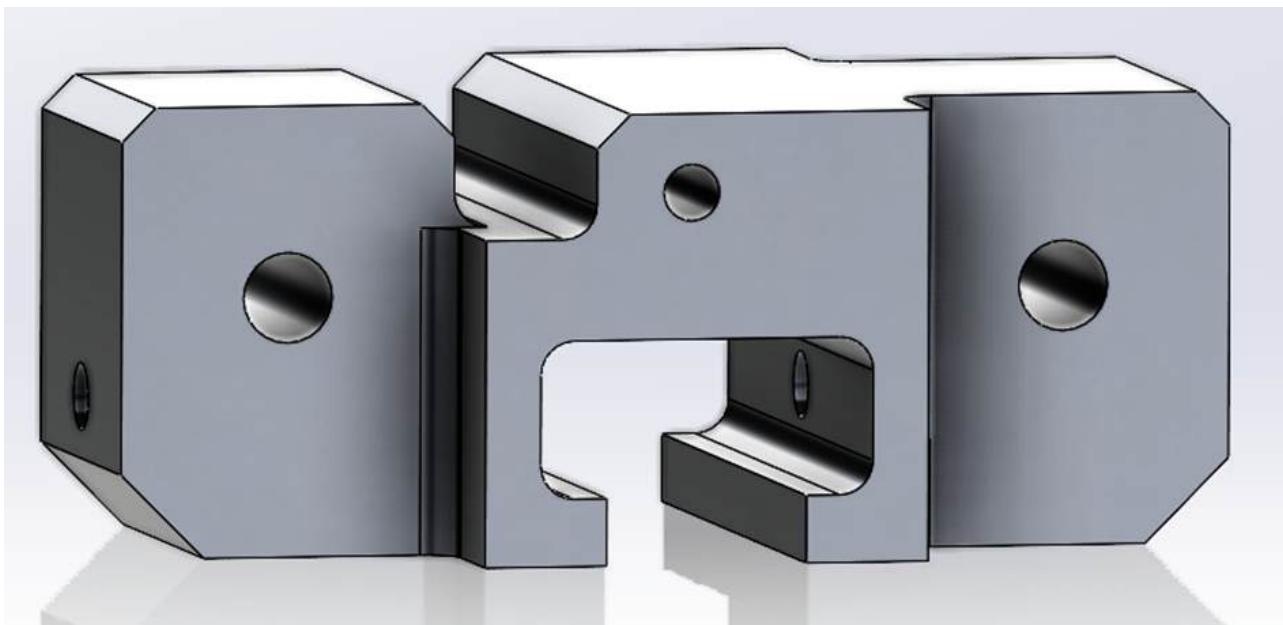


Figure 106: Iteration 4

For simplicity this design iteration ended up being bulkier until more of the dependent parameters could be determined. Resulting in a time efficient development process. At this stage a big challenge arose regarding LSS-HS1s lifting capacity and output torque.

<b>Mass properties of ToolhubIteration3</b>	
Configuration:	Default
Coordinate system:	-- default --
<b>Density = 0.002700 grams per cubic millimeter</b>	
<b>Mass = 350.124619 grams</b>	
<b>Volume = 129675.784907 cubic millimeters</b>	
<b>Surface area = 27202.325180 square millimeters</b>	

Figure 107: Iteration 4.1

This bulky tool hub had a mass of about 350 grams and required to be reduced as much as possible. Gear testing was initiated at the same time, leading to a new tool hub design from scratch.

*Design iteration 5:*

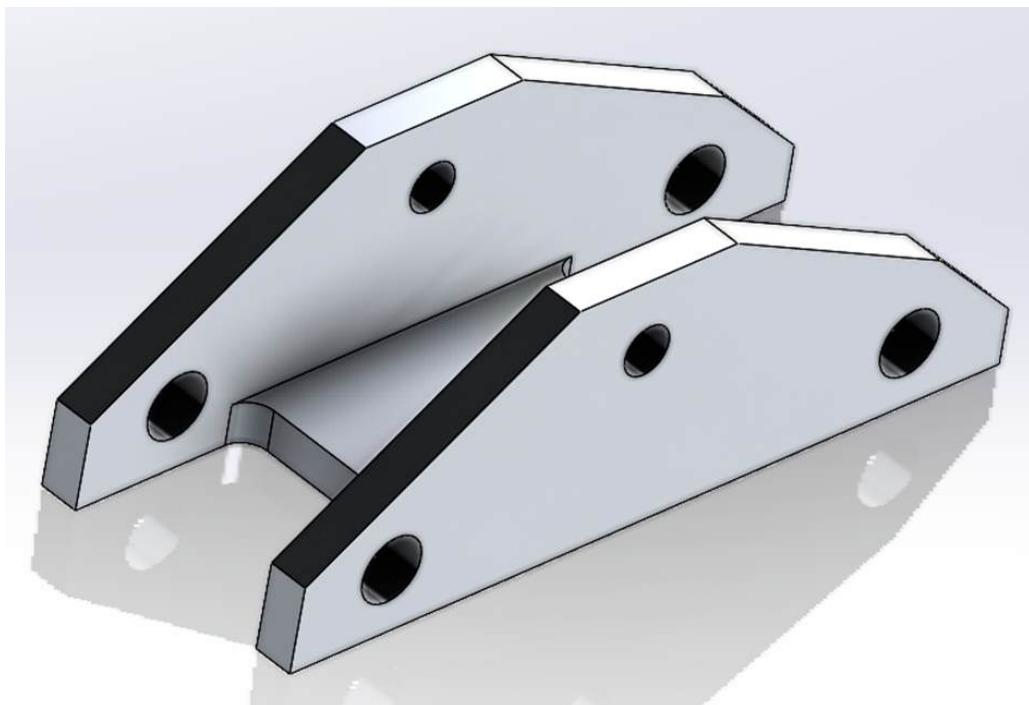


Figure 108: Iteration 5

This design was made as light as possible to reduce the maximum amount of mass. That included going backwards on the design and mounting the arms either on the outside or inside. At this stage, dimensions for the stabilizer hole at the top was decided. Additionally, removal of the quick release system resulted in a final mass of 71,4 grams and this could barely be reduced any further.

<b>Mass properties of TollhubRedusert</b>	
<b>Configuration:</b> Default	
<b>Coordinate system:</b> -- default --	
<b>Density = 0.002700 grams per cubic millimeter</b>	
<b>Mass = 71.407028 grams</b>	
<b>Volume = 26447.047260 cubic millimeters</b>	
<b>Surface area = 13335.666990 square millimeters</b>	

Figure 109: Iteration 5.1

The walls were 4mm thick and could be modified to 3mm, but then new parts for arm mounts had to be researched. Additionally, this is sheet metal thin and could potentially be produced by using a water-jet cutting the outline contour and holes of a flattened model with the same design. Afterwards, it could be bent by utilizing a metal sheet bender. This option was also available at TE but not in Hønefoss. The hub had to be sent to TE located in Moss and it would

take at least four weeks receiving the finished. Due to limited time left, this option was excluded.

*Design iteration 6:*

After testing with 1:2 gears, the results confirmed that the motors could lift at least 1200 grams bringing about the last iteration.



Figure 110: Iteration 6

The mass of each part connected to the LSS-HS1s on the fastening hub had to be low but not minimalistic low. This design is based on the last simplified iteration, maintaining low mass even with the quick release integrated.

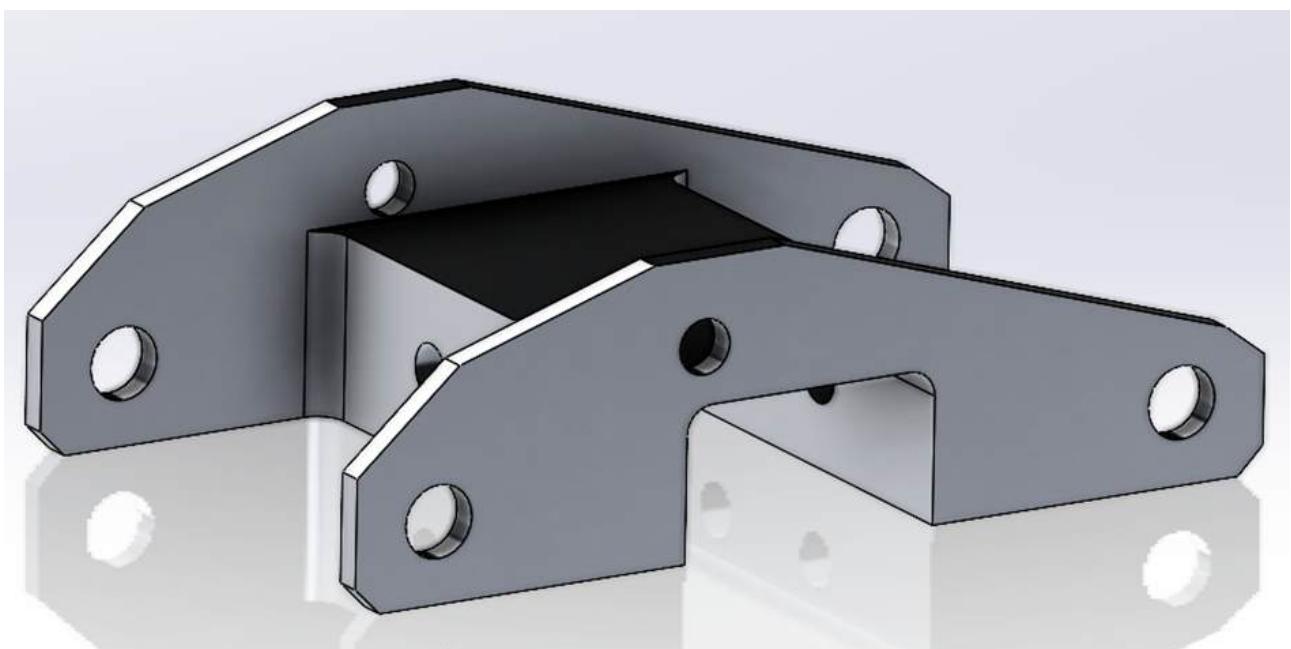


Figure 111: Iteration 6.1

Furthermore, modified to simplify machining processes and reduce production time by removing a few contour chamfers and tool shelf. Resulting in better structure on 2D measurements sent to TE for production. Additionally, the hub width was determined because the bracing design for sideway stabilizing arms was finalized. Lastly a R3 fillet was added on inside corners due to the Ø6mm end mill tool. As mentioned earlier, this fillet also reduced stress concentration in these areas.

**Exploded view and bill of materials:**

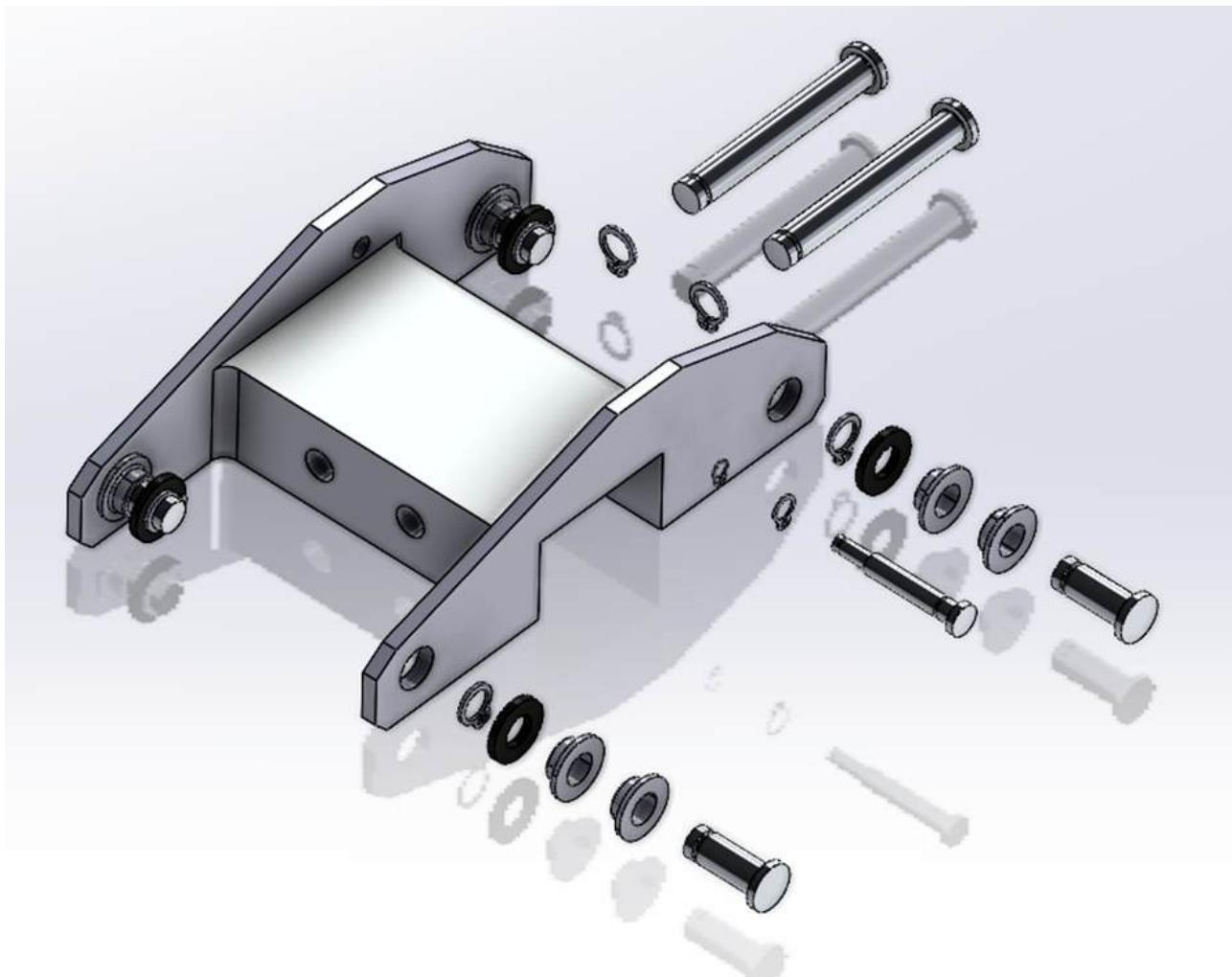


Figure 112: Tool hub exploded view

Shown above is an exploded view of the assembly with all its components. These are mated together as it is intended to be assembled. It has four unique parts for production at TE and five unique parts being ordered from either MiSUMi or IGUS.

Table 15: Tool hub Bill of Materials

Pos.	Title	Item no.	Material	Mass	Manufactured	Supplier	Part name	Article no.	Size	QTY
1	HingepinArm	160001	6063-T6	1.41	Yes					4
2	HingepinQuickRelease	160003	6063-T6	3.51	Yes					2
3	Toolhub	160004	6063-T6	83.57	Yes					1
4	Stabilisator adapter TOOL	160002	6063-T6	0.90	Yes					1
5	Flange bearingID6	300022	N/A	0.15	No	Igus	Sleeve bearing with flange	JFM-0608-04	d1 = 6mm, d2 = 8mm, b1 = 4mm	8
6	ThrusterBearing 6x1.5	300050	N/A	0.13	No	IGUS	iglide® G300, thrust washer	GTM-0612-015	d1 = 6mm, d2 = 12mm, s = 1.5mm	4
7	RetainingringØ3	300031	Spring steel	0.00	No	Misumi	Retainingring C-Type	STWN3	Nominal Ø3mm	1
8	RetainingringØ4	360050	Spring steel	0.01	No	Misumi	Retaining Rings	STWN4	Nominal shaft Ø4 C-type	1
9	RetainingringØ6	300032	Spring steel	0.02	No	Misumi	Retainingring C-Type	STWN6	Nominal Ø6mm	6

The bill of materials is showcasing each part used in the assembly. This includes its unique item number, material, mass, if its manufactured or ordered as well as supplier, part name, article number, size, and the total quantity of each part.

**Mass properties of Asm Toolhub**  
**Configuration: Default**  
**Coordinate system: -- default --**

**Mass = 98.99 grams**  
**Volume = 37822.63 cubic millimeters**  
**Surface area = 28673.08 square millimeters**

Figure 113: Assembly mass

Shown above is total mass for the assembly which is below 100 grams. This is a big difference from the earlier bulky design with fastening components excluded at about 350 grams.

### Parts ordered:

#### *Retaining rings:*

These are ordered from MiSUMi. Further details are found in Bill of Materials. [87]

#### *Thrust washer:*

These are ordered from IGUS. Further details are found in Bill of Materials. [87] [100]

*Sleeve bearing with flange:*

These are ordered from IGUS. Further details are found in Bill of Materials. [87] [101]

**Parts produced:**

*Hinge pin:*



Figure 114: Hinge pin

The hinge pins are produced using a lathe an operation known as a turning. In this operation the work-piece is rotating while the tool is stationary mounted on the tool-holder. Tool is then fed lengthwise on along the work-piece at a predetermined depth and feed rate. Material is then removed at a rate known as MMR which is dependent on depth of cut, feed, and the rotational surface speed of the work-piece. The process is repeated until desired diameter is achieved. Furthermore, the groove for retaining ring is cut and the component is parted away from its cylindrical start piece at the hinge pin head. Additional, beginning cuts are often rough cuts, typically at high feed rates and larger depths followed by finish cuts at lower feed rates and depths to produce a good surface finish with acceptable tolerances. In hindsight, it should have been used a cotter pin instead of a retaining ring since the retaining ring is placed utilizing a special tool. The cotter pin would have been put through a radially centered hole on the hinge pins cylindrical surface, at the same location as the retaining ring. [102] Additionally, an E-Type retaining ring could have been used instead of the C-Type requiring no special tool. [103]

*Tool hub:*

This part is very demanding and complicated to machine. It will be produced by utilizing a multi-axial CNC milling machine and must be mounted in a variety of positions and orientations, which can result in lower tolerance because the new zero points required for each mount. In addition, holes are drilled during this process. Furthermore, the machining procedure is time demanding and costly. In hindsight the hub should have been remodeled and split into three parts like the fastening hub. Then put together using bolts or maybe welds. If the part was split, the outside plates could have been cut with a water-jet and the middle including the quick release milled and drilled in two or three positions. [Bib2]

### 6.1.11 Production, deliveries and delays

AK | SG

IGUS and MiSUMi were suppliers capable of delivering components quickly and reliably. Unfortunately for the group, this proved not to be the case regarding our order. The IGUS delivery was postponed due to the lack of availability of components in the ordered articles. In addition, high demand for production from the main supplier. Consequently, there were additional parts not accounted for from our original order.

Originally the expected production time was estimated to be four weeks. When delays occurred with the design and ordering process, there was a low demand in the production facilities at TE. But the production got busy when our order finally was delivered.

In conclusion, there were unexpected delays in the following points: Composite production, polymer bearings, and frame parts from Item, in addition, the turning and milling of aluminum parts also experienced difficulties.

### 6.1.12 Simulation

SG | AK

**Introduction:** To asses the structural integrity of DuoArm, a FEA static simulation is conducted on the passive arms to obtain theoretical data. The objective is to analyze tearing of arm-connections and compare acquired information with the planned tension tests at USN's lab. The connection point chosen is of most interest, since it determines maximum load that the CFRP arms can withstand. A few simplification has been taken into account for practical reasons.

**Design and simplifications:**



Figure 115: Passive arm with constructed grip

In the figure shown above, the CFRP passive arm is colored blue, bolts in red and constructed grip in green. The constructed blocks are bonded to the bolt in one end, assuring that applied external load is acting in the correct direction. The material used for grips is titanium Ti-13V, it has a higher shear and elastic modulus than the CFRP composition utilized. The intended function of these are to simulate the grip of a tension machine, stretching the part upwards. In addition, the bolts on the other end is fixed in order to perform a theoretical tensile test. Bolts simulated is extruded to be the same size as the passive arm's hole.

The analysis is performed without using wear components in connection interfaces, since these are made from a ductile material, nylon. The resistance received from these parts is negligible compared to CFRP. These should also be neglected in the physical tensile test. The arms are made from CFRP, a brittle material and SW simulations cannot give relevant stress data on brittle materials, since FEA in SW utilizes Von Mises stress to calculate and visualize the analysis. For brittle materials Columb-Mohr stresses must be utilized and the only relevant data gathered from this in a static simulation is FOS. Furthermore, from the FOS it is possible to obtain the arms maximum load capacity by modifying the forces acting on the part. Additionally, running the simulation multiple times until a FOS of 1 is reached. Or simply by using this formula for brittle materials:

$$FOS = n = \frac{Shear - stress}{Ultimate - tensile - stress} \quad (5)$$

**Set up:**

*Connections:*



Figure 116: Connections

The figure above is showcasing in light blue the different local interaction. It includes bonding on the gripping blocks to the left, and contact between bolt and CFRP.

*Fixtures:*

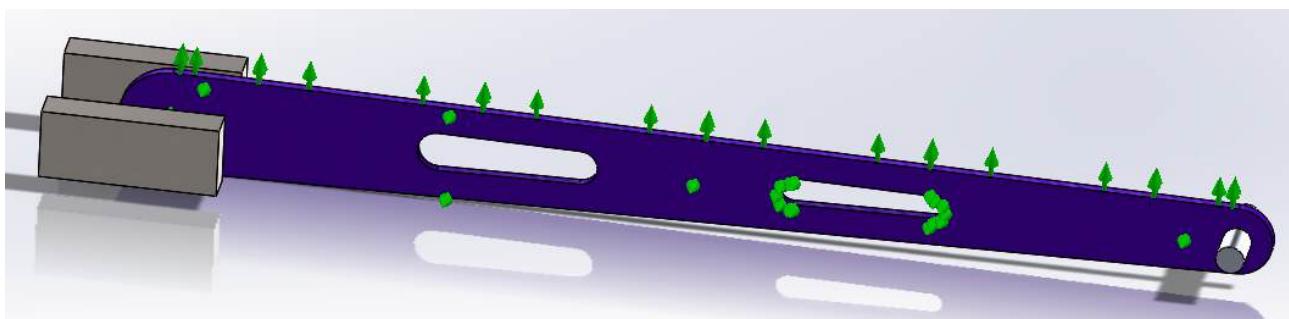


Figure 117: Roller/slider

Firstly, a roller and slider fixture was placed on the width and side, ensuring that the part can be stretched lengthwise according to applied force.

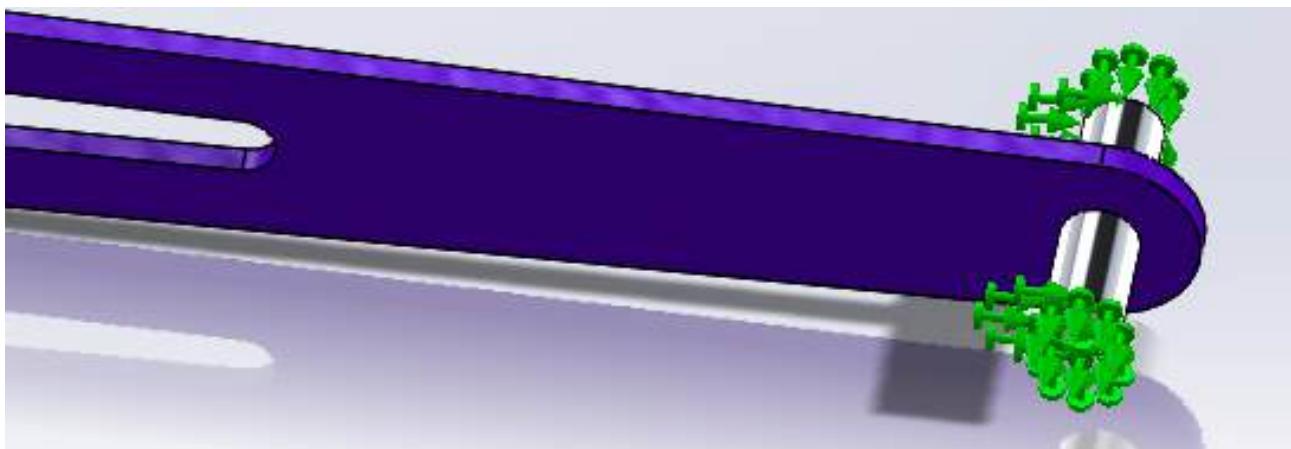


Figure 118: Fixed

Secondly, on the ends of the bolt shown above is a fixed fixture. This ensures that the lower part is fixed like it would be in the tensile test machine.

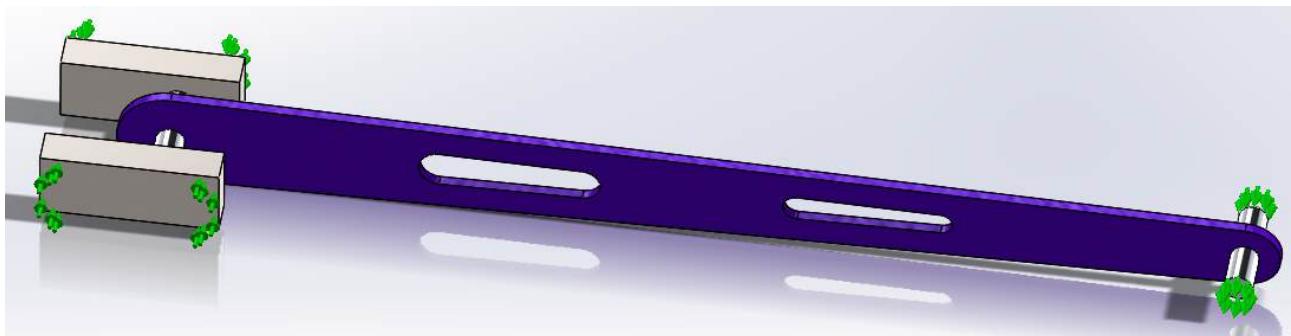


Figure 119: Roller/slider

Thirdly, another roller and slider had to be added on the components shown above, since the part could flex sideways. This was discovered by utilizing the motion study.

*External loads:*

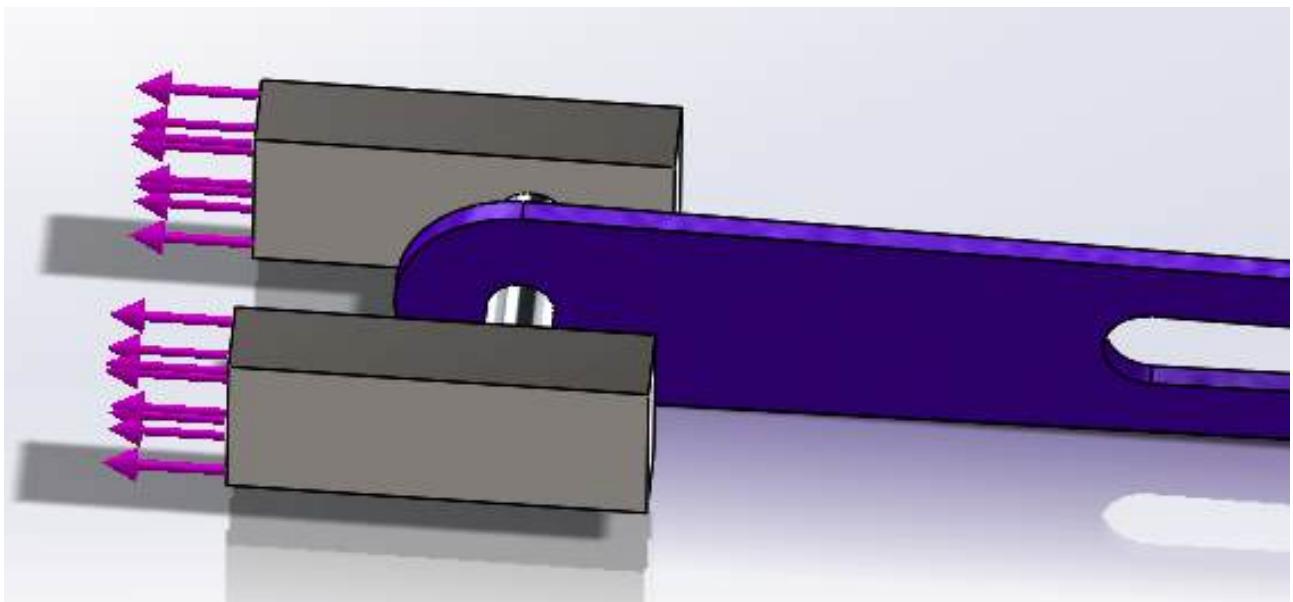


Figure 120: Force 5 000N

Regarding the applied load, a force of 5 000N was evenly distributed and added to the grips top faces. This equals a mass of 500kg. Different load values were tested to achieve a better understanding of the results given. *Mesh:*

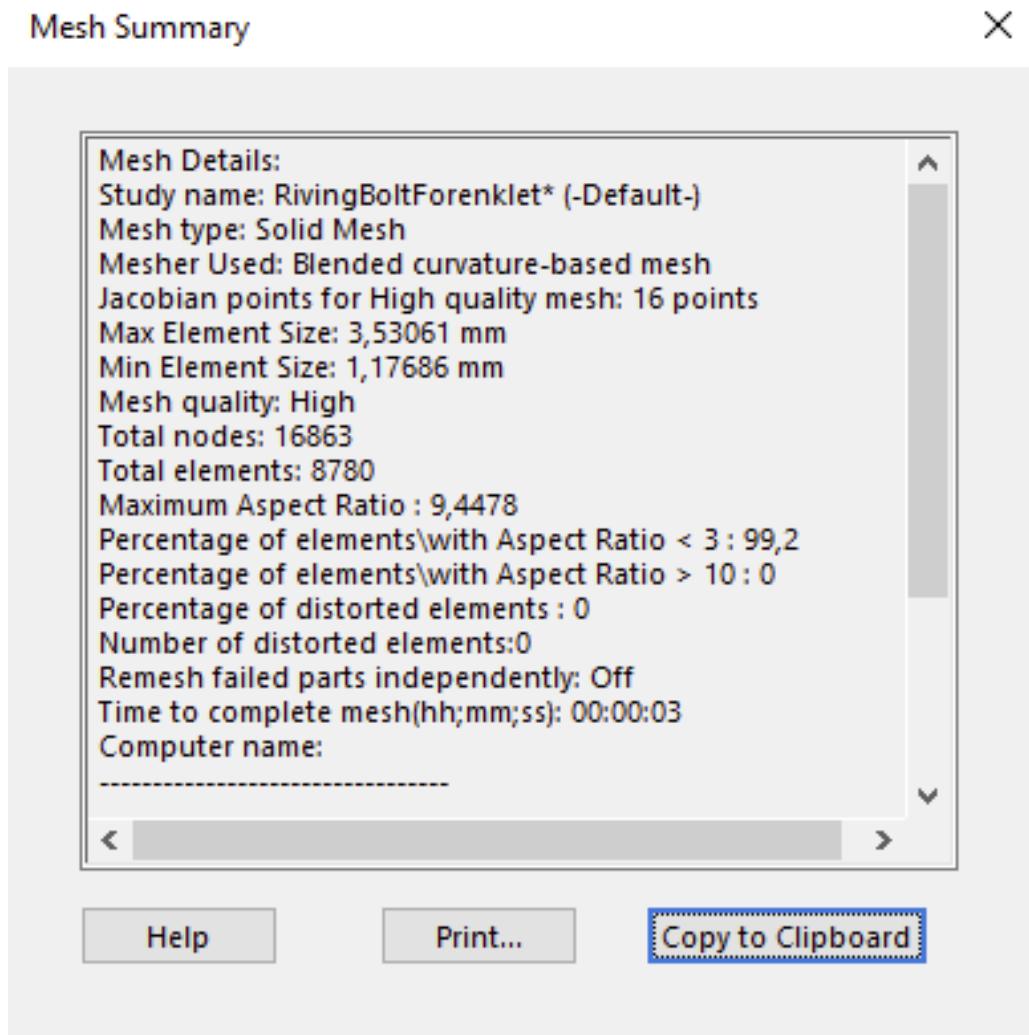


Figure 121: Mesh quality

Meshing the FEA components with an aspect ratio  $< 3$  at 97% or higher means that the components are meshed properly, assuring that the positioning, connection and amount of nodes are at a adequate level.

*Results:*

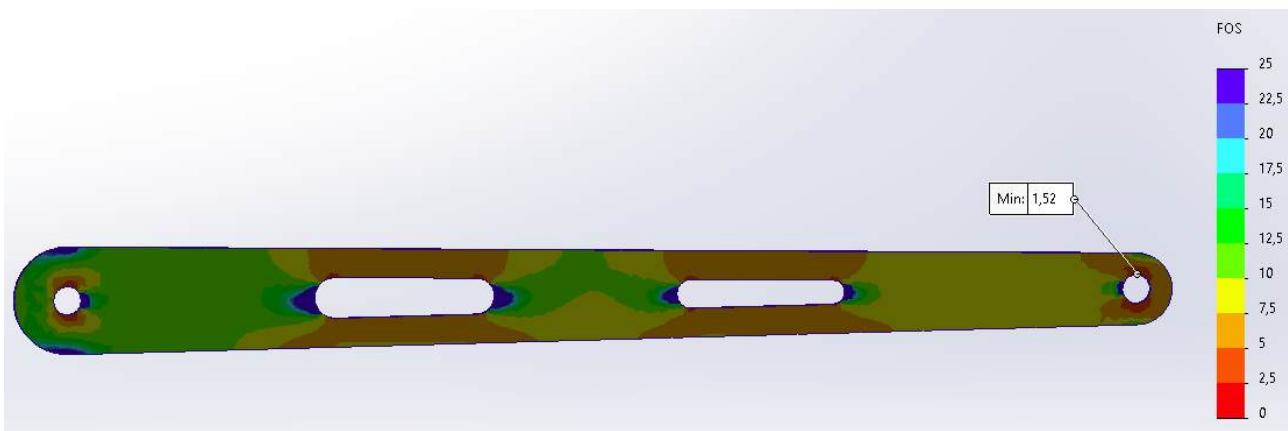


Figure 122: FOS

A plot of FOS from the analysis is shown above with the point of minimum value tagged. The bolts and grips are hidden to better reveal stress concentrations in these areas. On the right side is a color scheme for the different FOS values. These are visualized on the part, showcasing the variety in FOS. The regions in red, contains the largest amount of stress, meaning that these zones are most vulnerable to shear. The weakest point with an FOS of 1,52 is also the logical point of failure, being the area with the least amount of material to prevent stress propagation.

#### Defining material properties:

The simulation result is fairly reasonable for CFRP. It is remarkable that such a small plate is able to carry an evenly distributed load at 500kg, pulling in one end with a FOS of 1,52 on the other. The FOS is at the parts smallest cross-sectional area with a plate thickness at 3mm and shortest distance from hole contour to outside at 5mm. This implies in practical sense that the CFRP passive arm can withstand a load of 750kg before tearing. In reality, this is unlikely since the simulation is purely theoretical with perfect conditions. Additionally, the reason why it should be done tensile tests with pieces cut out of delivered plates. In addition, there is a  $\pm 0,2\text{mm}$  plate thickness variation that affects the properties.

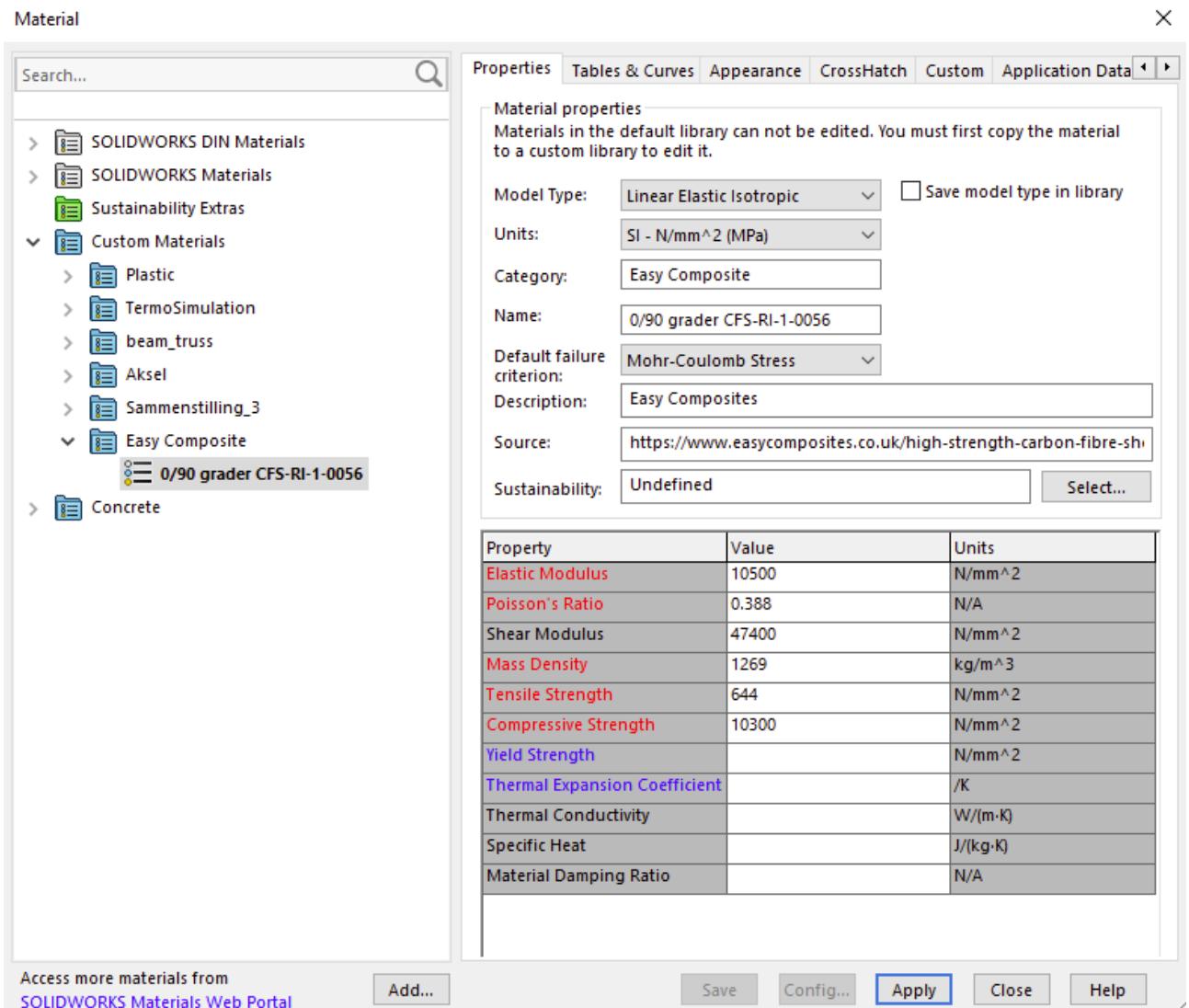


Figure 123: Linear Elastic Isotropic

The material properties was defined according to data derived from the products data-sheet found on easy composite. [93] Information available could only define the parameters for a model type where the properties does not vary with directions. This type is know as linear elastic isotropic. In addition, the properties was used for 0 and 90 degree sheet orientations, since the parts will be cut in this direction assuring optimal material properties. Because, tensile strength of fiber composites is dependent on fiber orientation and cohesion between fiber and epoxy, resulting in lower tensile strength in certain orientation. Additionally, default failure criterion had to be Mohr-Columb stress since this is utilized for brittle materials. Resulting in feasible data, despite the fact that plates ordered are orthotropic.

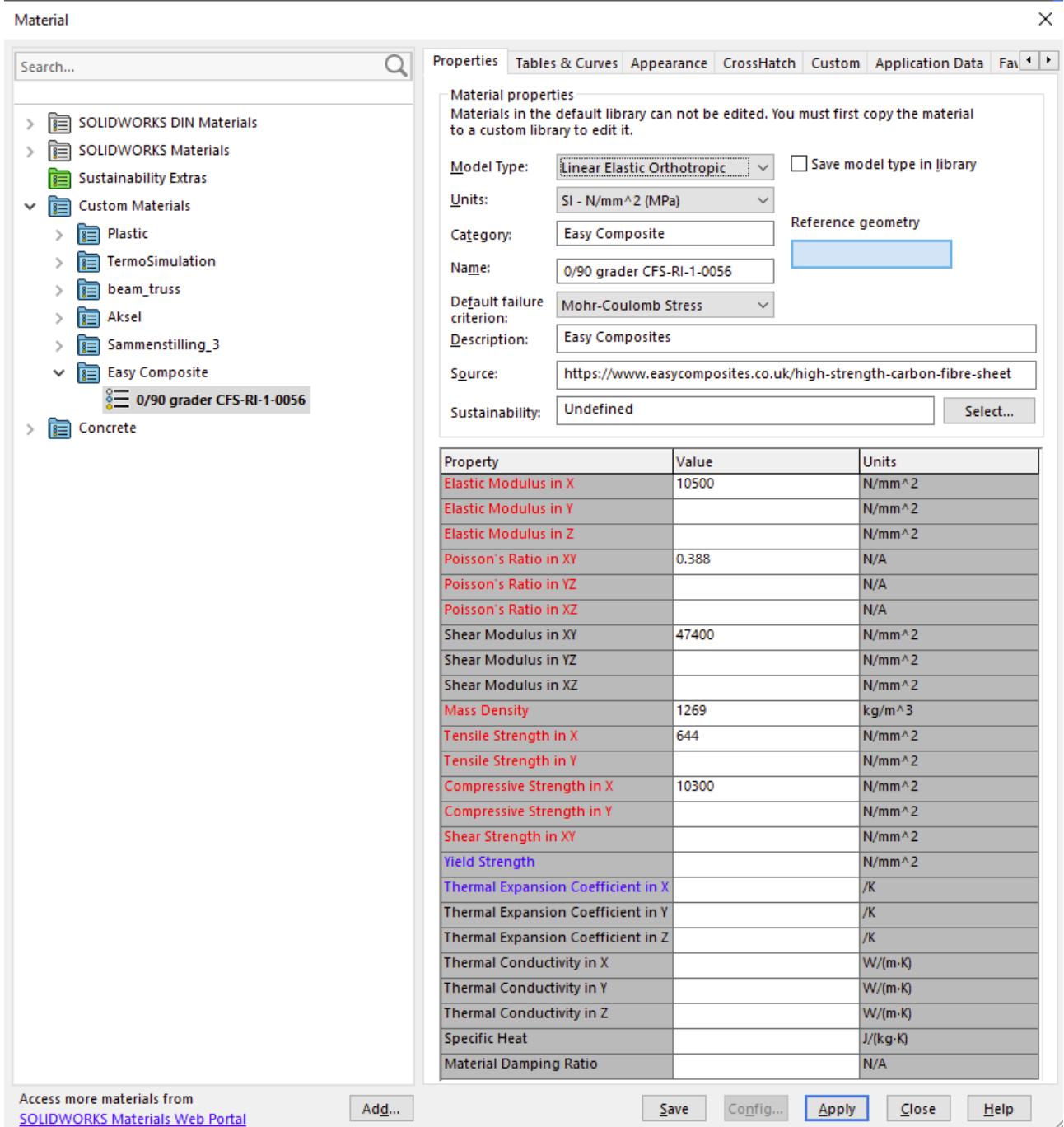


Figure 124: Linear Elastic Orthotropic

Achieving properly defined material properties for composites, the model type "linear elastic orthotropic" should be used. But in most situations, required data is not given in datasheets due to trade secrets and insufficient testing. Resulting in simulation simplifications and destructive testing with different orientations. In order to obtain a conclusion on material properties, multiple DT must be performed to achieve realistic product data.

### 6.1.13 Composites

AK,SG | LTR

**Introduction: SG | AK**

Composite materials are a composition of a matrix linked to fiber elements. Their properties derive from the chemical bonds between these and the construction of either unidirectional or bidirectional layers. It can also be a combination of both, depending on the intended purpose. They are categorized into three different groups, PMC, PMC, and PMC, determined solely by the utilized matrix. Composites are known for being anisotropic with varying properties and distinguish themselves from other materials due to their combination of strength, stiffness, and lightness.

**Fundamentals of composites: AK | SG**

*Definition:* "A composite material is a combination of two or more materials of distinctly different chemical or physical characteristics. Working together in collaboration, they create a new material combination of enhanced properties and characteristics that neither material on its own can provide." [104]

*Matrix, reinforcement, and additives:* As per definition, the Composite contains two main components, these are referred to as reinforcements and matrix. A reinforcement is usually the stronger of the two, providing the load-bearing properties. This is most commonly a fiber or flake. The matrix works as the binding component, holding the reinforcements together. This is usually an epoxy or resin blend which hardens in chemical processes. We classify composites in three main groups: Polymer matrix composites (PMC), composites where the fibers are embedded within a matrix consisting of polymerized organical compounds.[105] Metal matrix composites (PMC) are composites where reinforcements of ceramics or carbon fibers are bonded using, for example, titanium or aluminum. Lastly, Ceramic matrix composites (PMC), composites that utilize ceramic materials within a ceramic matrix, are usually used to withstand environments with high temperatures. [106] Additives in a composite are filler ingredients used to modify the composite's performance, including but not limited to fire retardants, UV inhibitors, conductive additives, and release agents. [107]

**Composite design: SG | AK**

*Design considerations:*

In general, the considerations are determined by the product's use area. It requires the adaptability to replace exchangeable components within assemblies, improving properties and typically at a reduced weight. It requires total control of multiple factors, such as corrosion, layer bonding, orientations, volume fraction, delamination from the machining process, etc. Thoroughly material investigations and testing has to be performed to obtain [108] [109]

*Mechanical properties:*

The properties of composites refer to their characteristics and behavior under load. These will vary depending on different factors. Such as the amount of layers and their orientations, fibers, matrix, and bonding. Composites can be designed and produced to match the desired requirements due to their beneficial versatility.[110] Mechanical properties are identified from reactions to applied forces, describing their resilience as tensile strength, hardness, and toughness. In addition, fibers are the main load carrier, and stresses are transmitted through the chemical bond with the matrix. Hence, continuous fibers are preferred for higher perpendicular loads because they reduce the number of connecting points between reinforcement and matrix. Furthermore, a higher fiber volume fraction increases the strength until it reaches an optimum value. Increasing any further results in reduced mechanical properties since it weakens the bond by mitigating the matrix purpose. The optimum values vary depending on the fiber and matrix. Additionally, fraction of fiber reinforcement is derived from the formula, Rule of Mixture, and obtained utilizing it with respect to densities.

$$\rho_c = v_f \rho_f + (1 - v_f) \rho_m$$

Figure 125: Rule of Mixture[5]. Where  $\rho_c$  is the density of the composite,  $\rho_f$  is the density of the fiber,  $\rho_m$  is the density of the matrix, and  $v_f$  is the volume fraction of the fiber.

If fraction of fiber reinforcement is not stated in the datasheet, then it can be obtained by reverse engineering the composite. This is achieved by carbonizing a composite piece in a burning chamber, where the matrix is completely scorched, leaving only the fibers. Furthermore, fiber can be weighted, and fraction of fibers calculated from the given density. [111].

*Testing:*

Physical DT of composites is a necessity, assuring the provided properties integrity. It is often a thickness variation with the same amount of sheet layers for the same product, affecting fiber volume fraction resulting in variable mechanical properties. Such as the CFRP plates ordered from Easy Composite [93], having a thickness variation of  $+0,2\text{mm}$ . Furthermore, physical testing was planned for two cases using a tensile test machine at Krona USN's lab. the first one reassuring possible deviations and the other testing the connection capacities as shown in simulations. These should be performed according to relevant standards. [112]

Planned test 1

Planned test 2

**Manufacturing process:***Methods: AK / SG Open molding:*

The method used for making CFRP flooring in our frame assembly, the fiber cloths are placed on a flat surface, applied with the desired matrix, and rolled to remove air. The resin cures in place and forms the finished composite. Though this is easy and cheap, the result depends on the fabricator and skill level. This can also be done in a vacuumed bag for better lamination results.

*Infusion: AK / SG*

The infusion process utilizes a vacuumed bag with two ports. One port is connected to the vacuum machine, and the other end is connected to a container filled with matrix. The fiber is laid dry, and the matrix is infused using vacuum pressure. When the resin has passed through without air bubbles, ports are sealed, and the matrix cures within the bag. Lamination quality is high with infusion as the fiber cloth absorbs the right amount of matrix.

*Pre-preg: AK / SG*

The process utilizes a pre-combined reinforcement and matrix. The pre-preg cloth is frozen to prevent the curing process. Then, the cloth is cut to shape and applied to the mold. Lastly, the covered mold is placed in a high-pressure oven called an autoclave, which initiates the curing process. This creates a precise fiber-matrix relation with high-performance properties. [113]

*Machining: AK / SG*

There are some important points when machining composites regarding their high strength, abrasive nature, and harmful micro-particles. For CFRP the most common machining processes are water-jet, drilling, and milling. As composites are highly abrasive, regular high-speed steels and carbide-cutting tools are prominent for a short lifespan. Therefore, CVD diamond-coated cemented carbide tools are preferred. Studies have proven improved tool life and high-quality cuts when utilizing a smaller cutting edge radius, rake angles of approximately 5°, 10-15° clearance angles, and high cutting speeds at 400-600m/min.

Prominent problems with milling are in regard to the BUE due to the reinforcement fibers and soft matrix. Additionally, delamination is especially prominent on CFRP with fiber a fiber-to-matrix ratio of 50-60% with thickness above 5mm.

*Adhesives and connections: SG / AK*

		SUBSTRATE 2					
		Metals	Fiber-Reinforced Epoxy	Fiber-Reinforced Thermosets	Thermoplastics	Other Thermoplastics	Fiber-Reinforced Nylon
SUBSTRATE 1	Metals	• Aluminum • Coiled Rolled Steel • Galvanized Steel	• Carbon Fiber (CFRP) • Glass Fiber	• Polyester (FRP) • Phenolic • SMC	• Polyolefin • PET	• Acrylic/PMMA • Polycarbonate (PC) • Rigid PVC and HIPS	
	FiberReinforced Epoxy	DP420NS DP125 Gray	DP420NS DP6310NS	DP6310NS DP8410NS	DP8010 Blue	DP8410NS DP6310NS	DP6310NS
	FiberReinforced Thermosets		DP420NS DP6310NS 760	DP6310NS DP8410NS 760	DP8010 Blue	DP8410NS DP6310NS	DP6310NS
	Thermoplastics			DP6310NS DP8410NS 760	DP8010 Blue	DP8410NS DP6310NS	DP6310NS
	Other Thermoplastics				DP8010 Blue	DP8010 Blue	DP8010 Blue
	FiberReinforced Nylon						DP6310NS

Figure 126: Bonding agents for combinable elements [6]

The bonding figure above displays a variation of 3M adhesive products utilized to bond lightweight composites to other materials. These variants are specialized for different combinations of polymer-based composites to each other and certain metals. These adhesives are specially developed to suit the specific bonding needs required by the two elements. Achieving optimal properties in the joint interface. The adhesives are epoxy-based, creating chemical reactions between the polymers and obtaining strong connections on an atomic level. Regarding connections, different types of tapes made from acrylic foam adhesive could replace bolts, rivets, and other fastening alternatives, increasing construction possibilities. In addition, different polymer-based adhesives, such as polyurethane and cyanoacrylates, could also be employed. [114]

## 6.2 The electrical setup

### 6.2.1 Power distribution (under development)

LTR | SG

Understanding the fundamentals of how a power supply operates is crucial when determining it. We are, therefore, interested in how the linear regulator and the switch-mode power supply operate.

A linear regulator uses a step-down transformer to lower the input voltage. After that, a rectifier converts this lowered voltage into Direct Current (DC). Filters are then applied to

clean the output by removing noise before sending it to the linear regulator. Keeping the noise low before it reaches the regulator is crucial as it will amplify the noise to a higher degree. A low pass filter is, therefore, often also implemented to remove the noise for a smooth DC voltage. Although this design is straightforward, it is not very energy-efficient, as it loses large amounts of energy in heat.

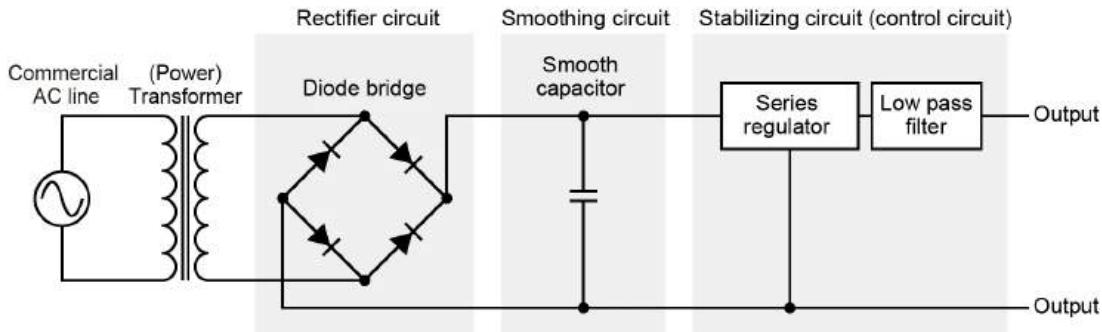


Figure 127: Linear power supply [7]

A SMPS operates with a unique approach. It initiates the process by employing an Electromagnetic Interference (EMI) filter, a crucial component that plays a significant role in cleaning the input signal. This filter is the first line of defense, ensuring the signal is free from EMI before it proceeds to the rectifier. The rectifier then converts the voltage to DC. In some SMPS units, additional filters are used after the rectifier to further refine the DC voltage. The filter at the input eliminates harmful voltage spikes from the sinusoidal signal, while the filter after the rectifier removes noise from the DC voltage.

Next, the voltage undergoes a crucial transformation through a switching configuration, a key element of which is the Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET)s. These MOSFETs play a vital role in converting the signal into a Pulse-Width Modulation (PWM) signal. This PWM signal alternates between high and zero voltage, effectively creating rectangular pulses. An Integrated Circuit (IC) driver then takes charge, sending pulses to the MOSFETs and controlling their on-and-off states. These pulses are transmitted through a step-down transformer, which adjusts the voltage to the desired level. After this, the voltage goes through another rectifier circuit with an output filter to clean the final signal. A part of the output is returned to the switching circuit via a feedback loop that includes a reference and error detector to correct any over-voltage issues. This signal is transmitted through an optocoupler, a component that isolates the two sides of the circuit by using an LED to transmit light to a phototransistor [115]. This isolation is crucial as it prevents any voltage spikes or noise from the output side of the circuit from affecting the input side, thereby ensuring the stability and reliability of the SMPS operation. Finally, on the other side of the optocoupler, there are protections such as overload protection and controllers that regulate the input to the transformer. This ensures that the power supply operates safely and efficiently. With the

overall design, a SMPS block diagram typically results in a similar design of Fig. 128 when a single output of DC is needed. However, SMPSs typically have some noise in the output voltage at a loss of quality.[116] [117]

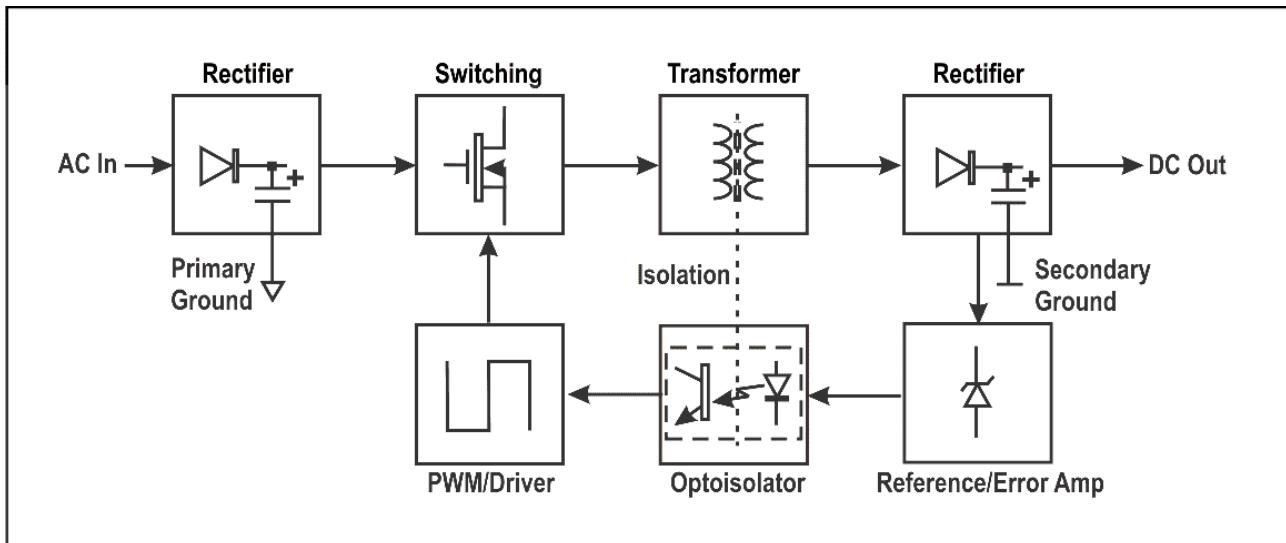


Figure 128: SMPS block diagram.[8]

### Comparison:

Comparing the two, we have one system that is quite simple and straightforward but needs excessive cooling to stay operative. It also uses voltage regulators, which have a limited current flow, which could be a problem for the more power-consuming parts of the system. On the other hand, while being more complex and incorporating more components, the SMPS significantly outperforms linear regulators in terms of efficiency and power handling, making it a superior choice for systems requiring high current flow without extensive cooling. The voltage it gives is typically more stable and has less noise. The other system is more advanced and uses more components, which adds more points of failure for the system. It does, though, do a great job at converting the voltage and still having a respectable amount of current flow. Given the importance of the current flow in our system, we have made the strategic decision to implement an SMPS. This choice aligns with our system's needs and allows for the potential addition of further filtering if required.

### Production and design:

In creating an SMPS, some factors are essential for a good design and good signal output. Factors such as the switching of the SMPS need to differ from what the output needs. Some types, such as the Buck converter, are very efficient and are used when the output needs to be lower than the input. Others, like the Boost converter, are used for the opposite purpose when the output needs to be higher than the input. A flyback converter, for example, is less efficient but can easily implement multiple outputs[118] [119]. Other factors are the placement of capacitors,

which will reduce their lifetime if they get too hot, or the thickness of the traces used on the PCB[120]. In some cases, if the thickness of the trace is too small, it can provide significantly higher noise levels. Keeping the traces as short as possible is also essential for designing the switching and loop junction. The use of a multilayered PCB is an excellent choice for an SMPS as it will not only ensure a small SMPS but also create short routes for the ground and other parts of the circuit that are impacted by long traces. Lastly, it is essential to keep a distance between the high-power and low-power sections of the circuit, as it may induce interference.

Due to all these factors, it is crucial to create a professional PCB and test the product to ensure the assembly is of the utmost quality. Given the narrow timeline, lack of electrical engineers on our team, and a PCB delivery time of approximately a month, we have carefully considered our options and decided to use an already existing SMPS called the RD-85A. This SMPS was chosen because it delivers 8 amperes at 5 volts with only  $80 V_{rp-p}$  and 4 amperes at 12 volts with  $120 V_{rp-p}$  [9], which perfectly aligns with our system's requirements. By choosing this SMPS we will have the block diagram shown in Fig. 129 and can ensure our system's successful implementation a power supply.

#### ■ Block Diagram

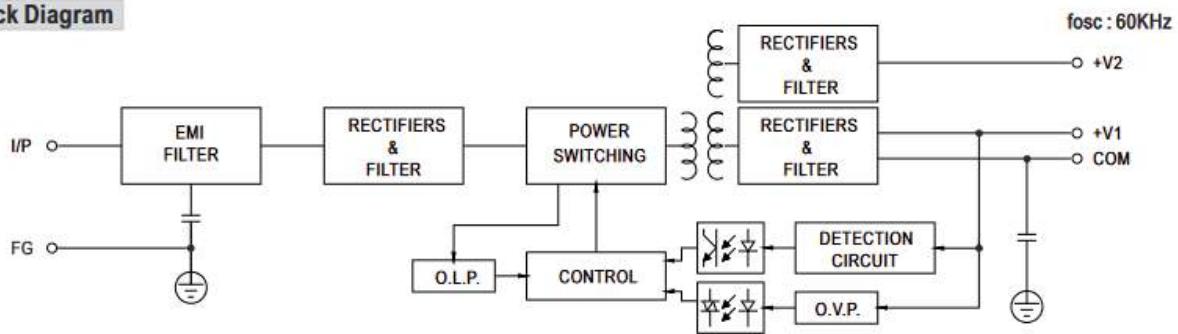
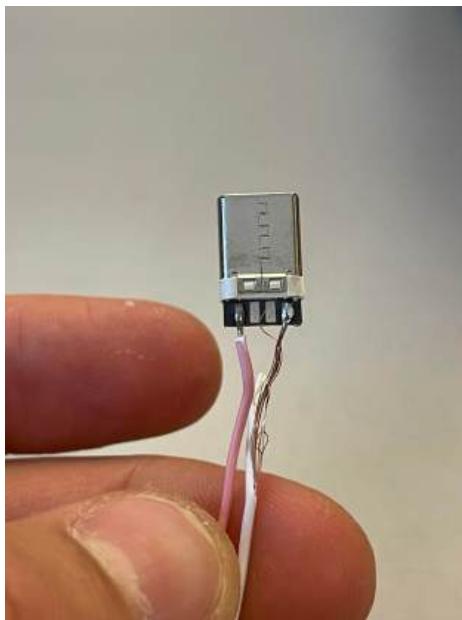


Figure 129: Block diagram for the RD-85A power supply. [9]

#### Challenges/setting up the system/powering alternatives/Testing phase

**Testing and assembling:** When assembling the electric system of the Senior, unexpected errors appeared. We cut the wire for a USB-C cable and connected it to the SMPS before connecting it to a Samsung tablet, but it would not charge. To confirm that the charger was the problem, we used a functioning iPhone charger for the same purpose, but it would still not charge. We, therefore, had our hypothesis of the causes that may be. The SMPS does have noise in the DC power, which may be the cause, as most phones and tablets do not want to charge with too noisy power. The reason behind this is its consequence of often setting batteries on fire if the power is too noisy. Another possibility is the wires low quality due to it being a free advertisement charger given at a stand. We, therefore, changed the wires to higher quality wires, but still nothing. Our last hypothesis was that the USB-C port was low-quality and may have fried when given a higher voltage than it is supposed to handle. To test this, we measured the conductivity from the power supply to the wire input to the USB-C port, which worked

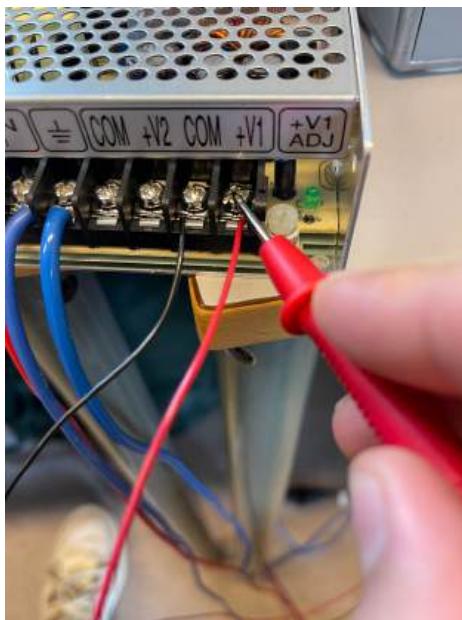
fine, but when the pin within the port responsible for power was tested, we had no response. Since the power source of the Raspberry Pi typically has a  $V_{rp-p}$  of 120  $V_{rp-p}$ , and our source is 80  $V_{rp-p}$ , the SMPS should be considered a safe source. To further confirm the theory, we needed to debunk our first hypothesis that noise was the main problem for the tablet, but due to the software team using the Raspberry Pi, we could not test it. [121]



(a) The original wires for the USB-C charger. Pink being voltage and white being ground.



(b) The new wires with red being voltage and black being ground.



(c) Testing the conductivity from power supply.



(d) The receive point from the conductivity test.

Figure 130: The controller design broken up. The red wires are 5 V, the black wires are ground, and the purple wires are for communication.

### 6.2.2 Control unit

LTR | AL

#### First iteration:

To control the system, we have designed a controller that communicates with the Raspberry Pi to operate the actuators. It features two buttons and a joystick, enabling precise control over the robot's movements.

When moved, the joystick offers precise manual control, transmitting analog signals to the Raspberry Pi. The Raspberry Pi then utilizes these signals to instruct movements between the two actuators. Pressing the joystick button activates various user control functionalities. One button demonstrates a predefined path, while the other is a software stop button to halt a task early if needed. To communicate with the Raspberry Pi, we have chosen to use an Arduino. Lastly, with all these requirements in mind, we constructed a design corresponding to Fig. 131.

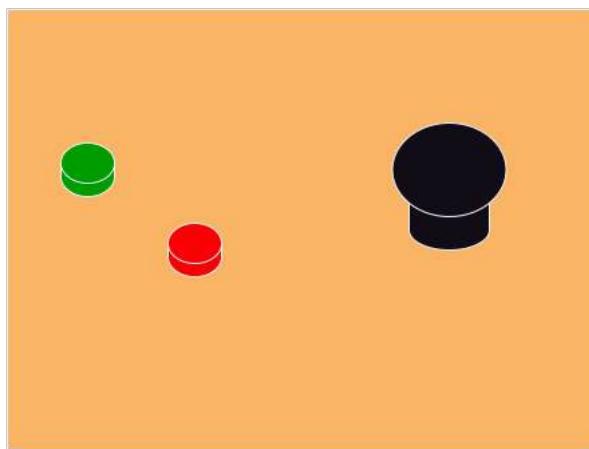


Figure 131: First iteration control design

After constructing the controller, we opted to not use a breadboard as it is unreliable, thick, and nonmodular, resulting in a bulky and overly complex design. A PCB design will bring more freedom but will be relatively more expensive. We have therefore opted for something in between, precisely a prototype PCB board, shown in Fig. 132. This board is designed with copper traces, which you use Through Hole (TH) components to work as a connection between wires you solder between the traces. If a trace connects with other components, which it is not supposed to, the trace can be scraped away.

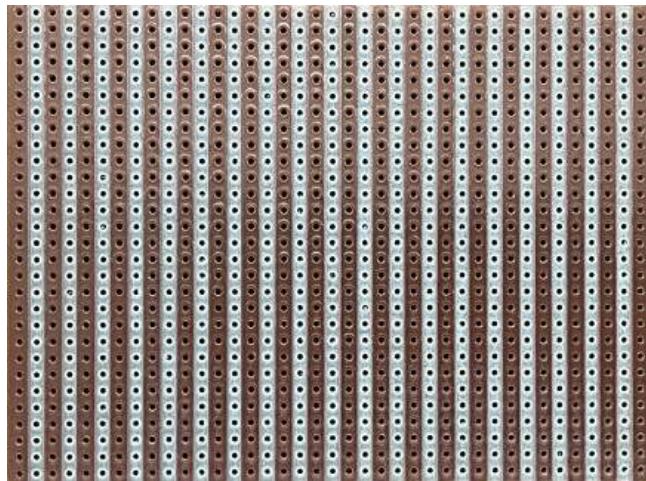


Figure 132: Prototype PCB board

When designing the systems buttons, we have implemented a hardware debounce mechanism, ensuring a highly reactive system that responds instantaneously with minimal system delays [122]. This works by using a low-value capacitor between the ground and the output pin, filling the ripple gaps to achieve a level that reads as constant HIGH. The need for this debounce system is due to the switch bounce between high and low when the button is pressed. The ripple causes the system to read multiple highs and lows as if the button is pressed multiple times, which may cause software problems. We therefore use 1 uF capacitors between the output and the ground to fill the gaps in between. 10 k $\Omega$  resistors are also used to ground the buttons to increase the stability of the logic level.

With the buttons in place and the joystick connected to the common ground and 5 V, we ended up with the design in Fig. 133. However, before further connecting the communication cables to mount it with the Arduino, we found that an Arduino Nano would be a better fit for this type of design as it can be mounted on top of the device instead of the button. This change alone will almost halve the size of the controller, resulting in a sleek and attractive design. We've also added another button, as the software now requires a button to configure the mapping sequence from the user interface. These changes are set to enhance the system's functionality and aesthetics. With these changes in mind, the new model in Fig. 134 has been constructed.

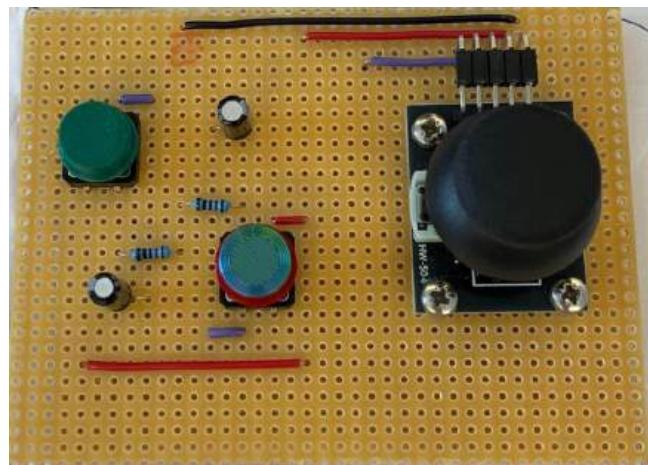


Figure 133: First iteration prototype controller

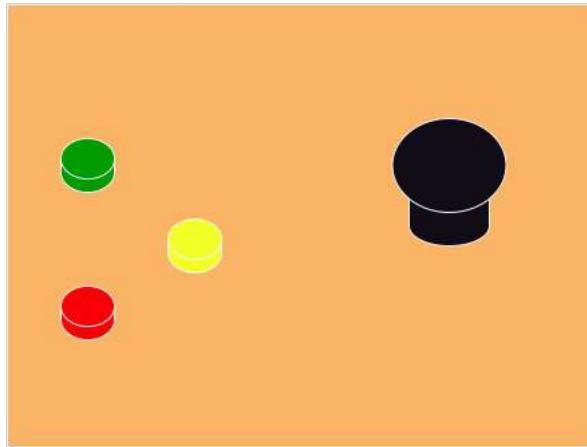
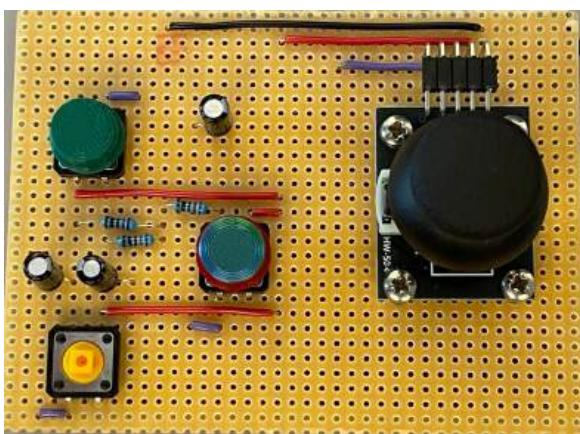


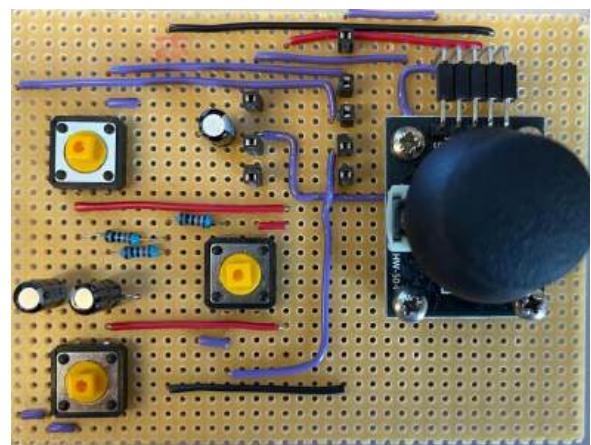
Figure 134: Second iteration control design.

### Second iteration:

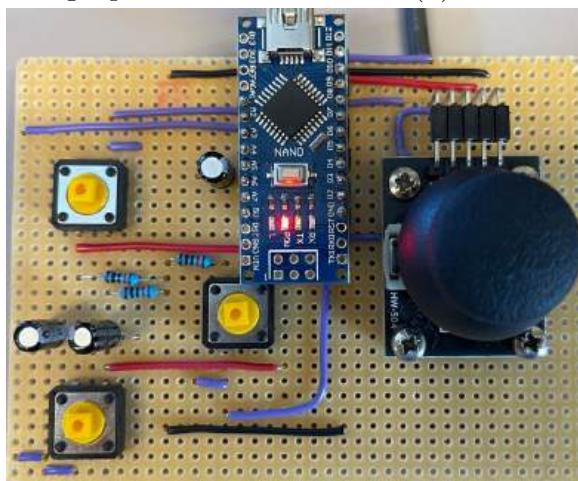
When implementing the new design, some changes were made in the paths of the controller, but to maintain the integrity of the design, as few changes as possible were made. This includes some wire movements and additional component placement for the buttons Fig. 135a. To satisfy the placement of the Arduino Nano's design, we used male to female headers and soldered only the specific pins used by the Arduino for a perfect placement every time Fig. 135b. Lastly, the Arduino Nano is placed on top with 5V being the bottom left pin and the ground being the bottom right pin in Fig. 135b to achieve the second design Fig. 135c provides a visualization. For a deeper understanding of the control system and its paths, visit App. M.



(a) The controller without logic paths.



(b) The controller with logical paths.



(c) The second controller design.

Figure 135: The controller design broken up. The red wires are 5 V, the black wires are ground, and the purple wires are for communication.

With the foolproof design finished, or so we thought, the controller was sent to the software development phase, where problems were encountered. The Arduino Nano we ordered turned out to be a cheap knockoff of the original Nano. The components it used were not up to the task and proved not to be reliable at all. When borrowing an original Nano, it did not live up to the task either, as the controller would not deliver the positions to the motors, which was weird due to the testing working perfectly on the Arduino Mega. There is a possibility that the Nano we borrowed had previous problems, which made it perform so badly, and that we could order another one, but due to creeping towards us, the Nano would not arrive in time. We therefore decided to use the Mega for the controller as well, using the modular design by connecting wires between the Mega and the controller, achieving a bulkier design but a much more reliable one. We looked into two options for redesigning the controller. Having the Mega under the controller or in front of it has different advantages for both designs. A controller needs to fit comfortably. Therefore, we decided to have it in front because the type A/B connector was further from the hands and gave a slimmer design rather than the blocky design it could have had. This resulted in the final design for the controller, Fig. 136.

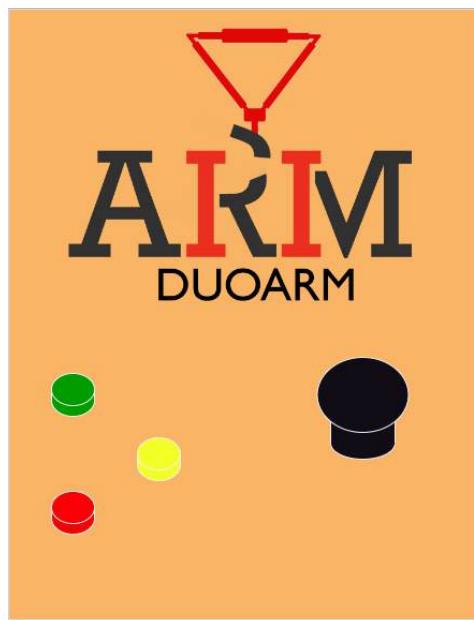


Figure 136: Controller design using the Arduino Mega.

### 6.2.3 Led light circuit design

**LTR | AL**

#### Construction:

With its aesthetically pleasing yet simple design, the LED light system serves a valuable purpose. It provides an intuitive way for the user to know which functions are active. The system's design incorporates P215J N-type MOSFETs [123],  $10\text{ k}\Omega$  resistors, a led strip using SMD 5050 LEDs, an external power supply, and an Arduino. Three MOSFETs are responsible for each color of the RGB, connecting the drain to a PWM pin on the Arduino, the gate to the respective color of the light strip, and the source to the common ground [124]. This setup shown in Fig. 137 allows us to send an analog signal to the drain between 0 and 255, precisely adjusting the color intensity using the MOSFETs as a sort of tap to increase the current flow to each RGB LED.

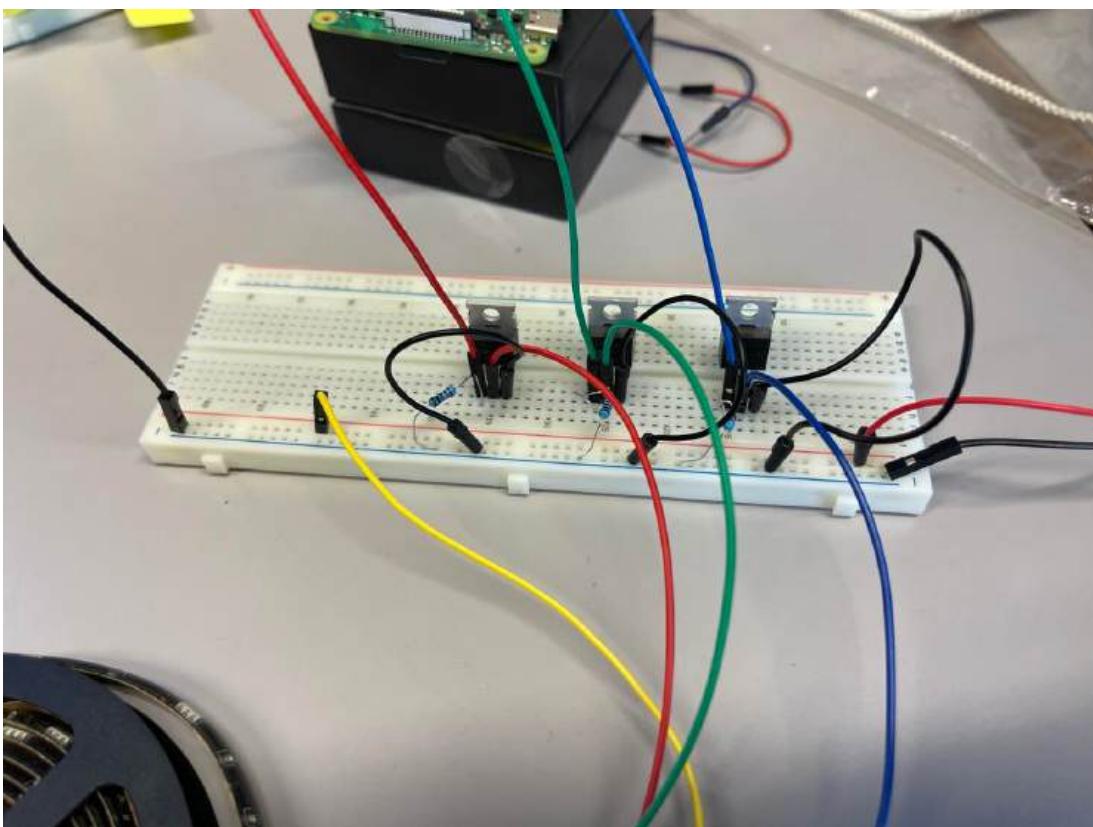


Figure 137: LED circuit using MOSFET for current adjustment for the LEDs using an external power source.

### Testing:

A part was cut when testing the LED strip to ensure that circuit shortings would not affect the whole strip. Starting by using the same button configuration for the controller to activate the red light. This was done by giving the rBright variable a value of 255 when the button is pressed and analog writing it for the MOSFET. With this test successful, functions for each color were made, and another button was added to act as the second button of the system, resulting in the first iteration of arduino code App. M. Though this solution seemed bulletproof, we encountered a problem. The LED lights were not acting correctly, with the green light emitting a tiny speck of light at all times. Due to the only difference in the system being the software, changes were made, but nothing seemed to work. The code was, therefore, rewritten from scratch, but this time, the red light seemed to operate in the same way as the green light, with only the blue light acting correctly. The hardware was then reconstructed to ensure the breadboard was not the problem. When reassembling the circuit, the MOSFETs had changed places, causing the green light to be the only LED emitting light, proving the MOSFETs to be the problem. This was initially not seen as the possible cause of the problem due to its robustness compared to the low power going through this circuit [123]. They were then switched to new ones, and the circuit worked perfectly.

We now had a successful interface system draft for the software team to integrate into ROS. The only area for improvement is its lack of aesthetic appearance. We found that a dimming and

brightening solution would significantly enhance the systems appearance, and a new iteration was begun. This means that the new functions dimLight and brightenLights have to be implemented in the system. We, therefore, opted to keep the xLight functions but send the variables to the functions from there. This way, we can make a general function that can change the intensity of the light with a variable to adjust the speed of the transitions by determining how quickly the lights shift. However, when returning the brightness value, the xBright would not be updated. To solve this, we manually changed the values in the xLight function. With these changes, our system could now perfectly dim between the colors depending on the different button presses, with minimal changes required to implement the logic to the system.

#### 6.2.4 System interaction

LTR | TM

Our system is designed to be user-friendly and flexible. Therefore, we have opted to use the C13 power cable with a C14 connector to supply the system with power, as it allows the power cable to be removed, enhancing the system's mobility. On the inside of the system, from the C14 connector, the cable goes to an emergency stop button, giving the user full control over the power supply, as it can be used as both an emergency stop and on/off function. From the emergency stop, the cable goes to the SMPS, which has internal EMI filtration as well as short circuit, overload, and over-voltage protections. This process of stepping down the voltage to 5 and 12 V ensures that no harmful power enters any sensitive components without the need for an external fuse. An XT60H power cable is then connected to the 12 V to supply the LSS-ADA board, which then distributes the 12 V to the three HS1 smart servos. The 5 V port supplies power to multiple components in the system, including the Raspberry PI and the LED lights. A couple of Arduino Megas are connected to the Raspberry PI USB ports. One is connected to the LED configuration, and the other is connected to the controller through a USB-a male-to-female extension to the front of the model. The extension's purpose is to have a short distance to the port from the front, allowing the user to remove the connection cord entirely and hide the controller in the electrical cabinet. Lastly, a third USB connection is made with a USB-C cable for bidirectional communication between the Raspberry PI and the LSS-ADA. All of these components are either locked in place in the electrical cabinet or can be placed within it, with the only exceptions being the USB a male-to-female cable, and LED lights, which are carried within the model, resulting in the design in Fig. 138.

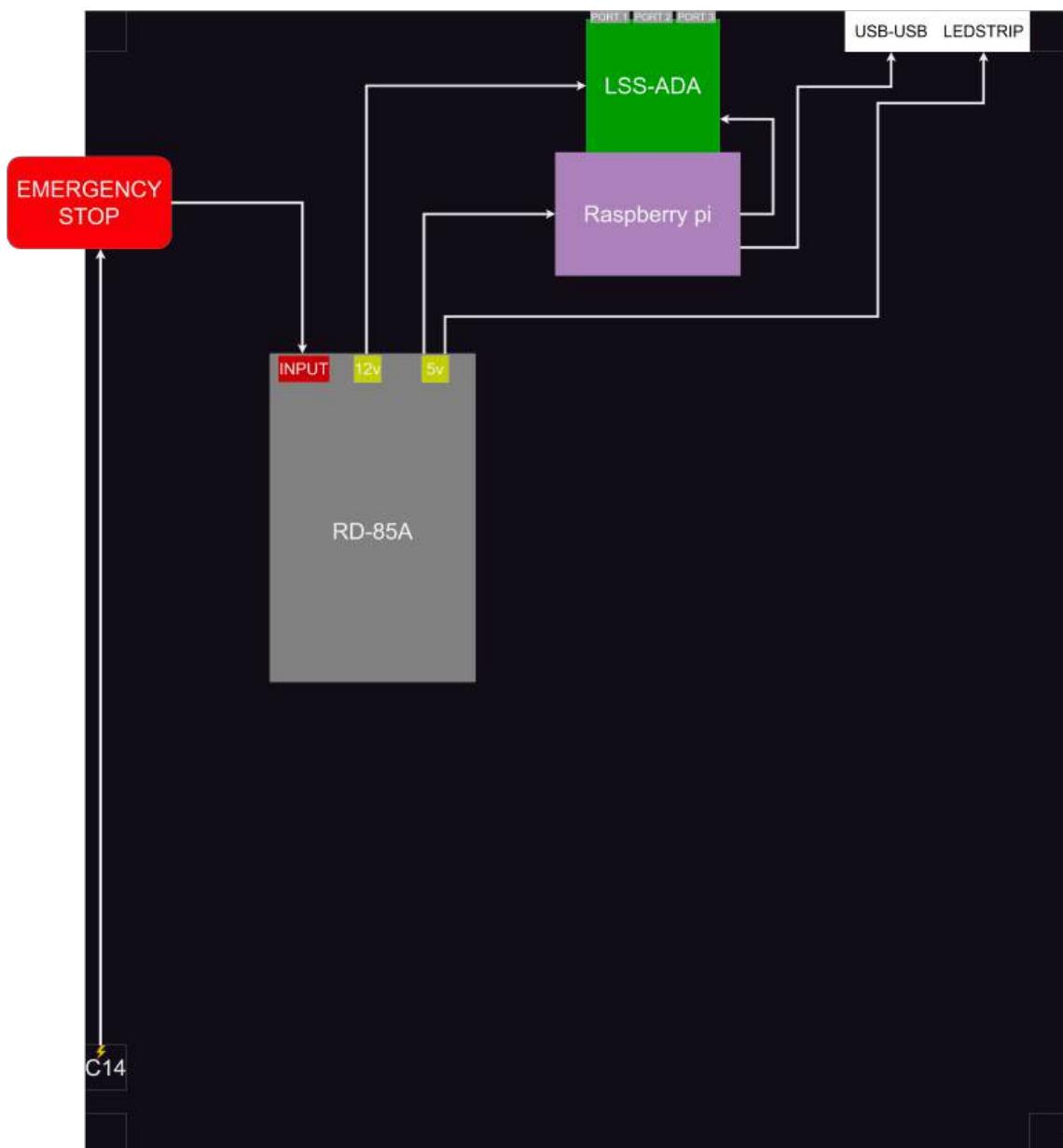


Figure 138: The electrical cabinet layout.

### 6.2.5 Conclusion and future improvements

**LTR | TM**

To secure the last parts of the system, tests on the Raspberry Pi and LSS-ADA board are needed. If there are problems with the USB-C power provider, another solution is simply powering the Raspberry Pi through the 5 V pins. The LSS-ADA also needs to be connected to the power supply, but from the 12V, and will need a new XT60H-F power cable that can be connected to the adapter board. This is possible as both the pins, together, secure enough amperage to the Raspberry Pi to power up and work securely. The risk comes in the form of a lost fuse, as the Raspberry Pi has integrated protection from the input if the wrong power supply is used. This should only be a form of last resort as, in the worst of cases, we may fry the Raspberry Pi if not done correctly. However, I suspect the power supply will work as the noise levels are lower than the recommended charger, making it a more stable power source.

The same applies to the LSS-ADA board, of which the power supply noise is over  $200 \text{ mV}_{rp-p}$ , but our 12 V supply only has  $120 \text{ mV}_{rp-p}$ , which is a massive improvement from the standard power supply.

### 6.3 The software development process

#### 6.3.1 Introduction

TM | AL

When developing a system, we start by looking at the requirements, this is essential as the requirements define what the system shall be able to do. These are the A requirements relevant for software:

- R1.0 - The prototype shall be able to perform a vertical and horizontal "pick and place" movement in the three coordinate axis-directions (x, y and z).
- R2.0 - The prototype must have an incorporated rail system to account for movement on the y-axis.
- R3.0 -The prototype's rail system must have movement capabilities up to 10 cm.
- R7.0 - The prototype must be user-friendly.
- R8.0 - The prototype must be able to be moved and used for exhibition.
- R9.0 - The prototype shall have a manual control system.

To derive functionality and concrete logic from these requirements we took use of UML, starting with making a use case diagram with these requirements in mind. The use case depicts how the user can interact with the system. From the use case we can derive activity diagrams, which depict the flow of actions and logic in the system. Activity diagrams support choice, looping and concurrency. From the activity diagrams we derived sequence diagrams which depicts how specific components in a system interact with each other, displaying the messages send back-and-forth between components. Generally, these diagrams were made before, during and after development.

#### 6.3.2 Top level architecture

TM, AL | LTR

**Developing use cases: AL | LTR, TM**

Developing a UML use case diagram is a beneficial step in every software modeling process. It not only helps communicate the system's overall functionalities to the stakeholder(s) and illustrates how an actor would interact with your system but also ensures that every developer of the project team is onboard with the system's area of use [125]. Additionally, it acts as a solid foundation to develop lower-level architecture, which provides a more detailed view of specific system aspects.

Besides catering to the overlying functional and non-functional requirements, we were given

free rein to develop specific actor actions or system functionalities. With this in mind, we designed and developed a software system that addressed the fundamental requirements while promoting simplicity and interactivity for users at TE and exhibition participants. The system's use cases or actor actions are visually depicted in Fig. 139 and can be listed as follows:

- Run a predefined path
- Stop the active process
- Initiate a mapping sequence
- Control the system with a joystick
- Activate the rail movement
- Shut down the system

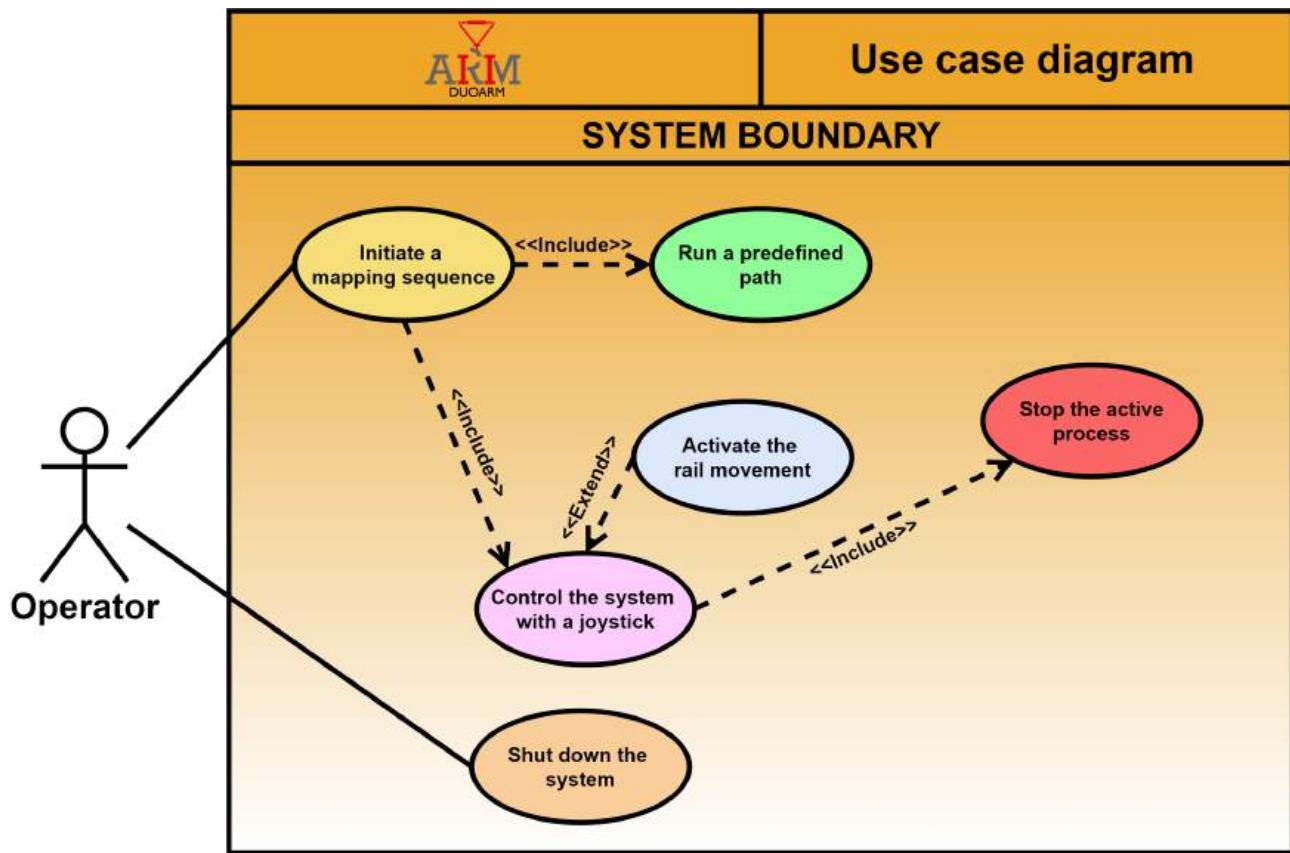


Figure 139: Use case diagram - Operator

Besides the use case: "Shut down the system," every use case was color-coded to adhere to the five operation modes or states of the DuoArm system and create a red thread when illustrating the lower-level architecture. The states or operation modes of the system can be viewed in the generic figure Fig. 140.

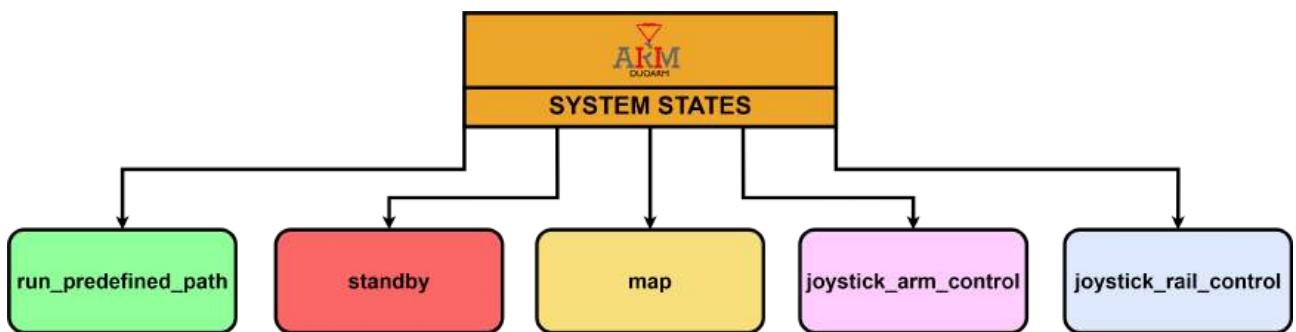


Figure 140: System states - Overview

In regards to the use case diagram in Fig. 139, the system flow and dependency relationship among the different use cases and the actor depict a system launch where there is **no** valid mapping data already stored. The dependencies and general system flow during this scenario can be described as follows:

- At initial system launch, the operator has one main associative user action, not considering the "shut down the system" action he or she must perform to receive the full system experience.
  - This is an initiation of a mapping sequence, which has to be conducted to control the system with a joystick in either the *map* or the *joystick\_arm\_control* state, or run a predefined path in the *run\_predefined\_path* state. Hence, the "include" dependencies between the use cases.
- After completing a mapping sequence the system is set to *standby* and the operator stands free to set the system in the *joystick\_arm\_control*, *run\_predefined\_path*, or back to the *map* state to run another mapping sequence.
  - From the *joystick\_arm\_control* state the operator can choose to activate the rail system. Hence, the "extend" dependency. However, to stop the active process, and set the system back to the *standby* state, it first has to be in the *joystick\_arm\_control* state. Hence, the "include" dependency.

After an initial run and a reboot of the system, valid mapping data from the last run will be loaded, and the operator stands free to set the system in the *joystick\_arm\_control*, *run\_predefined\_path* or *map* state from the beginning.

**System overview: TM | AL****System architecture and design principle:**

The generic software overview architecture represents the system's structure from the perspective of a high abstraction level, its defined by how components are connected to each other through their respective interface. When building software there are several design principles that can be incorporated for the system architecture to be organized, reliable and scalable. During development there was specifically one principle that we tried to incorporate into the design. This principle is not defined by ROS but rather by what we have found online whilst doing research on ROS and software architectural design. This principle is Modularity through encapsulation.

Encapsulation is a principle where each component is responsible for specific functionality and manages its own behaviors and states. Encapsulation establishes modularity, making the system more robust as errors in one component will have minimal impact on the rest of the system. This principle encourages reusability and simplifies maintenance, as components can be switched out and errors become easier to track.[126]

**Our architecture**

In a ROS system, a component is called a node, and these nodes are placed in packages, allowing developers to group functionality together in a meaningful way that makes development and maintenance easier. In our system we have three packages, each containing nodes grouped together after their uses: Firstly, we have the hardware\_center package, which is responsible for interacting with hardware and includes drivers for servos. Secondly, we have the control\_center package which is responsible for state management, executing commands and provide informative logger messages. Lastly, we have the map\_and\_path\_center package, which contain nodes for mapping and path runs. In the diagram below, we can see our ROS project with its packages, nodes, and the communication between them. The diagram also describes hardware like servo motors, micro-controllers and their communication with ROS.

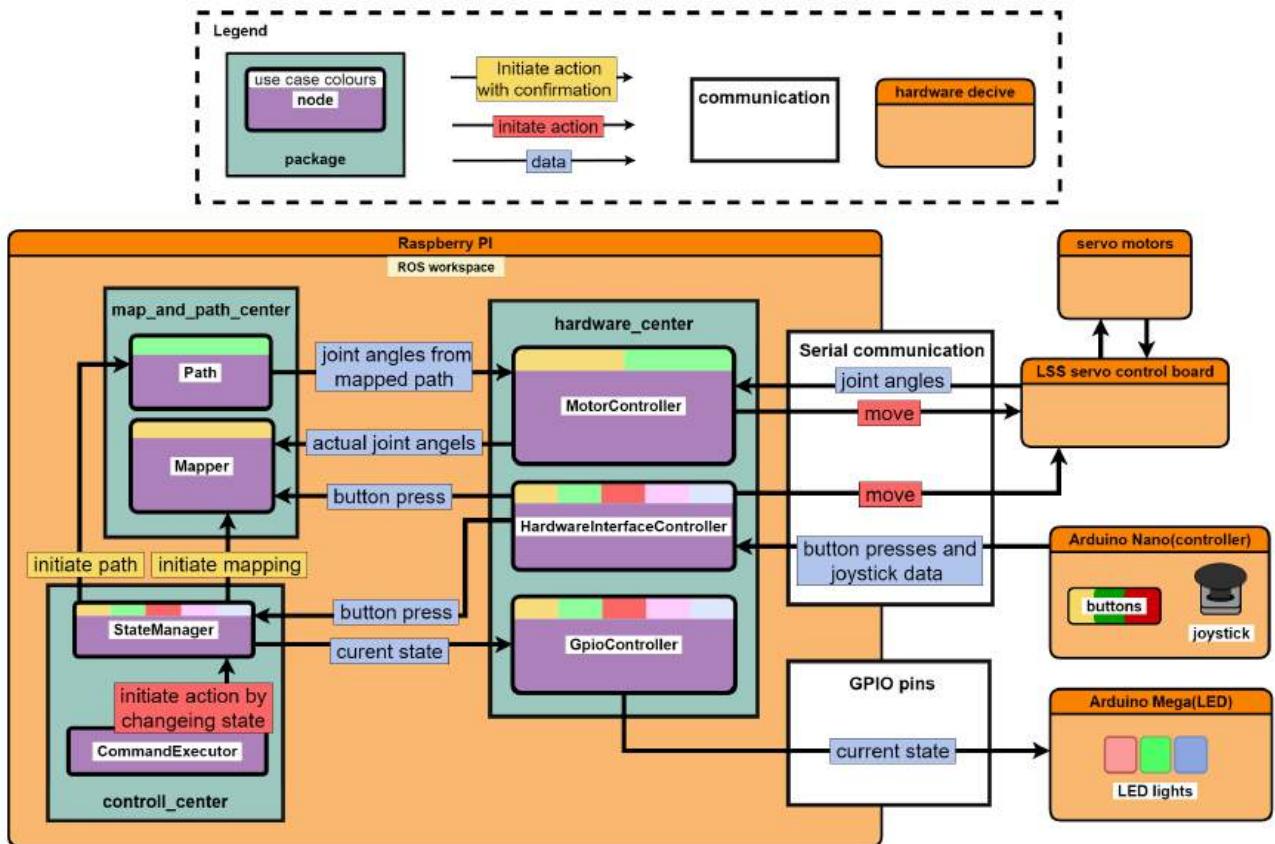


Figure 141: Software architecture diagram

### Configuring use case events: AL | TM

In an attempt to depict every event or pre-conditions associated with the execution of each use case, an overarching use case table was created. The table adds unique identifiers to each use case and their belonging events, and should serve as an overview for the reader, enabling them to easily branch and trace different use cases and their respective events through the lower-level architecture. The color coding of the various events indicates the use case they are connected to, and in the instances where colors are a mix of two, it indicates that the events are connected to more than one use case. The use cases along with their respective events can be viewed in Tab. 16.

Table 16: Overarching use case table

Use case table				
Use case		Event		
Identification (ID)	Description	ID	Description	
UC1	Run predefined path	UC1-E1	An operator presses the "Run predefined path" button	
UC2	Stop the active process	UC2-E1	An operator presses the reset/"Stop the active process" button when the system state is in any of the other states beside the joystick_arm_control	
		UC2-E2	An operator presses the reset/"Stop the active process" button when the system state is equal to joystick_arm_control	
UC3	Initiate a mapping sequence	UC3-E1	An operator presses the map button when the system state is equal to 'standby'	
		UC3-E2	An operator presses the map button when the system state is equal to 'map'	
		UC3&4-E1	An operator moves the joystick towards the north or northeast	
		UC3&4-E2	An operator moves the joystick towards the east or southeast	
		UC3&4-E3	An operator moves the joystick towards the south or southwest	
UC4	Control the system with a joystick	UC3&4-E4	An operator moves the joystick towards the west or northwest	
		UC3&4-E5	An operator pauses the movement of the joystick	
UC5	Activate the rail movement	UC4&5-E1	An operator presses the incorporated button of the joystick	
UC6	Shut down the system	UC6-E1	An operator presses the emergency stop button	Comment This operation is solely performed through electronic circuit design, hence the lack of accompanying UML diagrams

### 6.3.3 Configuring communication and data transmission between the hardware components

AL | TM

#### Introduction:

Based on the fact that the final iteration of the DuoArm system operates with several components: a controller with a microcontroller, joystick and four buttons, including the incorporated button of the joystick, two LSS-HS1 motors, an LSS adapter board, a single board computer, and an additional microcontrollers connected to LED lights, it became vital to ensure sufficient communication between the different components, without overcomplicating the mode of operation. Developing the communication and data transmissions has been a thorough process and multiple iterations were required to finalize the end result.

#### Communication between Raspberry Pi, the LSS adapter board and LSS motors:

To ensure communication between the single board computer or Raspberry PI, the LSS adapter board, and the LSS-HS1 motors, the official LSS libraries for python were utilized. These libraries not only set up the communication, but also provide classes and member functions to control and activate each LSS-HS1 motor. The only requirement was to create objects of the class LSS with a unique identification number or digit associated with each LSS-HS1 motor, and assign the correct USB port between the LSS adapter board and the Raspberry Pi to a port constant of the LSS global variables. To assign an identification number or digit to each servo the official LSS configuration software were utilized. This process was performed once, as the unique identification was stored in EEPROM. EEPROM facilitates data retention between power-offs and reboots.

#### Communication between Raspberry Pi and the user interface controller:

In regards to the communication between the Raspberry Pi and the microcontroller, which handled inputs from the controller, we implemented the pySerial module in our software. The pySerial module encapsulates access to serial ports [127], and through it, we were able to create an instance of the module that facilitated transmission between the two components. The instance was defined correctly by passing the corresponding serial port and baudrate of the transmitting Arduino as parameters in the constructor method.

During the configuration of data transmissions between the Arduino controlling the user inputs and the Raspberry Pi, two principles were specifically considered. The first principle was that a button press, whether an instant press or a long hold, should only be registered as a single high signal. This was crucial to prevent the main software from executing functionality in response to

two or more repeated button presses, when the user intended for only one. The second principle focused on providing real-time system feedback to the user regarding movement of the joystick and button presses. The former proved a challenge as we, unknowingly, were operating with defect hardware components. After this problem was sorted, we encountered another issue with the data being forwarded to the Raspberry Pi. Initially, we opted to transmit an array of six integers, totaling 12 bytes, using the `Serial.write()` method. However, due to synchronization between the microcontroller and the Raspberry Pi - presumably after the ROS was shutdown and rebooted while the microcontroller had continued to run - it resulted in the Raspberry Pi misinterpreting the data due to a discrepancy in byte ordering. This led to the assignment of incorrect values to variables or, in some cases, system crashes as a result of attempting to read invalid start bytes.

The final solution implemented a combination of `print()` - functionalities from the `Serial` library, while employing the `millis()` function to create non-blocking delays and control the rate of data transmissions and readings in accordance with the second principle. Additionally, the button inputs were also read on their rising edge, except for the joystick, which registers presses on the falling edge, to adhere to the first principle. The final iteration of the transmission software in Arduino can be viewed in App. N, under section: "Code documentation", subsection: "Arduino software" and subsubsection: "Transmission software for the Arduino handling the controller". On the receiving component, the Raspberry Pi, the byte values printed to the serial port from the Arduino are read until a newline character appears. They are then decoded and stripped at `","`, and stored in a temporary array, before the bytes at the various indexes are converted back to integers and assigned to their corresponding variables in the `hardware_interface_controller` node for further handling. At the launch off every ROS node, the input buffer of the serial instance are also reset to mitigate the possibility of encountering system readings of invalid start bytes.

### **Implementing communication for the LED lights:**

In regard to the LED lights, they were implemented as an alternative or additional element to a visual display and, by resembling the colors associated with the active state of the DuoArm system, they serve as an additional or sole level of guidance for users throughout the system's operation. This implementation process also demanded thorough consideration. We initially chose to use the `PiGPIO` library to directly control the signal sent from the Raspberry Pi's GPIO pins to the LEDs. However, despite defining the correct GPIO output pins, the resulting brightness was severely dimmed. Consequently, we experienced with PWM signals, intending to provide duty cycles between 0 - 100 to control the brightness. Unfortunately, this approach also failed to produce the desired outcome, so we decided to use an additional Arduino Mega to power the LEDs, as this had proven successful in earlier testing.

At the time being, we now had two Arduino Megas connected, resulting in the opening of two ACM ports. To differentiate between them, we initially attempted to capture every ACM port and forward an ID query to each one. However, we encountered a setback when we discovered that ROS blocks the execution of the serial.write() - functionality. This led us to consider rosserial, a protocol that enables forwarding of ROS messages through serial ports. Unfortunately, we quickly realized that this is not supported in ROS2. As a potential alternative, we explored the possibility of establishing Micro-ROS on the microcontrollers. However, we discovered that neither an Arduino Mega or Nano are compatible with this solution. As a final alternative we opted to utilized the built-in GPIO pins to establish signal transmission from the single board computer to the Arduino. This proved successful.

The final solution of the data transmissions between the LEDs, the other Arduino microcontroller and the Raspberry Pi, where realized utilizing the ROS node: *GpioController*, which reacts to system state updates on the topic: "system\_state", by triggering a callback function. This method configures the parameters for the set\_led\_state function, which outputs digital signals to three specific digital pins of the Arduino Mega. Different led states are then determined based on the combination of these input signals, and the resulting combination of led states assigns a specific number to a variable that is evaluated in a switch case. "When a case statement is found whose value matches that of the variable" ([128]), analog values are written to the correct output pins of the Arduino, which are connected to the lights. For the final iteration of the receiver software for the Arduino controlling the LEDs see App. N, section: "Code documentation", subsection: "Arduino software" and the sub-subsection: "Receiver software for the Arduino controlling the LED lights". This software properly activates the LEDs according to the desired outcome.

### **Additional challenges:**

During the process of establishing the communication and data transmission between the different components, we also faced other challenges, which required adjustments. At first, we opted for an Arduino Mega to relay information between the controller and Raspberry Pi. But we decided to swap it for an Arduino Nano, which was, shortly after, discovered to be a clone of the original microcontroller. Clones of Arduino products are more likely to have internal issues and induce communication problems, because of the utilization of substitute parts to make the boards more affordable [129]. Our challenge was rooted in abrupt and unreliable communication halts, where the port connection between the Raspberry Pi and the Nano would unexpectedly shut down and freeze the user input flow. This occurred even though we had the drivers installed and was verified, through software debug messages, that the correct user input values were initially being forwarded over the communication port. To troubleshoot the challenge, we carried out several actions. Firstly, we made sure that the freeze was not caused by errors in other code blocks within the software system. For this, the Arduino Nano was

temporarily replaced by a Mega and a full system run was initiated and successfully completed. This quickly disproved the theory. Secondly, we ensured that the correct serial communication drivers were installed and updated, and that the appropriate board settings and boot-loader were configured before re-uploading the sketch to the Nano from the Arduino IDE. However, as the problem was not rooted here either, we tried to replace the clone for an original Nano. Our hope was that it would bypass the issues related to the cheap substitution parts. Unfortunately, this component proved to be defect, so we had to rely on an approach that changed the planned design of the controller: Simply, pull wires from the female connection pins to the appropriate inputs on an Arduino Mega to establish efficient communication between the controller and the Raspberry Pi. This approach, naturally, leads to a more hefty controller design, but in return we will have greater reliability in the user input communication.

Arduino Mega microcontrollers usually create serial ports using the ACM protocol, which can vary in numbering depending on the rate at which a device is removed and reconnected. Since we were operating with two ACM ports, we demanded a reliable way of differentiating the port associated with the microcontroller controlling the LEDs from the port associated with the Arduino handling the joystick and button inputs. Despite multiple attempts to make this process dynamic, we had to settle on a more pragmatic approach, at least for the time being: Simply connect each device one by one and manually assign the correct port to the serial object of the Arduino controlling the joystick and button inputs and the port variable associated with the LSS adapter board. This, preferably, temporary compromise was made due to time constraints. More information about the attempt to make the port assignment dynamic can be read under the sub-subsection: "Ensuring the system is exhibition ready". See App. N, section: "Code documentation" and subsection: "Documentation generated by Doxygen" for documentation of the *hardware\_interface\_controller* and the *State\_Manager* node.

#### 6.3.4 Implementation of use case: "Initiate a mapping sequence"

TM | AL

##### What is the mapping sequence?

The mapping sequence in our robotic system is placed in its own node and is designed to serve several functions, including setting null points for the servo motors, as it establishes a consistency between the physical and virtual arm. The mapping sequence also includes setting boundary angles that play a crucial role during joystick movement. Additionally, the mapping sequence includes the recording of angles that make up a predefined path, which the robot arm can follow in subsequent operation. Executing a predefined path is one of the requirements for our system, for this requirement to be fulfilled we must first map a path.

##### How does it work?

By pressing the map button, the operator initiates the mapping sequence, but only when no other sequence is running. As the button is pressed the *Hardware\_Interface\_Controller* node sends a request to the *StateManager*, asking for a state change. This request is only approved if the current system state is *standby*. When the mapping node receives the *map* state, the sequence initiates. It starts with sending a message to the *MotorController* node, initiating a function that sets null points and gives the operator instructions on what to do next. Through the command line they are prompted to move the arm to the highest reachable point using the joystick, then press the map button again. When the button is pressed, the *Mapper* node sends a request to the *MotorController*, asking for the current angles read by the servos. The angles are saved in the *Mapper* node as boundaries and the operator is prompted to move to the next set of boundaries and press the button. The *Mapper* and *MotorController* nodes work tightly together as this back-and-forth communication will occur at each button press. The operator must map two sets of boundaries and three points, and when all points are mapped the data is saved to a JSON file, and a state change request is sent to *StateManager* for the system state to be set back to *standby*.

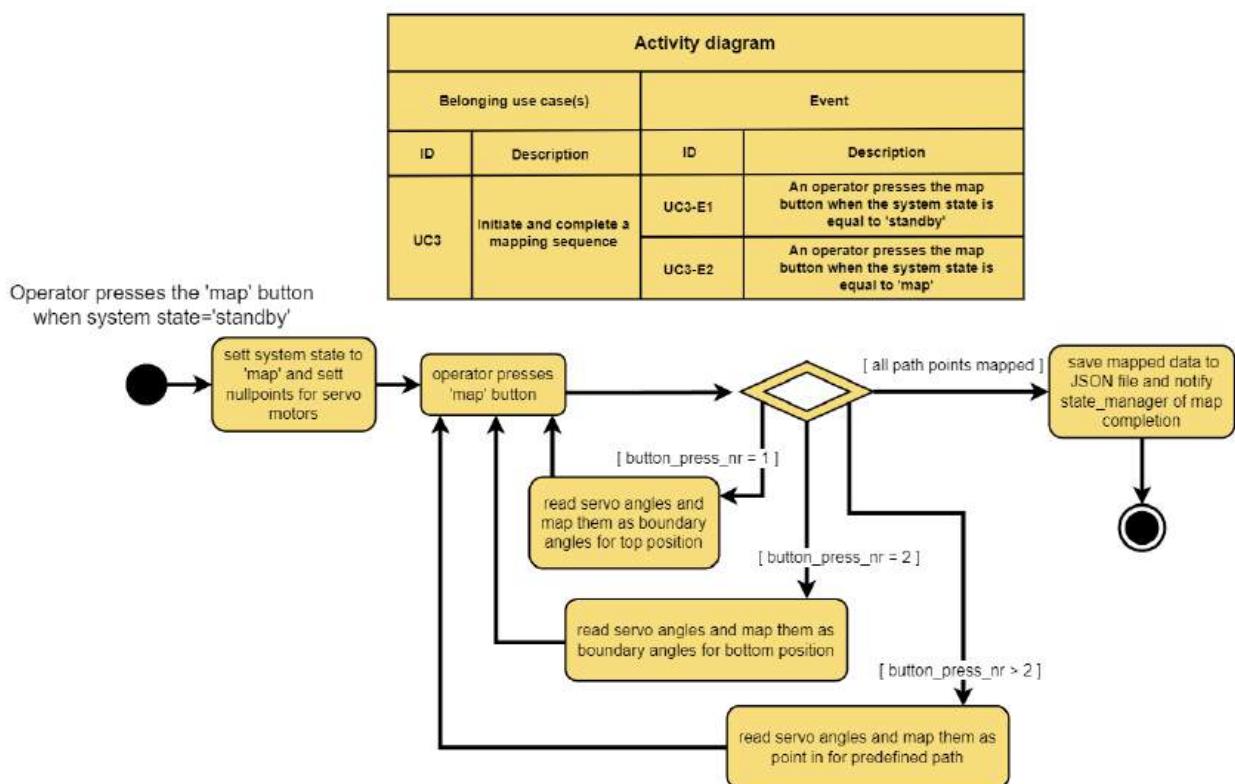


Figure 142: Activity diagram describing how the operator interacts with the system during the mapping sequence.

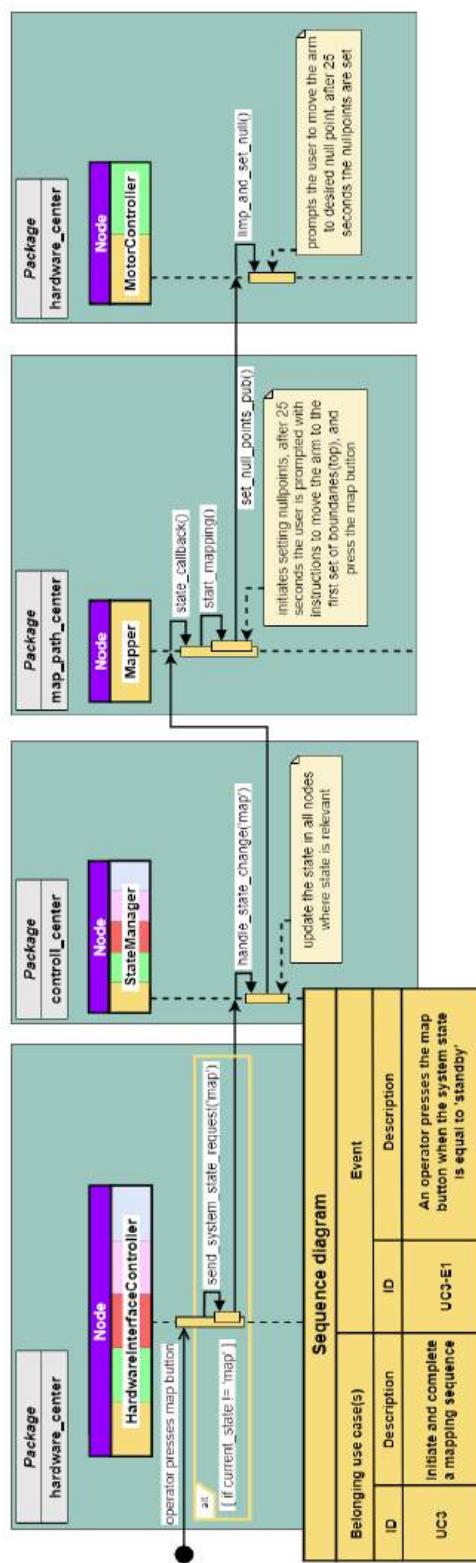


Figure 143: Sequence diagram describing how the mapping sequence works from a lower level. Here we can see each node in operation during the sequence and the messages and function calls between them.

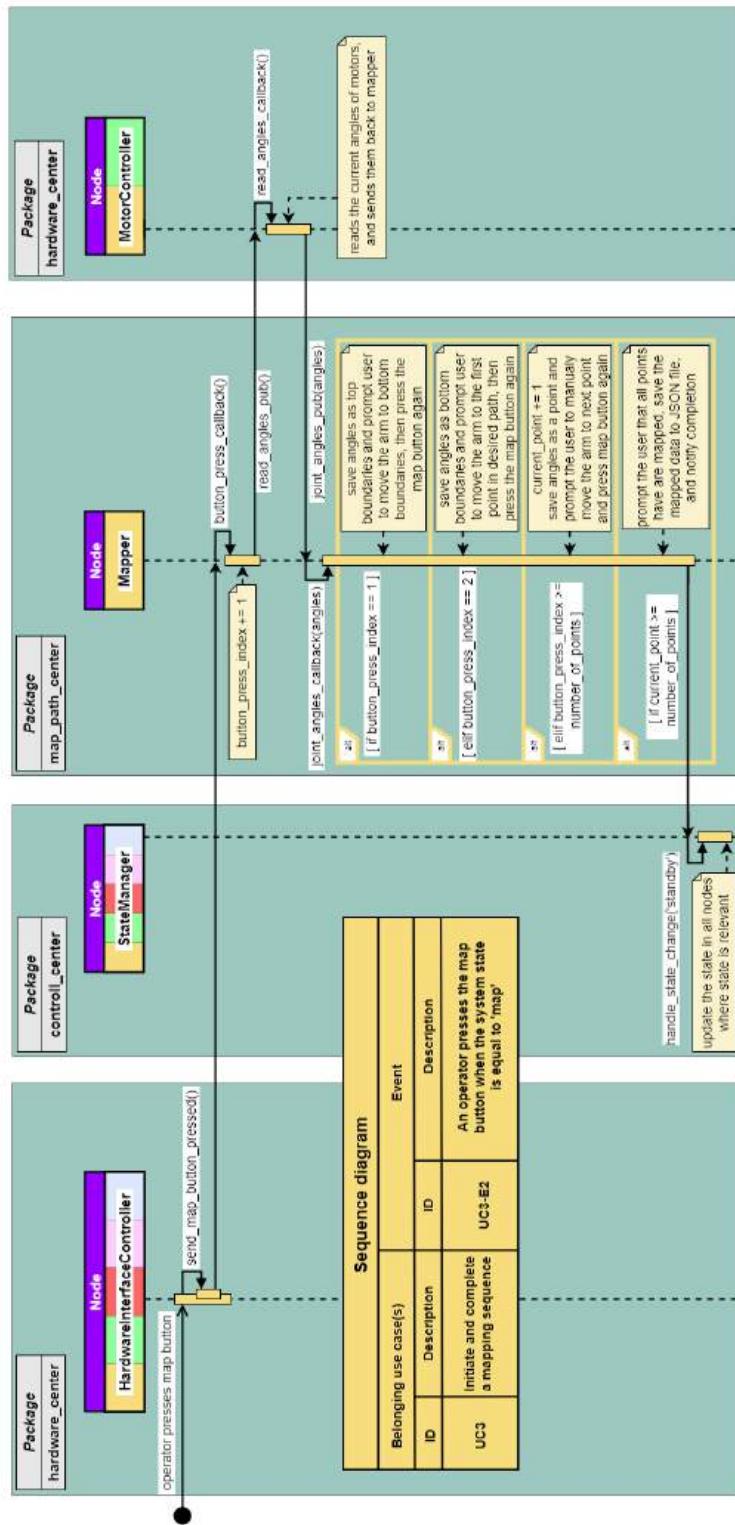


Figure 144: Sequence diagram describing how the mapping sequence works from a lower level. Here we can see each node in operation during the sequence and the messages and function calls between them.

Naturally the mapping sequence has undergone changes during development, this is the second iteration and is simple compared to the first. You can read about the first iteration in App. N, under the section: "The first iteration of the mapper node".

### 6.3.5 Implementation of use case: "Run a predefined path"

TM | AL

#### What is the path execution and why we need it?

Executing a predefined path answers the system requirement *R1.0*, which proves that the robot arm is able to perform the movement needed by TE in their packing machine.

#### How it works.

The functionality for executing a predefined path is split between the *Path* and *MotorController* node, ensuring modularity through encapsulation. When the operator presses the path button, the *Hardware\_Interface\_Controller* sends a state update request to the *StateManager* node. The request is approved if the current system state is *standby*. If this is the case, the system state is updated to "run\_predefined\_path", and the *Path* node initiates the sequence. The node reads the JSON file generated by the *Mapper*, if the file does not exist or contain the right data, the system state is set back to *standby* and the operator is prompted with a relevant message. If the file contains angles, they will be stored in an array, and sent to the *MotorController* for execution. In the *MotorController* there are three main functions responsible for moving the servos through the path. The first function, *follow\_path()* is executed as the *MotorController* receives the array with *target\_angles*. The function is responsible for reading the array and adding each set of angles to a queue, one set at the time. This will happen multiple times during a path execution, as the function reads the array from front to back, and then from back to front to reverse the movement. It also keeps track of the number of completed movements, ensuring that the sequence is correctly stopped after a certain number of iterations. Afterwards, the *StateManager* is notified of completion.

When the *MotorController* node starts up, it initializes two threads, one for each servo. Both threads are running the function *servo\_control\_loop(servo\_key)*, which is a constant loop reading the queue for new target angles, that calls another function *control\_servo\_speed(servo\_key, target\_angle)*. This function is responsible for moving the servos given a target angle, and stopping when it reaches this angle within a specified tolerance. Its designed to control the speed and direction of servo movement using a Proportional integral derivative controller that compares the target angle and the current angle, ensuring precision and smooth movement.

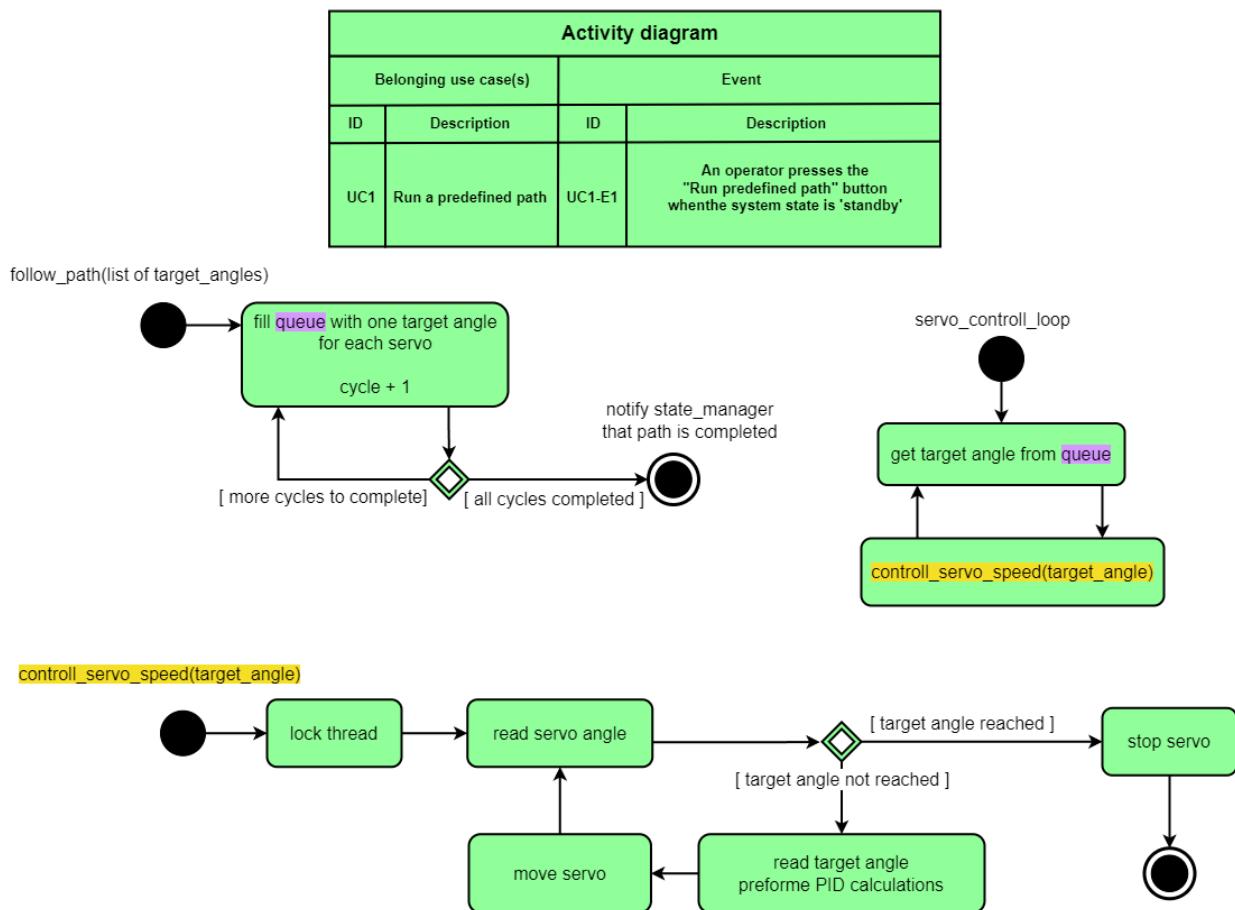


Figure 145: Activity diagram describing the flow of path execution in the Motor Controller node, each diagram represents a function

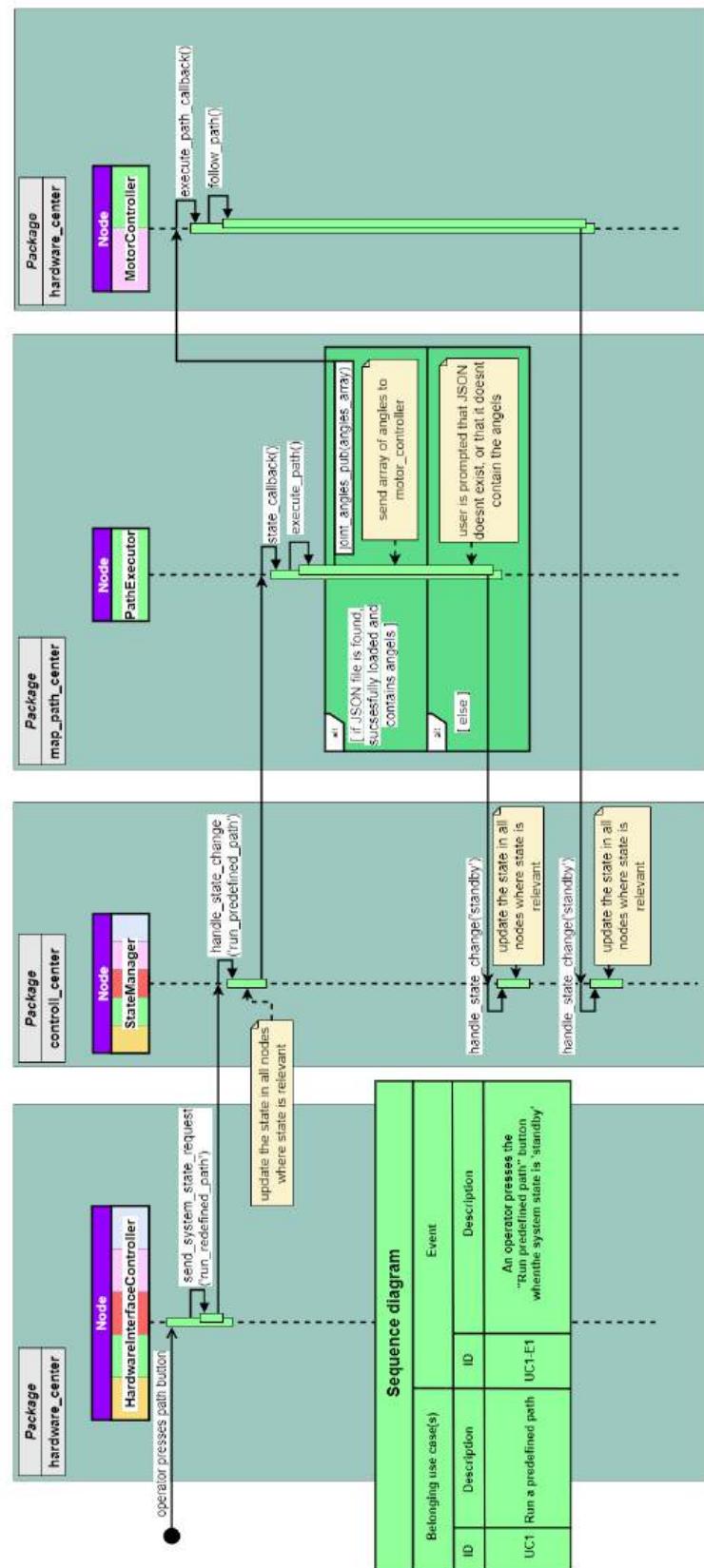


Figure 146: Sequence diagram describing the interaction between nodes during the path execution.

How does the PID controller work?

The PID controller is a control mechanism used to regulate the output so that it matches the decided value. Its ideal for regulating process variables like temperature, speed, pressure and more. The controller works by continuously calculating and adjusting the control output based on the difference between the desired set-point and the actual measured value. The control output is the sum of three component the **proportional**, **integral**, and **derivative** values of the error. Each component also includes a gain variable that defines how much the component will affect the output value. Developers fine tune the gain values to achieve a responsive and stable output value. [130][130]

- **Proportional (P):** The proportional element of the PID controller produces an output that is directly proportional to the current error. The purpose of the proportional, is to have a large immediate reaction on the output to bring the process value close to the set point. As the error becomes less, the influence of the proportional value on the output becomes less. [130][131]

Equation for proportional:

$$P = K_p \cdot \text{Error} \quad (6)$$

$$\text{Error} = \text{target value} - \text{current value} \quad (7)$$

$$K_p = \text{proportional gain} \quad (8)$$

- **Integral (I):** When the PID controller performs the calculation, the new calculated integral value, is added to the integral total. The integral will normally not have as much immediate influence on the output as the proportional, but because the integral is continuously accumulating over time, the longer it takes for the process value to reach the set point, the more effect the integral will have on the output. The purpose of the integral part is to make smaller fine-tuning of the output. [130][131]

Equation for integral:

$$I = I_{\text{previous}} + K_i \cdot \text{Error} \cdot \Delta t \quad (9)$$

$$I_{\text{previous}} = \text{the accumulated integral value from the previous calculation} \quad (10)$$

$$\text{Error} = \text{target value} - \text{current value} \quad (11)$$

$$K_i = \text{integral gain} \quad (12)$$

$$\Delta t = \text{Time interval between calculations.} \quad (13)$$

- **Derivative (D):** This component calculates the rate of change of the error, offering predictive control. By assessing how quickly the error is changing, the derivative control can apply a damping effect, which smooths the approach to the set-point and minimizes overshoot. [130][131]

Equation for derivative:

$$D = K_d \cdot \frac{\text{Error} - \text{Error}_{\text{previous}}}{\Delta t} \quad (14)$$

$$\text{Error} = \text{target value} - \text{current value} \quad (15)$$

$$\text{Error}_{\text{previous}} = \text{the error from the previous calculation} \quad (16)$$

$$K_d = \text{derivative gain} \quad (17)$$

$$\Delta t = \text{Time interval between calculations} \quad (18)$$

Using a PID controller is a great choice in this scenario, as it adjusts speed and direction of the motors dynamically based on real-time feedback, delivering precise and smooth motion control.

### 6.3.6 Implementation of use cases: "Control the system with a joystick" and "Activate the rail movement" & the combination between use cases: "Control the system with a joystick" and "Initiate a mapping sequence"

AL | TM

#### Developing an initial test script:

The implementation process, of the first two use cases, described in this sub-subsection, was the precursor to the creation of the *Hardware\_Interface\_Controller* node and required continuous iterations and adaptability. The original plan involved utilizing robot position coordinates, associated with angles, generated from the mapping sequence. By combining the robot's current coordinate with analog values, the tool hub would then be moved towards the max-coordinate in alignment with the direction indicated by the analog values. The movement was supposed to be generated by the move() - method of the LSS library, which takes a (10 \* X) value as a parameter, where X represents the destination angle. However, due to the fact that the mapping sequence was not fully refined and completed at the time of initiation, another approach was considered: Controlling and moving the LSS-HS1 motors without direct angle manipulation. The first alternative presented itself through the utilization of the PWM signals, but as we lacked the necessary cabling to realize the alternative, the idea was quickly discarded in favor for the built in wheelRPM() - method of the LSS library. This method take a set RPM as a parameter and rotates indefinitely at the given speed until an RPM of 0 is passed as a parameter. With this in mind, a test script was created to combine the inputted analog values with fitting movement of the servos, and the alternative approach proved successful. The test script worked by running the main functionality in a simple "while True" loop, and the analog ranges were divided into a cardinal and ordinal direction system with appropriate intervals, providing the user with flexibility, while accounting for imprecise values from the joystick. See Fig. 147 for a visual representation of the analog value ranges.



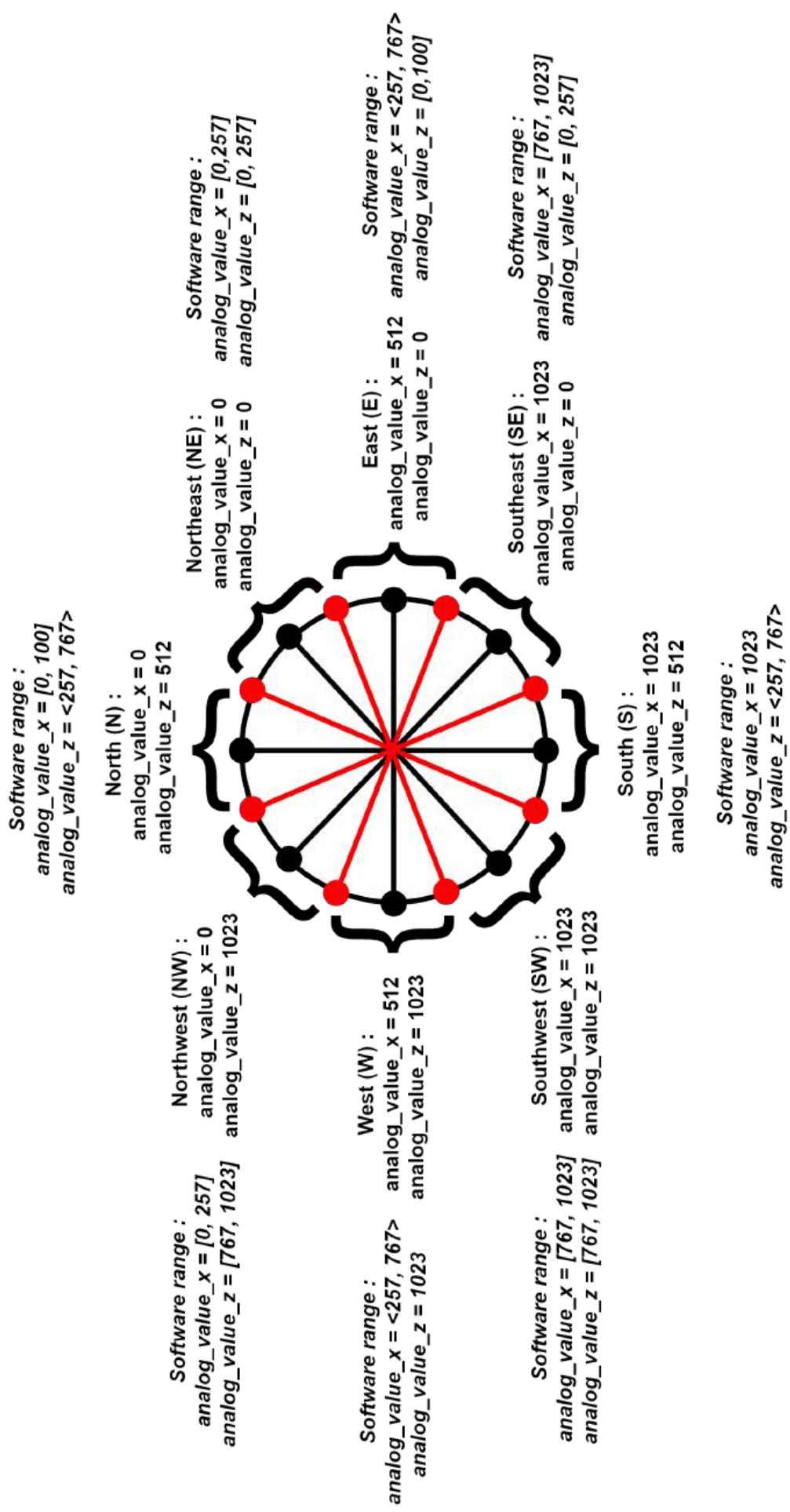


Figure 147: Analog value ranges in a cardinal and ordinal direction system

**Implementing boundary checks:**

After several test runs, it became clear that implementing boundary checks was essential to prevent users from attempting to extend the active arms beyond the physical boundaries of the framework. Such attempts would likely trigger the motors to enter emergency mode and potentially cause damage to the arms and/or supporting structures. We began configuring the feature using the clamp - functionality from the NumPy library. This functionality controls a given value with respect to two other parameters: a maximum and minimum value, by adjusting it to ensure it always remains within the maximum and minimum values. In this case, the evaluated value was the LSS-HS1 motors' active position, and the parameters were the boundary limits of the servos. Unfortunately, during testing, the clamp functionality exhibited unpredictable triggering, which led to accidental crashing and skipping of the gear teeth. This also led to a downward spiral as the boundary positions changed. The solution came in the form of a simpler approach: utilizing the boundary limit positions directly in if-conditions, this alternative solution proved successful.

**Implementation of joystick button presses and rail system activation:**

At the time, joystick button presses and a third LSS-HS1 motor were also implemented into the software design of the test script. The joystick button presses were intended to activate the joystick control of the arms, along with the third LSS-HS1 motor, which would represent the rail system. Implementing the joystick button presses required several back-and-forths between attempting to utilize the GPIO pins with an Analog-to-Digital (ADC) converter chip or the Arduino as a intermediary communication device, due to an incorrect number of high signals being processed during a single button press. But we managed to end up with a solution that adhered to the first principle mentioned in the sub-subsection: "Configuring communication and data transmission between the hardware components".

**Transforming the test script to a ROS node:**

Subsequently, the test script was ready to be transformed into the *Hardware\_Interface\_Controller* ROS node. A publisher and a subscriber variable were implemented and the main functionality of the test script was added to separate methods of the node class. The publisher's task was to send state requests to the *state\_manager* node trough a topic called: "system\_state\_request", while the subscriber would listen for state updates on another topic called: "system\_state". When a state update occurred, it would trigger a callback function to activate the appropriate functionalities. The joystick button press was intended to provide the first publish request and initiate the communication between the two nodes. Therefore it was also launched in a separate thread. This thread performed as expected and published a request to set the system in the *joystick\_arm\_state*. However, an issue arose when the callback function failed to activate upon

the second button press, indicating a user request to activate the rail system and set the system state to *joystick\_rail\_control* state. This problem emerged due to a fundamental limitation in ROS: If function calls are implemented inside callback methods, the callback functions cannot be recalled until they have completed their last function call. To counter this problem we created a "current system state" variable of the *Hardware\_Interface\_Controller* node class, which was to be assigned a string in a single operation within the callback method.

The main functionalities, controlling the arm with the joystick and activating the rail movement, were also separated into additional threads, meaning we now had three threads operating. The idea was to use the "current system state" variable and, depending on the stored system state, activate the correct functionalities. This proved easier said than done, as the threads executed functionalities independently of the change of the internal state variable. We tried to solve the issues with time blocks, additional condition checks and event-driven programming, but to no luck. At this point, a decision was made to revert parts of the software design back to the test script, while maintaining the overall design as a ROS node. The main functionality threads were removed, and the activation of functionalities happened through different if-condition checks in a "while true" loop within a try - except block inside the main() - method. Lastly, The rcply.spin() method, a function of the Python client library for ROS2 necessary to enable publishing and subscribing between nodes, was launched in a separate thread to prevent blocking of code execution inside the try-except block. This design represents the last structure of the *Hardware\_Interface\_Controller* node, allowing for modular development in the simple sense of adding additional condition checks to the "while true" loop.

### **Integration with the mapping sequence and further development:**

Further development involved finalizing the handling of the three buttons: "run a predefined path", "reset" and "map", and the incorporated button of the joystick, on both the transmitting Arduino script and on the receiving *Hardware\_Interface\_Controller* node, as well as refine the "control arm with joystick" functionality to adhere to the *map* state and finalize the mode of operation for the rail activation. In regards to the button presses of the joystick, the activity diagram in Fig. 148 illustrates the generic system processes required to properly handle the input. For a detailed illustration of the system flow of the same event, see the sequence diagram with descriptions in App. N, under the section: "Lower-level architecture" and the subsection: "UC4&5-E1 - An operator presses the incorporated button of the joystick".

Activity diagram			
Belonging use case(s)		Event(s)	
ID	Description	ID	Description
UC4	Control the system with a joystick	UC4&5-E1	An operator presses the incorporated button of the joystick
UC5	Activate the rail movement		

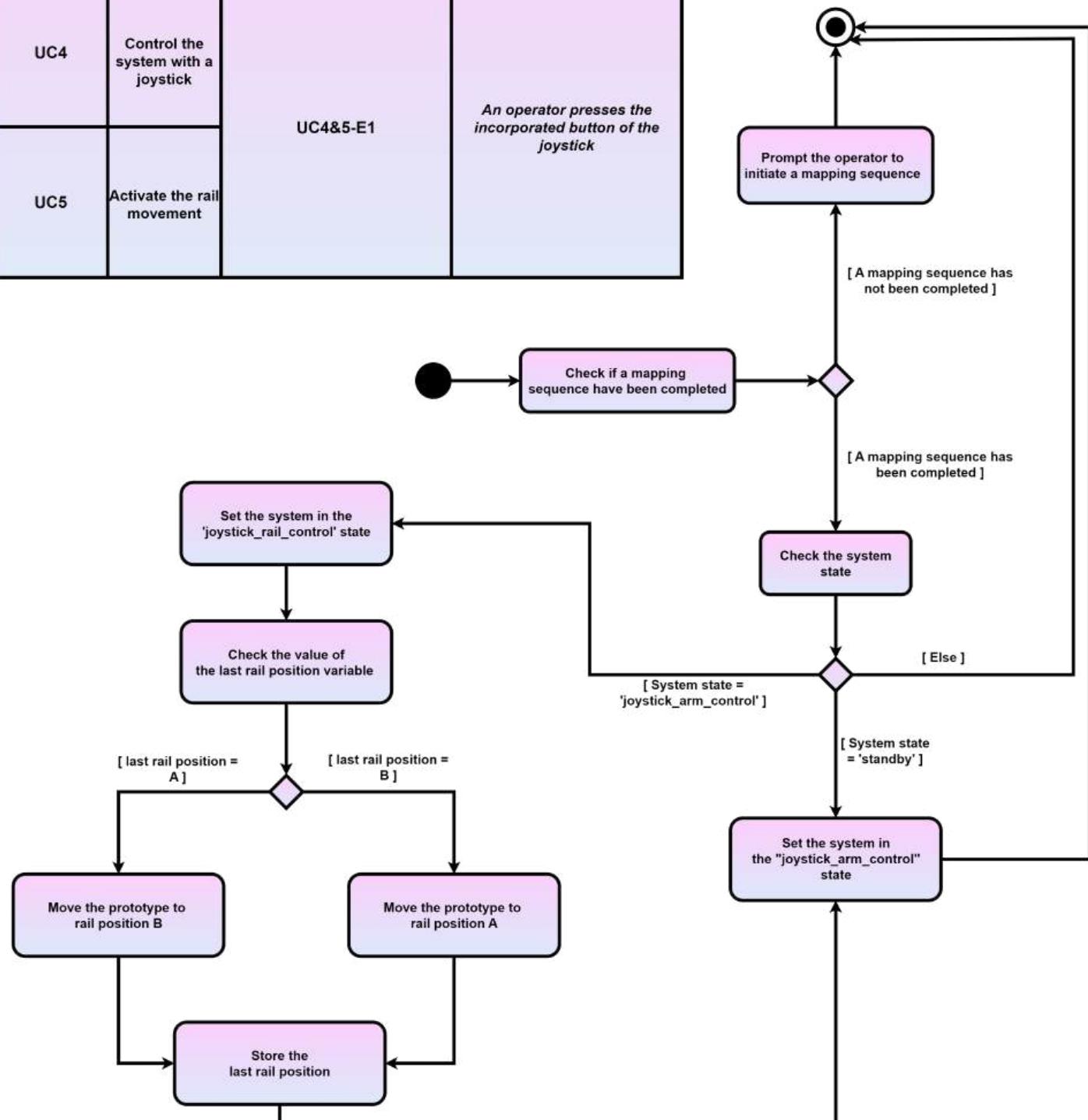


Figure 148: Activity diagram - An operator presses the incorporated button of the joystick

In regards to the "control arm with joystick" functionality, which was now going to be used in the mapping sequence, we deemed it a necessary measure to meet safety demands and the established plans for the mechanical structure. Initially, our plan was to make the user physically move the arms to map null points, boundaries and path points during a mapping sequence, but as this was less user-friendly in terms of security, the "control arm with joystick" functionality was modified to be used here as well. The main difference in how the functionality operates between the *joystick\_arm\_control* and *map* state, lay in the way the boundary limits were handled. Since the users are going to set their own top and bottom boundaries during a mapping sequence, it made sense to eliminate the boundary checks. Some may argue that this places too much trust in the operator to properly handle the prototype, but it was a compromise we deemed necessary to achieve the desired functionality. Another option could involve setting absolute maximum and minimum boundaries. However, since these depend on the position of the null points, we would need the users to establish the exact same null point positions as us for the boundaries to effectively control the workspace. Naturally, hoping for this scenario is not reliable. Fig. 149 describes the system flow for executing a north or northeast movement of the joystick, resulting in north or northeast movement of the prototype. For a detailed description of the system flow of the same event, see the sequence diagram with descriptions in App. N, under the section: "Lower-level architecture" and the subsection: "UC3&4-E1 - An operator moves the joystick towards the north or northeast.

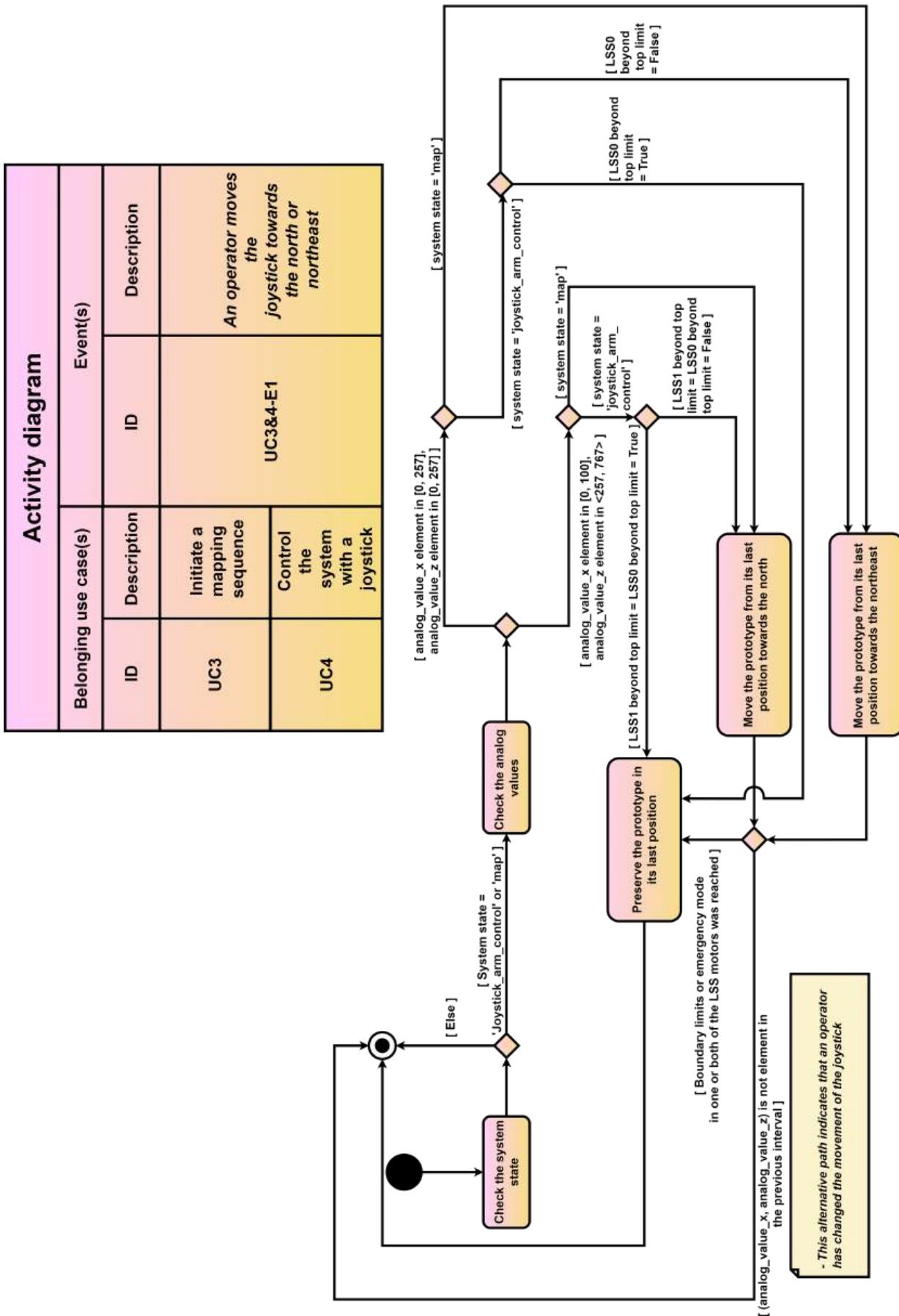


Figure 149: Activity diagram - An operator moves the joystick towards the north or northeast

As for the other cardinal and ordinal directions, their respective activity and sequence diagrams can be viewed App. N, under the section: "Lower-level architecture" and the subsections with event identifications: "UC3&4-E2 ⋯ E4". These do not have additional explanations as the execution processes, beside the boundaries and analog values condition checks, are identical to the description flow of executing the north or northeast movement. The flowchart under the subsection with event identification: "UC3&4-E5", also illustrates the system flow of executing a scenario where the operator stops moving the joystick. To clarify the movement directions of the prototype in correlation with the joystick, Fig. 150 was made. The movements of the prototype are relative to a fixed cardinal and ordinal direction system on a hypothetical line between the joint transitions of the left and right active and passive arms.

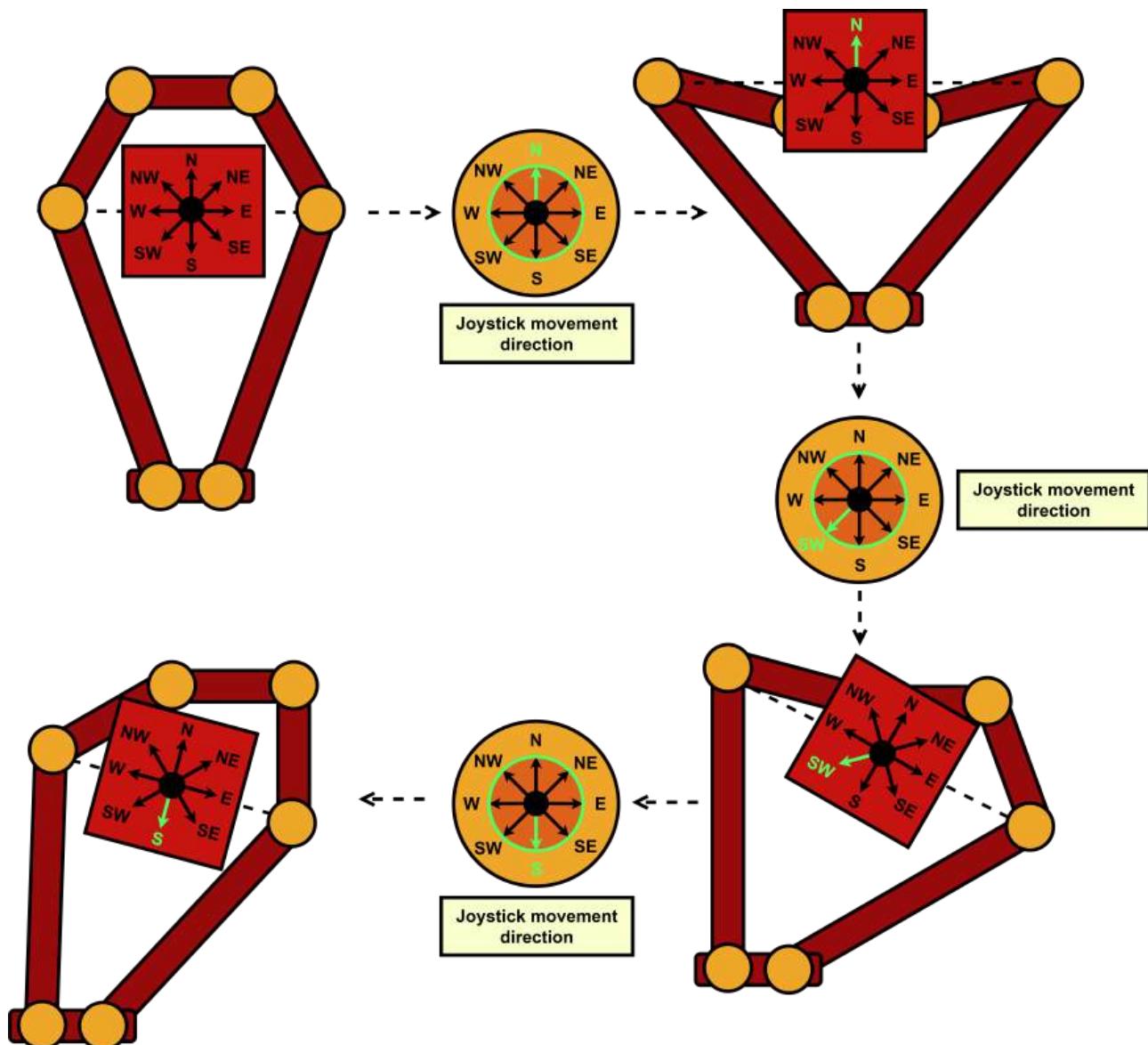


Figure 150: Visualization of the cardinal and ordinal direction system in arm and joystick movements

**6.3.7 Implementation of use case: "Stop the active process"**

AL | TM

During the process of implementing the reset button, compromises had to be made to finalize the end result. Originally, the plan was to enable the operator or user to cancel a run of a predefined path, the activation of the rail system and control of the arms in the *joystick\_arm\_control* state. However, due to synchronization issues in the *run\_predefined\_path* state, accompanying time constraints, and the structural code build of the rail activation, these options were discarded from the software design for handling reset button presses. Regarding the rail activation code, a reset button press would cause problems because the rail movement depends on the LSS-HS1 motor with ID: 2, to rotate for specific time at a given rpm, to move the prototype back-and-forth between a distance of 10 cm. A potential midway reset and reactivation of the rail system would cause the LSS-HS1 motor with ID: 2 to rotate for a period equivalent to a longer distance than the actual distance to the present endpoint. This would most certainly lead to skipping of gear sheet, and/or damage to the LSS-HS1 motor. The problem could have been solved by implementing joystick movement to control the rail system, but at the time of deciding the mode of operation the alternative was set aside in favor of a more simplistic approach to achieve the requirement with ID: R3.0. Ultimately, we were only able to reset the *joystick\_arm\_control* state back to *standby*. But this does not make our system less secure. The emergency stop button can still instantly shut down all active processes in case of crisis. Fig. 151 illustrates the system flow of executing an event where the operator presses the reset/"Stop the active process" button.

Activity diagram			
Belonging use case(s)		Event(s)	
ID	Description	ID	Description
UC2	Stop the active process	UC2-E1	An operator presses the reset/ "Stop the active process" button when the system state is in any of the other states beside the joystick_arm_control
		UC2-E2	An operator presses the reset/ "Stop the active process" button when the system state is equal to joystick_arm_control

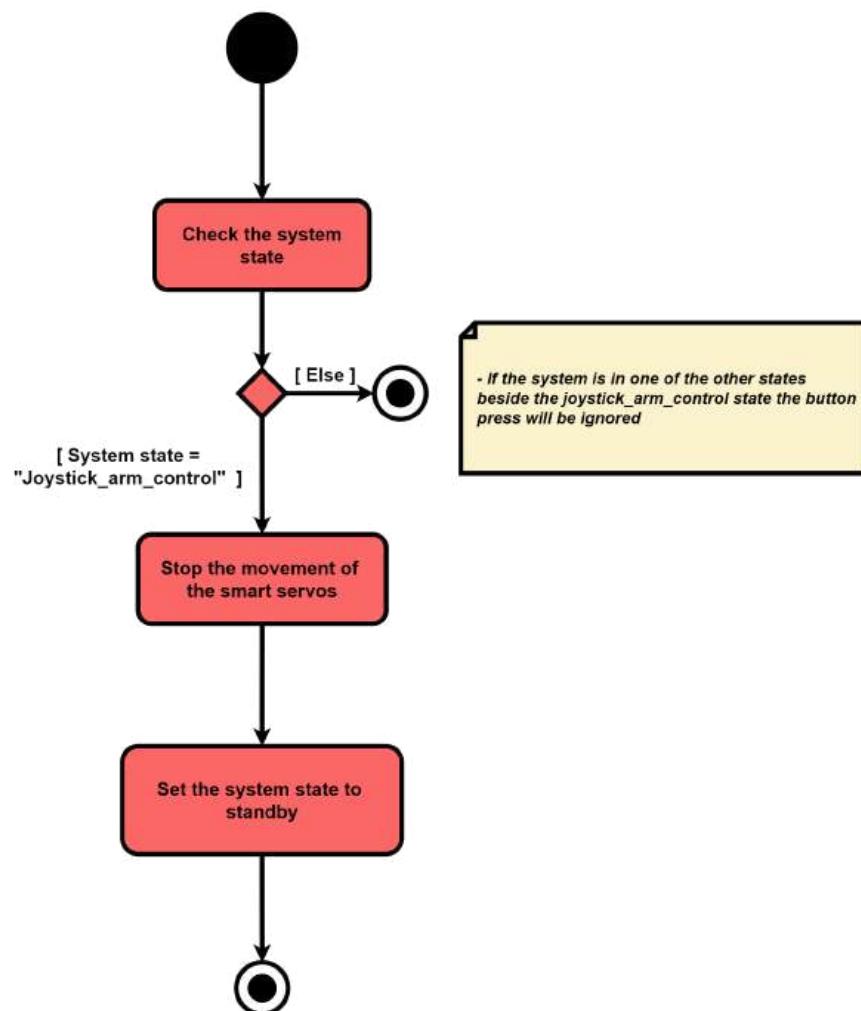


Figure 151: Activity diagram - An operator presses the stop the active process button

### 6.3.8 Software development principles

TM | AL

#### From idea to code

The development process adopted involves an incremental and modular approach, starting with an initial idea and gradually building the system. By utilizing this method it's important to validate each component's functionality before progressing to the next, ensuring robustness and reliability. Throughout this process, the overarching architecture of the system is kept in focus, guiding the placement and integration of each component. Logical grouping of related functionalities under appropriately named categories not only aids in making the codebase organized, but also enhance the understandability and maintainability of the system.

#### Debugging

Logging is an essential tool for debugging during software development, particularly in complex systems like those built with ROS2. It enables developers to track the flow of execution and the state of the system, providing insights into what the system is doing at any given time. By inserting log statements at critical points within the code, developers can output useful information such as variable values, error messages, and the status of operations, which helps in identifying potential breaking points. In our case this has been extremely valuable during the development in ROS2, as we are dealing with asynchronous events and interactions between different system components.

#### Isolated development and testing

In a large system where components coexist and are dependent on each other we need methods for testing a component in isolation. The Mapper functionality requires the operator to interact with the system, but during early development, the functionality for reading button presses was not yet realized. Therefore, we used GUI buttons to initiate the mapping sequence, but this was later switched out for a physical button press as we succeeded in establishing the functionality. We also made a simple command system, preceding the implementation of the mapping button, that would let us read angles and set nullpoints for the servos.

### 6.3.9 Ensuring the system is exhibition ready

TM | AL

One of the system requirements is that the system must be able to be moved and used for exhibition (R8.0). To fulfill this requirement on the software end we have set up the Raspberry PI to automatically login and run a bash script on boot, which builds and launches the ROS2 project. This aided us in fulfilling both R8.0 and R7.0 (The prototype must be user-friendly), as the operator does not have to manually login and navigate to the right file to launch the project. We also made additional efforts to make the Raspberry PI identify the Arduinos dynamically at startup. This was done through the use of Ubuntu's udev rules and the device manager for the Linux kernel to create persistent device names for serial ports. Each Arduino is identified using unique hardware attributes such as serial numbers extracted from the USB

device's descriptors. We wrote specific udev rules that matched these attributes, assigning predictable and consistent symlink names to the serial ports. Enabling ROS to identify devices dynamically at startup.

### 6.3.10 Key takeaways and recommendations in hindsight

AL | TM

By evaluating the software development process from the time of initiation to the time of closure, multiple insights and key takeaways were acquired. First of all, is the key takeaway and recommendation that conducting thorough research is an essential part for knowing your system's limitations. Putting effort into this process, shortens the time spent in debugging and troubleshooting, and aids you in foreseeing the impact of different hardware choices. Since ROS was a completely new framework to the software engineers of our group, and the initial framework research was predominantly academic, it was only natural that some limitations went unseen during this period. An additional research element we could have implemented, which we also recommend to future students or groups, is to run small open-source test projects of your new framework of choice to also grasp the practical applications of the academic insights. Not only does this mitigate the likelihood of spending countless hours researching approaches for features that are not supported, like for instance, in our case, writing over serial in ROS utilizing the standard pySerial library or opting to continuously trigger callback methods before they are done compiling, but it also provide you with an even greater understanding to construct solid software architectural diagrams, which significantly accelerates the software development.

Should it be the case that you are operating with a single board computer, the SSH remote extension of VSC serves as an invaluable component to enable efficient testing, troubleshooting and iterative development. This software component was deemed absolutely essential, in our case, since issues and the need for adjustments and testing arose on a consistent basis from day to day. Lastly, we would heavily emphasize the importance of acquiring robust and solidly manufactured hardware to reduce the likelihood of encountering issues, such as the communication halts we experienced with the Arduino Nano clone. On the other hand, should you decide to purchase and implement unofficial hardware devices, it is crucial to bear in mind the potential issues that could arise as a consequence of opting to reduce expenditures, and establish appropriate strategies to efficiently tackle them.

## 7 Economy

### 7.1 Product economy

AK | AL

In the beginning of the bachelor project, the group was tasked with the development of a Delta robot. For this task, various research topics were derived, including product economy. Cost efficiency is of the utmost importance for our stakeholders at TE, both at the production level, when establishing production methods, and when ordering parts externally. Therefore, our group intended to perform a deeper assessment using a Techno Economic Analysis. TEA is a process for businesses to gain insight on profitability, based on engineering and manufacturing, and how these aspects affect their product. Techno refers to the energy and material utilization and how insights can be applied to develop cheaper and more efficient products. Once the technical aspect is assessed the economical analysis can proceed. This involves acquiring data on topics included but not limited to: raw materials, energy consumption, manufacturability, labor and infrastructure. [132]

However, after redefining the scope, the time frame for conducting development and detailed research was reduced. Therefore, no TEA has been conducted. Nonetheless, there have been an emphasis on cost efficiency and beneficial production processes. As for the ordered components, TE provided a list of suppliers, many of which provided company deals and discounts for bulk orders. In regards to the parts designed for production at TE, these are configured in consideration to various manufacturing processes, such as reductions in machining time and material waste. For example, the motor brackets and fastening hub were initially intended to be milled, but ended up being utilized waterjet cutting and sheet metal. See Figs. 152 and 153 for the expenditures provided by TE.

Nr	Beskrivelse			Lev.dato	Antall	Enhett	Nettopris	Beløp
	Lengde	Bredde	Enhett	Antall		Merke		
	mm	mm	mm	2		HL!		
0.0.370.03	Profil 5 20x20			07.05.24	6	m	149,25	895,50
	Lengde	Bredde	Enhett	Antall		Merke		
	3 000	0	mm	2		HL!		
0.0.425.03	Vinkel 5 20x20 Zn			07.05.24	32	STK	71,25	2 280,00
0.0.370.01	Spormutter 5 St M5			07.05.24	150	STK	12,60	1 890,00
0.0.677.77	Vinkel 5 20, hvit Al.			07.05.24	8	STK	51,75	414,00
<b>Totalt ekskl mva NOK</b>								<b>5 479,50</b>

Figure 152: Expenses provided by TE 1

IGRFRM282835TR12x3	DRYSPIN FLANGE LEAD SCREW NUT w/SPANNER FLAT RFRM-282835TR12x3	21.05.24	1,000	PCS	218,30	0,00	218,30
IGRJUM0112	DRYLIN LINEAR BEARING RJUM-01-12	21.05.24	4,000	PCS	227,50	15,00	773,50
IGGFM141604-0010	IGLIDUR G FLANGE BEARING GFM-1416-04	21.05.24	10,000	PCS	42,20	20,00	337,60
IGGTM0612015-0010	IGLIDUR G THRUST WASHER GTM-0612-015	21.05.24	11,000	PCS	41,60	20,00	366,08
IGJSM060810	IGLIDUR J SLEEVE BEARING JSM-0608-10	21.05.24	2,000	PCS	87,60	20,00	140,16
IGJSM030409	IGLIDUR J SLEEVE BEARING JSM-0304-09	21.05.24	2,000	PCS	89,90	20,00	143,84
IGGFM030402	IGLIDUR G FLENSLAGER GFM-0304-02	21.05.24	6,000	PCS	74,70	20,00	358,56
IGKBLM03	IGUBAL ROD ENDS KBLM-03	21.05.24	6,000	PCS	92,70	15,00	472,77
IGJFM060804-0025	IGLIDUR J FLANGE BEARING JFM-0608-04	21.05.24	30,000	PCS	31,20	20,00	748,80
						<b>Sum order lines:</b>	<b>3 559,61</b>
						<b>Servicefee:</b>	<b>0,00</b>
						<b>Postage:</b>	<b>0,00</b>
						<b>Sum order:</b>	<b>NOK 3 559,61</b>

Figure 153: Expenses provided by TE 2

## 7.2 Project economy

AL | AK

Apart from the expenditures provided by TE, Fig. 154 displays a list of expenditures provided by our group members. These temporary outlays was a necessary expense to support present activities, complete minor goals and facilitate progression across disciplines.

Project economy			
Product name	Date of purchase	Vendor	Sum (NOK)
1x RINGPERM ESSELTE A4			
1x HULLEMASKIN P2 2-HUL	02.02.2024	Norli Akademisk Kongsberg	137
1x SKILLEBLAD KARTONG A			
2x NEODYM-MAGNET	06.02.2024	Clas Ohlson	200
2x NEODYM-MAGNET	13.02.2024	Clas Ohlson	200
1x PLA FILAMENT SVAR	05.03.2024	Clas Ohlson	300
1x Power Cable (EU Plug)			
1x Lynxmotion SES-V2 100-240VAC to 12VDC 6A Power Sup	15.03.2024	RobotShop	829,18
1x Lynxmotion SES-V2 LSS Adapter Serial & Power Distr			
3x Lynxmotion (LSS) - 300mm Serial Cable			
1x MEAN WELL Switching Power Supply			
2x MCP3008-I/P mva	15.04.2024	RS	692,31
1x Vanntett LED lyssløyfe med fjernkontroll 2,5 meter Sort	16.04.2024	Elkjøp	145
1x Apparatinntak C14 for kabinet			
1x Apparatkabel C13 svart 3 m Frakt	16.04.2024	Kjell & Company	238,8
1x Arduino Nano ATmega328p CH340g utviklingskort	01.05.2024	ArtigereLiv	160
1x ST 012520 SUPERLIM HH 2G	02.05.2024	JULA	1213,9
Belkin USB 2.0 forlengelseskabel (4,8 m)	13.05.2024	Elkjøp	199
USB2-KABEL TYPE A	13.05.2024	Elkjøp	130
1x XT60 kit 10 par Hane/Hona, 40/40cm krym	18.05.2024	Kjell & Company	199,9
1x EXTREME BORRELÄS	18.05.2024	Clas Ohlson	180
1x USB 3.0 kabel A/Ah Forlengningskabel, svar	18.05.2024	Kjell & Company	149,9
		Total	4974,99

Figure 154: Expenditures provided by the group members

## 8 Conclusion

During our project's lifespan, we have made continuous efforts to produce an academic thesis that addresses TE's need for thorough documentation, while constructing a product that aligns with the technical aspects of our task description and fulfills TE's fundamental requirements listed in the requirement table (See App. B). Our group has explored the functionality of multiple pick-and-place robots, provided insights regarding disadvantages and advantages for various concepts in correlation with TE's packaging machinery, examined production possibilities for machining processes of mechanical parts, and developed a downscaled solution that aims to provide the similar mode of operation as the delta robot, while also being more cost-efficient in comparison.

However, as most projects often involve complex tasks and coordination of interdisciplinary teams, the likelihood of encountering challenges and issues are almost certain, and our project was no exception. Our most significant issues, that impacted our team across disciplines, mainly stem from inadequate motor research in the project's early stages. At the time of purchase, the motors seemed sufficient and optimal for the intended use, which led us to promptly acquire the hardware. The latter was performed with an effort to rapidly initiate various development processes across disciplines. Unfortunately, after the motors proved to be inefficient during the initial dynamic payload testing without gearing, a severe ripple effect originated. The implementation and creation of gears, along with theoretical downscaling of the Senior model, created roadblocks in the expected delivery of 2D drawings for production. These roadblocks or challenges included mass reduction, redesign of components within assemblies, and cross-integration of assemblies to ensure coherence between the various assemblies when combined into a bigger system. Delegating additional resources to these processes was necessary to assure the projects integrity and overarching goal, but it came at an expense. Our group, unfortunately, neglected the importance of maintaining the internal meetings without supervisors, stated in our group contract, which led to minor communication issues and occasional reductions in the general project grasp. Luckily, we immediately strove to handle these challenges, ensuring that no additional amount of time was unnecessary wasted and that every team member's understanding remained aligned.

Towards the latter half of semester, after the 2D drawings were delivered, we experienced unfortunate logistical transport incidents and an elongated production time. To successfully deliver a finite product, a few concluding efforts from each discipline are needed. Acquiring the ordered materials, including IGUS bearings, MiSUMi parts, and the Item profile for the frame to construct the Senior model represents the concluding steps for our mechanical engineers to finalize the prototype. Regarding the efforts related to the electrical discipline, performing tests and integration between the power supply and the electrical components represent the final procedures. Lastly, the software engineers need to ensure automatic port setting with a

more reliable microcontroller.

## References

- [1] “Galvanic corrosion&galvanic cell explained.” [Online]. Available: <https://galvanizeit.org/corrosion/corrosion-process/galvanic-corrosion>
- [2] “drylin® sht-12 lead-screw driven actuator with shafts,” May 2024. [Online]. Available: <https://www.igus.com/product/20520>
- [3] n. null, *SmallRig 2143 Quick-release Clamp Arca-type CEWE Japan Photo*. [Online]. Available: <https://www.japanphoto.no/produkt/smallrig-2143-quick-release-clamp-arca-type>
- [4] ——, *SmallRig 2143 Quick-release Clamp Arca-type CEWE Japan Photo*. [Online]. Available: <https://www.japanphoto.no/produkt/smallrig-2143-quick-release-clamp-arca-type>
- [5] “Fiber volume fraction - an overview.” [Online]. Available: <https://www.sciencedirect.com/topics/engineering/fiber-volume-fraction>
- [6] “3m adhesives & tapes solutions,” May 2024. [Online]. Available: [https://www.3m.co.uk/3M/en\\_GB/bonding-and-assembly-uk/substrate/composites-adhesive-glue/](https://www.3m.co.uk/3M/en_GB/bonding-and-assembly-uk/substrate/composites-adhesive-glue/)
- [7] “Linear vs. switching power supplies,” Mar 2022. [Online]. Available: [https://www.matsusada.com/column/linear\\_switching\\_difference.html](https://www.matsusada.com/column/linear_switching_difference.html)
- [8] n. null, “Zl2pd introduction to switchmode power supplies,” Jun 2010. [Online]. Available: <https://www.zl2pd.com/introSMPS.html>
- [9] R. Components, “Varta cr2450 coin cell datasheet,” <https://docs.rs-online.com/049f/0900766b81622002.pdf>, accessed: 2024-05-16.
- [10] “Lss - specifications,” Sep 2020. [Online]. Available: <https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/lynxmotion-smart-servo/lss-specifications/>
- [11] “Galvanic series (electrochemical series),” May 2024. [Online]. Available: [https://structx.com/Material\\_Properties\\_001.html](https://structx.com/Material_Properties_001.html)
- [12] T. Engineering, “About tronrud engineering.” [Online]. Available: <https://www.tronrud.no/en/about/>
- [13] K. Moholth, “Veiledningsavtale for bacheloroppgave tse3000 vår 2024,” Jan 2024.
- [14] E. Helgerud, O. A. F. Resistad, P. A. Stadheim, S. L. Aronsen, T. S. Vatle, and . J.-C. Åslie, “Robotrim,” Ph.D. dissertation, USN, 2018.
- [15] D. Miller, “What is project management framework? & its benefits,” Jul 2021. [Online]. Available: <https://www.proprofsproject.com/blog/project-management-framework/>

- [16] Iconpacks.net, “Free icon.” [Online]. Available: <https://www.iconpacks.net/free-icon/logo-17027.html>
- [17] ——, “Free settings icon.” [Online]. Available: <https://www.iconpacks.net/free-icon/settings-8183.html>
- [18] ——, “Free pros and cons icon.” [Online]. Available: <https://www.iconpacks.net/free-icon/pros-and-cons-22296.html>
- [19] ——, “Free people icon.” [Online]. Available: <https://www.iconpacks.net/free-icon/people-12348.html>
- [20] ——, “Free handshake icon.” [Online]. Available: <https://www.iconpacks.net/free-icon/handshake-1939.html>
- [21] <https://www.iconpacks.net/free-icon/notes-and-pencil-red-notepad-22361.html>, [Accessed 28-02-2024].
- [22] W. Maclay, “Project management series: the importance of defined requirements in project management,” Jan 2022. [Online]. Available: <https://www.volersystems.com/blog/the-importance-of-defined-requirements-in-project-management>
- [23] E. Golightly, “How to write effective project requirements with examples,” Aug 2023. [Online]. Available: <https://clickup.com/blog/project-requirements/>
- [24] Jul 2023. [Online]. Available: <https://www.nasa.gov/reference/appendix-c-how-to-write-a-good-requirement/>
- [25] K. Eby, “The essential guide to project risk analysis,” Nov 2022. [Online]. Available: <https://www.smartsheet.com/content/project-risk-analysis>
- [26] *Methodology for Rapid Risk Ranking of H2 Refuelling Station concepts*, Sep 2002. [Online]. Available: [http://www.eihp.org/public/documents/RRR%20methodology\\_final\\_SEP2002](http://www.eihp.org/public/documents/RRR%20methodology_final_SEP2002)
- [27] “Articulated vs. pick-and-place robot arm: What’s the difference?” May 2024. [Online]. Available: <https://standardbots.com/blog/articulated-vs-pick-and-place-robot-arm-whats-the-difference>
- [28] <https://www.linearmotiontips.com/what-is-a-cartesian-robot/>, [Accessed 28-02-2024].
- [29] “7 types of industrial robots: Advantages, disadvantages, applications, and more,” May 2024. [Online]. Available: <https://www.wevolver.com/article/types-of-robots>
- [30] “What is a SCARA Robot? - Robots Done Right — robotsdoneright.com,” <https://robotsdoneright.com/Articles/what-is-a-scara-robot.html>, [Accessed 28-02-2024].
- [31] <https://epson.com/scara-robots-product-family>, [Accessed 28-02-2024].

- [32] <https://standardbots.com/blog/the-pros-and-cons-of-cylindrical-robots>, [Accessed 28-02-2024].
- [33] <https://robotec.org/blog/types-of-industrial-robot.html>, [Accessed 28-02-2024].
- [34] <https://processsolutions.com/what-are-the-different-types-of-industrial-robots-and-their-applications/>, [Accessed 28-02-2024].
- [35] <https://www.mwes.com/types-of-industrial-robots/polar-spherical-robots/>, [Accessed 28-02-2024].
- [36] <https://electricalworkbook.com/polar-robot/>, [Accessed 28-02-2024].
- [37] <https://www.linkedin.com/pulse/robot-pros-cons-everyone-automation-needs-know-jacob-sanchez>, [Accessed 28-02-2024].
- [38] <https://thnet.co.uk/thnet/robots/5a.htm>, [Accessed 28-02-2024].
- [39] <https://howtorobot.com/expert-insight/pros-and-cons-collaborative-robots-flexibility-vs-efficiency>, [Accessed 28-02-2024].
- [40] <https://www.wevolver.com/article/7-types-of-industrial-robots-advantages-disadvantages-applications-and-more>, [Accessed 28-02-2024].
- [41] <https://www.packagingstrategies.com/articles/95454-new-human-collaborative-robot-offers-20-kg-payload-for-variety-of-tasks>, [Accessed 28-02-2024].
- [42] <https://robotsdoneright.com/Articles/what-is-a-delta-robot.html>, [Accessed 28-02-2024].
- [43] <https://robotsdoneright.com/Articles/advantages-of-delta-robots.html>, [Accessed 28-02-2024].
- [44] <https://www.weiss-world.com/gb-en/products/robots-10327/delta-robots-211>, [Accessed 28-02-2024].
- [45] <https://www.dcm-italia.it/en/products/duopods/>, [Accessed 28-02-2024].
- [46] “Articulated vs. Pick-And-Place robot arm: What’s the difference? - Standard Bots — standardbots.com,” <https://standardbots.com/blog/articulated-vs-pick-and-place-robot-arm-whats-the-difference>, [Accessed 28-02-2024].
- [47] <https://www.electromate.com/meca500-six-axis-industrial-robot-arm/>, [Accessed 28-02-2024].
- [48] “Pugh matrix for problem-solving: How to make a decision matrix,” Jun 2022. [Online]. Available: <https://www.masterclass.com/articles/pugh-matrix>
- [49] “Design of 2-dof parallel manipulator,” Nov 2014. [Online]. Available: [https://www.researchgate.net/publication/319730332\\_Design\\_of\\_2-DOF\\_Parallel\\_Manipulator](https://www.researchgate.net/publication/319730332_Design_of_2-DOF_Parallel_Manipulator)

- [50] n. null, *Hex Socket Head Cap Screw With Spring Lock Captive Washer from MISUMI MISUMI*. [Online]. Available: [https://uk.misumi-ec.com/vona2/detail/110301996520/?rid=cat\\_&CategorySpec=](https://uk.misumi-ec.com/vona2/detail/110301996520/?rid=cat_&CategorySpec=)
- [51] “Optimal design of a 2-dof pick-and-place parallel robot using dynamic performance indices and angular constraints.” [Online]. Available: <http://dx.doi.org/10.1016/j.mechmachttheory.2013.07.014>
- [52] “Lss - specifications.” [Online]. Available: <https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/lynxmotion-smart-servo/lss-specifications/>
- [53] E. Nantel, “Lss-ada board (type-c),” May 2023. [Online]. Available: <https://wiki.lynxmotion.com/info/wiki/lynxmotion/view/servo-erector-set-system/ses-electronics/ses-modules/lss-adapter-board-type-c/>
- [54] “Ros official documentation,” 2018. [Online]. Available: <https://wiki.ros.org/ROS/Introduction>
- [55] “Ros official website/blog/why-ros.” [Online]. Available: <https://www.ros.org/blog/why-ros/>
- [56] “Ros used by nasa.” [Online]. Available: <https://www.therobotreport.com/open-robotics-developing-space-ros/>
- [57] “Example 1 of ros in the industry.” [Online]. Available: <https://robotsguide.com/robots/pr2>
- [58] “Example 2 of ros in the industry.” [Online]. Available: <https://www.slamcore.com/industrial-robotics/>
- [59] “Ros in the industry.” [Online]. Available: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-robot-operating-system/>
- [60] “Review of the pi 4 model b.” [Online]. Available: <https://www.tomsguide.com/reviews/raspberry-pi-4-model-b>
- [61] “Pi 4 model b specs.” [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>
- [62] “sw\_urdf exporter ros wiki,” May 2024. [Online]. Available: [https://wiki.ros.org/sw\\_urdf\\_exporter](https://wiki.ros.org/sw_urdf_exporter)
- [63] “Onshape product development platform,” May 2024. [Online]. Available: <https://www.onshape.com/en/>
- [64] “Power hd gts7 hv digital bl servo 70kg/0.10s,” May 2024. [Online]. Available: <https://www.elefun.no/p/prod.aspx?v=61230>

- [65] *RMD-L-7015-23T BLDC Motor Integrated With Motor Controller And Position Sensor CAN BUS* Oz Robotics, May 2024. [Online]. Available: <https://ozrobotics.com/shop/rmd-l-7015-10t-blde-motor-integrated-with-motor-controller-and-position-sensor-can-bus/>
- [66] “Spur gears, pom, milled, module 2,” May 2024. [Online]. Available: <https://www.maedler.de/product/1643/1618/1034/1050/stirnzahnraeder-kunststoff-pom-gefraest-modul-2>
- [67] “Galvanic corrosion,” May 2024. [Online]. Available: <https://galvanizing.org.uk/corrosion/galvanic-corrosion/>
- [68] “Stainless steel definition, composition, types,&facts britannica,” May 2024. [Online]. Available: <https://www.britannica.com/technology/stainless-steel>
- [69] “Kjemisk binding institutt for biovitenskap.” [Online]. Available: <https://www.mn.uio.no/ibv/tjenester/kunnskap/plantefys/kjemi/binding.html>
- [70] “Air composition, oxygen, nitrogen britannica.” [Online]. Available: <https://www.britannica.com/science/air>
- [71] “Noble metal definition, list, & facts britannica.” [Online]. Available: <https://www.britannica.com/science/noble-metal>
- [72] “What are valence electrons?” May 2024. [Online]. Available: <https://chemistrytalk.org/what-are-valence-electrons/>
- [73] “Periodic table - ptable,” May 2024. [Online]. Available: <https://ptable.com/?lang=en>
- [74] “Galvanic corrosion,” Jan 2022. [Online]. Available: <https://www.ssina.com/education/corrosion/galvanic-corrosion/>
- [75] “Profile 5 20x20 item,” May 2024. [Online]. Available: <https://www.item24.com/en-it/profile-5-20x20-natural-61145?supplyUnit=STUECK&nominalLength=6000&nuten=-1&category=profile-technology>
- [76] “Profile 5 20x20 item,” May 2024. [Online]. Available: <https://www.item24.com/en-it/profile-5-20x20-natural-44804?supplyUnit=STUECK&nominalLength=3000&nuten=-1&category=profile-technology>
- [77] “Angle bracket zn item,” May 2024. [Online]. Available: <https://www.item24.com/en-de/angle-bracket-5-20x20-zn-white-aluminium-similar-to-ral-9006-42503?type=BR5&outerDimensions=2020&category=profile-technology%2Ffastening-technology#technical-data>
- [78] “Angle bracket item,” May 2024. [Online]. Available: <https://www.item24.com/en-de/angle-bracket-5-20-right-angled-white-aluminium-similar-to-ral-9006-67777?type=BR5&material=SintSt&category=profile-technology%2Ffastening-technology>

- [79] “Hinge st item,” May 2024. [Online]. Available: <https://www.item24.com/en-de/hinge-s-t-white-aluminium-similar-to-ral-9006-64947?color=11&category=profile-technology%2Fhinges#technical-data>
- [80] “Handle pi item,” May 2024. [Online]. Available: <https://www.item24.com/en-de/handle-pi-80-m5-pa-grey-67907?thread=5&color=7&category=profile-technology%2Fgrips-locks-and-catches>
- [81] “What is the difference between screw pitch and lead?” [Online]. Available: <https://www.thomsonlinear.com/en/support/tips/difference-between-screw-pitch-and-lead>
- [82] “Linear guides (linear motion guides) design and selection | thk official web site [japan/english],” May 2024. [Online]. Available: <https://www.thk.com/opm/jp/en/linear/thklinearguide/>
- [83] “How does belt driven linear motion systems work?” May 2024. [Online]. Available: <https://www.limonrobot.com/how-does-belt-driven-linear-motion-systems-work.html>
- [84] “Belt drive or lead screw? the answer is in the application.” Feb 2020. [Online]. Available: <https://pbclinear.com/blog/2020/february/lead-screw-or-belt-drives>
- [85] “Basic belt-driven actuator with stepper motor,” May 2024. [Online]. Available: <https://www.igus.com/product/20374>
- [86] n. null, *Metric Coarse Thread Data - Newman Tools*, May 2024. [Online]. Available: <https://www.newmantools.com/tech/threadm.htm>
- [87] MISUMI, *Retaining Rings / External / C-Type from MISUMI MISUMI*. [Online]. Available: [https://uk.misumi-ec.com/vona2/detail/110300258330/?rid=cat\\_&CategorySpec=](https://uk.misumi-ec.com/vona2/detail/110300258330/?rid=cat_&CategorySpec=)
- [88] NSK, *Deep groove ball bearings / single row / cover selectable / outer ring selectable / material selectable / tolerance selectable / NSK1-100 Pieces Per Package from NSK MISUMI*. [Online]. Available: <https://uk.misumi-ec.com/vona2/detail/221000058301/?searchFlow=results2products&KWSearch=ball+bearing>
- [89] R. Lorentsen, K. Michelsen, and S. Seljevoll, *Verksted håndboken*. Gyldendal: Hartvig Hartvigsen.
- [90] “iglide g300, sleeve bearing with flange, mm.” [Online]. Available: <https://www.igus.com/product/64>
- [91] “iglide® j, sleeve bearing, mm.” [Online]. Available: <https://www.igus.com/product/3>
- [92] “iglide® g300, thrust washer, mm,” May 2024. [Online]. Available: <https://www.igus.com/product/125>

- [93] “Carbon fibre sheet; 1mm, 2mm, 3mm - easy composites,” May 2024. [Online]. Available: <https://www.easycomposites.co.uk/high-strength-carbon-fibre-sheet>
- [94] “igubal rod end bearing with female thread | igus,” May 2024. [Online]. Available: [https://www.igus.com/product/igubal\\_KBRM\\_KBLM?artnr=KBLM-02](https://www.igus.com/product/igubal_KBRM_KBLM?artnr=KBLM-02)
- [95] “iglide g300, sleeve bearing with flange, mm,” May 2024. [Online]. Available: <https://www.igus.com/product/64>
- [96] “iglide j, sleeve bearing, mm,” May 2024. [Online]. Available: <https://www.igus.com/product/3>
- [97] I. P. Volt, *Advantages of Nylon*, May 2024. [Online]. Available: <https://voltplastics.com/about-us/news-and-articles-page/66/advantages-of-nylon>
- [98] *iglidur plain bearings made of high performance plastics*, May 2024. [Online]. Available: <https://www.igus.eu/info/plain-bearings>
- [99] *Vacuum Pressure Basics*, May 2024. [Online]. Available: <https://www.mks.com/n/vacuum-basics>
- [100] *iglide G300, thrust washer, mm*. [Online]. Available: <https://www.igus.com/product/125>
- [101] *iglide J, sleeve bearing with flange, mm*, May 2024. [Online]. Available: <https://www.igus.com/product/66>
- [102] GANTER and MISUMI, *Spring cotter pins, Steel from GANTER MISUMI*, May 2024. [Online]. Available: <https://uk.misumi-ec.com/vona2/detail/221006469134/?KWSearch=cotter%20pin&searchFlow=results2products>
- [103] MISUMI, *Retaining Rings / External / E-Type from MISUMI MISUMI*, May 2024. [Online]. Available: <https://uk.misumi-ec.com/vona2/detail/110300258420/?KWSearch=e%20type%20retaining%20ring&searchFlow=results2products>
- [104] Composites Knowledge Network. (Year Accessed) KPC/A100. Accessed: Date Accessed. [Online]. Available: <https://compositeskn.org/KPC/A100>
- [105] “Polymer matrix - an overview,” May 2024. [Online]. Available: <https://www.sciencedirect.com/topics/materials-science/polymer-matrix>
- [106] “Exploring various matrix composites.” [Online]. Available: <https://masrath-sultana.medium.com/exploring-various-matrix-composites-3719e4af7d1c>
- [107] “Additives & fillers - composite materials,” May 2024. [Online]. Available: <https://compositeslab.com/composite-materials/additives-fillers/index.html>

- [108] “4.1.8. general laminate design guidelines.” [Online]. Available: <https://www.abbottaeospace.com/aa-sb-001/4-materials/4-1-composite-materials/4-1-8-general-laminate-design-guidelines/>
- [109] “Quick step-by-step guide for composites design,” May 2024. [Online]. Available: <https://www.addcomposites.com/post/best-step-by-step-guide-for-composites-design>
- [110] “Properties of composites,” May 2024. [Online]. Available: <https://www.composites.co.nz/properties-of-composites.html>
- [111] “Fundamentals of composite materials - a100,” May 2024. [Online]. Available: <https://compositeskn.org/KPC/A100>
- [112] “Astm d3039: tensile test on composites.” [Online]. Available: <https://www.zwickroell.com/industries/composites/astm-d3039-tensile-test-on-composites/>
- [113] “Composite manufacturing methods.” [Online]. Available: <https://explorecomposites.com/articles/design-for-composites/basics-manufacturing-methods/>
- [114] “What adhesive types are best for bonding composites?” May 2024. [Online]. Available: <https://www.permabond.com/resource-center/adhesive-types-bonding-composites/>
- [115] “How photocouplers / optocouplers are used,” May 2024. [Online]. Available: <https://www.renesas.com/us/en/products/interface/optoelectronics/how-photocouplers-optocouplers-are-used>
- [116] “Switching power supply: Uses advantages and working principle.” [Online]. Available: <https://www.monolithicpower.com/en/switching-power-supply>
- [117] A. Abacus, “Understanding switched-mode power supplies (smps).” [Online]. Available: <https://my.avnet.com/abacus/solutions/technologies/power/the-design-engineers-guide/switched-mode-power-supplies/>
- [118] “Switch mode power supply topologies: A comparison,” May 2024. [Online]. Available: <https://www.we-online.com/en/news-center/blog?d=switch-mode-power-supply>
- [119] J. null, “A noise-free diy switching power supply - how hard can it be?” vol. 542, May 2022. [Online]. Available: <https://community.element14.com/challenges-projects/element14-presents/project-videos/w/documents/27520/a-noise-free-diy-switching-power-supply---how-hard-can-it-be---episode-542?CMP=SOM-YOUTUBE-PRG-E14PRESENTS-e14Presents-542>
- [120] M. Fortunato, “Ensure long lifetimes from electrolytic capacitors: a case study in led light bulbs,” May 2013. [Online]. Available: <https://www.analog.com/en/resources/technical-articles/ensure-long-lifetimes-from-electrolytic-capacitors-a-case-study-in-led-light-bulbs.html>

- [121] R. Barnes, “Power your raspberry pi: expert advice for a supply,” May 2017. [Online]. Available: <https://magpi.raspberrypi.com/articles/power-supply>
- [122] J. Errington, “skillbank: Experiments and projects with an arduino microcontroller.” [Online]. Available: <http://www.skillbank.co.uk/>
- [123] I. Technologies, “Irf520n datasheet,” [https://www.infineon.com/dgdl/Infineon-IRF520N-Datasheet-v01\\_01-EN.pdf?fileId=5546d462533600a4015355e340711985](https://www.infineon.com/dgdl/Infineon-IRF520N-Datasheet-v01_01-EN.pdf?fileId=5546d462533600a4015355e340711985), accessed: 2024-05-16.
- [124] “Led strips,” May 2024. [Online]. Available: <https://www.instructables.com/LED-Strips/>
- [125] C. Staff, “How a uml use case diagram can benefit any process.” [Online]. Available: <https://nulab.com/learn/software-development/how-a-uml-use-case-diagram-can-benefit-any-process/>
- [126] “What is software architecture.” [Online]. Available: <https://resources.github.com/software-development/what-is-software-architecture/>
- [127] “Welcome to pyserials documentation pyserial 3.4 documentation.” [Online]. Available: <https://pyserial.readthedocs.io/en/latest/>
- [128] “switch...case - arduino reference.” [Online]. Available: <https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/>
- [129] “5 most common arduino nano clone problems and their solutions.” [Online]. Available: <https://minov.in/5-most-common-arduino-nano-clone-problems-and-their-solutions-2/>
- [130] “Explination of a pid controller.” [Online]. Available: <https://pidexplained.com/pid-controller-explained/>
- [131] “Explination of a pid controller.” [Online]. Available: <https://www.watelectronics.com/pid-controller/>
- [132] “Tea assessccus,” May 2024. [Online]. Available: <https://assessccus.globalco2initiative.org/tea/>
- [133] trihedral, “trihedralarduinolatexlisting.” [Online]. Available: <https://github.com/trihedral/ArduinoLatexListing>

## Bibliography

- [Bib1] K. Tangchaichit, *Mechanical Engineering Design*. Richard G. Budynas and J. Keith Nisbett.
- [Bib2] S. Kalpakjian and S. R. Schmid, *Manufacturing, Engineering and Technology*, seventh edition ed. Serope Kalpakjian and Steven R. Schmid.

# **Appendices**

## **Appendix A**

### **Additional explanations figure for the requirement table**

Additional explanations	
Terms	Initials
<b>Priorities:</b>	<b>AK:</b> Adrian Kristiansen
<b>A - Most critical requirements</b>	<b>MO:</b> Marte Overgaard
<b>B - Further potential improvements</b>	<b>SG:</b> Sophus Gjerstad
<b>C - Optional/nice to have</b>	<b>LR:</b> Lorentz Rasmussen
<b>Category:</b>	<b>TM:</b> Theo Magnor
<b>Functional</b> - Defines what the system should do and the functionalities it must provide.	<b>AL:</b> Ask Lindbråten
<b>Non-functional</b> - Defines the criteria that the system as a whole should meet.	
<b>Completion status:</b>	
Not started	
Ongoing	
Testing	
Completed	
<b>Working area:</b>	
Defines the range of movement in all three axes, where the prototypes tool-adapter should be able to move.	
<b>Design area:</b>	
Defines the total range of movement in all three axes we cannot exceed, while designing, mounting and moving the prototype's arm(s).	

## **Appendix B**

### **Requirement table**

**Requirements table**

Downscaled prototype							
Requirement ID	Derived from	Category	Requirement	Priority	Verification ID(s)	Verification method(s)	Verification responsible(s)
R1.0	User story, Task description & Tronrud Engineering	Functional	The prototype <u>shall</u> be able to perform a vertical and horizontal "pick and place" movement in the three coordinate axis-directions (x, y & z).	A	V1.0 V1.1	Verify through the use of a manual control system (joystick, arrow keys or input command(s)). Enable the appropriate movement mechanics through fitting construction techniques.	AL/TM SG/MO/JAK
R2.0	Tronrud Engineering	Non-functional	The prototype <u>must</u> have an incorporated rail system to account for movement on the y-axis.	A	V2.0	Verify the movement through the use of a control system or code.	AL/TM
R3.0	Tronrud Engineering	Functional	The prototype's rail system <u>must</u> have movement capabilities up to 10 cm.	A	V3.0	Verify the motion range by measuring the distance travelled from a reference point on the y-axis.	MO

R4.0	Task description & Tronrud Engineering	Non-functional	The prototype <u>must</u> conform to Tronrud Engineering's material bonding and composite requirements.	A	V4.0	Verify through the use of carbon fiber composites and bonding between carbon fiber and aluminium.	SG/MO/AK
R5.0	Task description, user story & Tronrud Engineering	Non-functional	The prototype <u>must</u> be cost-efficient.	A	V5.0	Verify through the use of TEA (Techno Economic Analysis).	AK Ongoing
R6.0	Tronrud Engineering	Functional	The prototype <u>must</u> be powered with electricity from a socket (Voltage of 220-240).	A	V6.0	Verify that the prototype performs its desired functionalities with power supply from a socket.	LR Ongoing
R7.0	User story & Tronrud Engineering	Non-functional	The prototype <u>must</u> be user-friendly.	A	V7.0	Verify that a manual tool change does not require additional implements.	SG Ongoing
				A	V7.1	Verify that the prototype is controllable using a simple control system.	SG/LATM Ongoing
			Ensure proper cable management in the switch cabinet, and around the rig construction.	LR	V7.2		Ongoing

R8.0	Tronud Engineering	Non-functional	The prototype <u>must</u> be able to be moved and used for exhibition.	V8.0 V8.1 V8.2 V8.3 V8.4
R9.0	Tronud Engineering	Functional	The prototype <u>shall</u> have a manual control system.	V9.0
R10.0	Tronud Engineering	Non-functional	The prototype's working area in height (movement in the z-direction) <u>must</u> have a 20 % (percent) safety margin to account for further design.	V10.0
R11.0	Tronud Engineering	Non-functional	The prototype's potential building area in height <u>must</u> not exceed 600 mm.	V11.0

R12.0	Tronrud Engineering	Non-functional	The prototype must be exhibition-friendly by design.	V12.0 V12.1 V12.2	Create the rig construction using carbon fiber or aluminium T - slot. Install three windows of plexiglass on the rig construction for spectators to easily view the system, while adhering to spectator safety. Implement our logo on the rig construction.
R13.0	Tronrud Engineering	Non-functional	The prototype <u>must</u> have an emergency procedure.	A V13.0	Implement an emergency stop button on the rig construction.
R14.0	User story & Tronrud Engineering	Functional	The prototype should achieve performance parameters.	B V14.0 V14.1	Perform physical stress tests, and compare them to FEM (Finite element method) simulations. Verify that the prototype are able to perform 15 "pick and place" movements in 60 seconds.
R15.0	User story	Non-functional	The prototype should have an option of mounting different gripping tools.	B V15.0	Verify through affixing a universal adapter implement that facilitates the mounting of different tools sharing identical attachment points.

R16.0	Non-functional	The rail system should be visible from above.	B	V16.0	Verify through mounting plexiglass.  MO Ongoing
R17.0	User story	The prototype <u>should</u> have a tool attached to the implement adapter.	C	V17.0	Attach a magnet or hook to the tool adapter.  SG Not started

## **Appendix C**

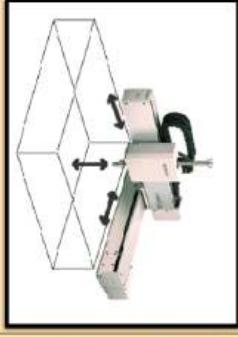
### **Project timeline**

## Project timeline

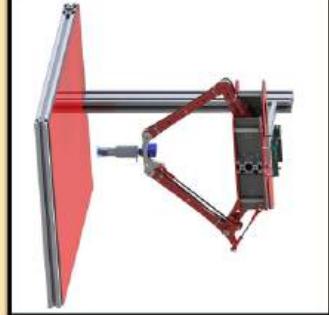
## **Appendix D**

### **Concept examination**

## Concept examination

Technology concept	Visualization	General description	Advantages	Disadvantages
Articulated robots		<p>A type of industrial robot that's designed as a human arm, composed of various segments connected by multiple joints.</p>	<p><b>Versatile:</b> Multiple joints and degrees of freedom, gives this type of robot the ability to complete complicated and intricate tasks like welding, packaging, surgical procedures, animaltronics and agricultural processes.</p> <p><b>Range of environments:</b> Multiple joints and degrees of freedom also offer a greater range of motion and more flexibility, making it applicable in a wider range of environments, including areas where spatial orientation can be challenging.</p> <p><b>Precision:</b> These robots are capable of precise positioning, which is crucial for tasks that require high levels of accuracy.</p>	<p><b>Sophisticated programming:</b> As a result of having to tweak multiple degrees of freedom to perform intricate movements and complex tasks, this type of robots may require severely advanced programming.</p> <p><b>Larger cost:</b> At initial investment, these types of robots are more expensive than ordinary pick-and-place robots, but they can give a more favorable ROI (return on investment) over a longer period of time due to the broader range of applications.</p>
Cartesian robots		<p>A type of linear system, that works as an alternative to articulated robots. It moves in a straight line along a rail system and consists of multiple axes that run perpendicular to each other, which creates a cubic or rectangular work area.</p>	<p><b>Configuration options:</b> These robots can be configured with either 2 or 3 axes (for instance an X, y, x, z- or x,y,z setup).</p> <p><b>Repetitive tasks:</b> They are normally used for repetitive pick-and-place tasks, as well as assembly and dispensing, where the required traveling distance is <math>\leq</math> (less-than) 1 meter.</p> <p><b>Simplicity:</b> Due to the simplified linear movements, the required programming becomes significantly easier compared to articulated robots. Additionally, the work area also becomes well defined, which in turn leads to two advantages. Firstly, it makes it easier to design and implement safety measures around the robot's workplace, and secondly, it makes it simpler to precisely position and orient parts within the robot's operating space.</p>	<p><b>Payload capacity:</b> Because of the fact that these robots only use one of their outer axes to support workloads, the payload capacity may be slightly reduced.</p>

<p><b>Axes:</b> Gantry robots use all three axes on their grid system (x, y &amp; z), and differ from Cartesian robots in that they utilize two x-axes, or sometimes two y- or z axes. As a result, this type of robot can achieve increased levels of productivity.</p> <p><b>Centralized workload placement:</b> The area where tasks are performed or payloads are manipulated is concentrated within the physical space accompanied by the actual robot. This in turn maximises the robot's efficiency, since it doesn't have to travel large distances to access different parts of the workspace. Additionally, this also helps to overcome the payload limitations of the Cartesian robot.</p> <p><b>Applicability:</b> These robots are ideal for repetitive processes that require precise linear movements and a traveling distance &gt; (greater than) 1 meter, like part transfers, palletizing, picking, machine loading and assembly for instance.</p>	 <p>Gantry robots</p>
<p><b>Speed:</b> They are faster than most other types. About 30% faster than Scala robots. Their fast operating speeds comes from the three axis design, since the axes are connected mechanically in parallel opposed to axes joined in series. A parallel-link design allows faster movement, acceleration and higher duty cycles. The speed also comes from the weight distribution of the motors which makes the weight stationary, allowing for lightweight arms and quick response time.</p> <p><b>Productivity:</b> Their high productivity rate is due to their exceptional speed and acceleration. This means that the delta robot can process more parts per minute than most other robots in a safe and effective manner.</p> <p><b>Footprint:</b> Delta robots takes up a relative small space (Depending on arm movement in z-direction) and are typically mounted overhead. This makes the robot suitable for places with limited space and easy to integrate into a system.</p> <p><b>Applications:</b> Rapid pick and place applications like in a packaging machine or even soldering works. There are many different applications for a delta arm as long as it can be mounted overhead.</p>	<p><b>Flexibility:</b> Because of the robots simplified linear movements, it may be less suitable for more intricate tasks that need more complex motions. Additionally, it also has reduced range of motion compared to articulated arms.</p> <p><b>Setup:</b> Due to how the robot's constructed it needs to be mounted to a ceiling to operate properly.</p>
<p><b>Speed:</b> They are faster than most other types. About 30% faster than Scala robots. Their fast operating speeds comes from the three axis design, since the axes are connected mechanically in parallel opposed to axes joined in series. A parallel-link design allows faster movement, acceleration and higher duty cycles. The speed also comes from the weight distribution of the motors which makes the weight stationary, allowing for lightweight arms and quick response time.</p> <p><b>Productivity:</b> Their high productivity rate is due to their exceptional speed and acceleration. This means that the delta robot can process more parts per minute than most other robots in a safe and effective manner.</p> <p><b>Footprint:</b> Delta robots takes up a relative small space (Depending on arm movement in z-direction) and are typically mounted overhead. This makes the robot suitable for places with limited space and easy to integrate into a system.</p> <p><b>Complexity:</b> Ensuring that every part fit so that the robot can work as intended (length of arms, movement, etc).</p> <p><b>Programming:</b> Due to the robots complex movements in three axis the code and testing is more demanding than other types.</p>	

 <p><b>Duopod robot</b></p>	<p><b>Speed:</b> They are faster than most other types. About 30% faster than Scara robots. Their fast operating speeds comes from the design, since the axes are connected mechanically in parallel opposed to axes joined in series. A parallel-link design allows faster movement, acceleration and higher duty cycles. The speed also comes from the weight distribution of the motors which makes the weight stationary, allowing for lightweight arms and quick response time.</p> <p><b>Productivity:</b> Their high productivity rate is due to their exceptional speed and acceleration. This means that the Duopod can process more parts per minute than most other robots in a safe and effective manner.</p> <p><b>Footprint:</b> Duopod robots takes up a relative small space (Depending on arm movement in z-direction) and are typically mounted overhead. This makes the robot suitable for places with limited space and easy to integrate into a system.</p> <p><b>Applications:</b> Rapid pick and place applications like in a packaging machine or even soldering works. There are many different applications for a Duopod as long as it can be mounted overhead.</p>	<p><b>Loads:</b> The duopod robot can't carry too heavy loads due to their design and workarea. The further away from center position the less weight can be carried.</p> <p><b>Complexity:</b> Ensuring that every part fit so that the robot can work as intended (Length of arms, movement, etc).</p> <p><b>Programming:</b> Due to the robots complex movements in three axis the code and testing is more demanding than other types.</p>
 <p><b>SCARA robot</b></p>	<p><b>Cost:</b> Due to their fewer axes of motion and smaller size, less materials and components are required to build SCARA robots, making them considerably more affordable when compared to other types of robots.</p> <p><b>Setup:</b> These robots can be mounted using both pedestals, floors and walls, making them more applicable for various work environment setups.</p> <p><b>Axis configuration:</b> Four axis configuration enables them to move along the x, y and z coordinate system, while also permitting 360 degrees of rotational movement around the z - axis, resulting in a cylindrical work envelope.</p> <p><b>High-speed and accuracy:</b> Due to their number of axes, small footprint and fewer joints to control, these robots are well known for achieving greater speeds compared to both articulated and cartesian robots. They are also particularly suited for tasks that require precise and repeatable movements, such as small-part assembly lines (for example electronics etc).</p>	<p><b>Payload capacity:</b> These robots generally have low payload capabilities ranging from between 0.5 to 20 kg.</p> <p><b>Range of motion- and workspace size:</b> Due to the fact that their arm is designed to move primarily in the horizontal plane, they have a limited range of motion in the vertical plane. They are also less flexible than 6 - axis robots for instance, and are confined to a relatively small workspace with limited task manipulation abilities, typically restricted to tasks that require &lt;=1.0 meter reaches. These limitations makes them unsuitable for tasks that require more flexibility and complexity, a large range of vertical motion and/or adaptability.</p>

**Cost - effectiveness:**

This type of robot has few parts resulting in reduced expenditures for manufacturers. The simpler design also cuts down on points of failure, which in turn could reduce the need for maintenance fees. In contrast, articulated robots for instance, would need regular check-ups and lubrication due to their multiple axes and moving parts.

Cylindrical robots, with fewer motor-components, often consume less power compared to other types as well, potentially reducing the ongoing operational cost.

**Compact design:**

These robots offer a compact design, which is particularly advantageous for businesses looking to make the most out of limited space. Due to its vertical orientation, this type also minimizes its footprint, allowing it to fit into areas where machines and workstations are closely packed, such as assembly lines or small scale manufacturing units. In addition, the vertical reach ensures interaction with multiple shelves without needing extra floor space.

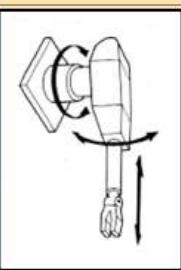
**Versatility within their operational envelope:**

Within their operational envelope, cylindrical robots are highly versatile. They are often found in settings ranging from manufacturing to medical environments, and is designed to withstand various conditions, such as high temperatures, moisture, or the presence of corrosive materials.



Cylindrical robot

This is a type of industrial robot built with joints that allow for rotational motion in two directions and linear motion along one axis.



Polar/Spherical robot

**Lifting/loading capacity:**

Due to the design and having a robust linear arm, this type of robot carry and load more weight than other types.

**Size:**

The robot has a large footprint and takes up a lot of space.

**Versatility:**

Polar and spherical robots are not very versatile and has a low vertical reach.

**Cost:**

The cost of these robot types are usually high and more difficult to maintain, due to their semi complex design.

**Limited range of motion:**

Although cylindrical robots are versatile within their work envelope, they are still limited in terms of adaptability to more complex tasks compared to articulated robots. The restricted range of motion often results in longer cycle times as well, especially for tasks that require multiple steps. This limitation can lead to inefficiencies in the production process and arises because they cannot move as freely.

**Speed constraints:**

By design, these robots have slower operational speeds compared to some other robot types. Their two rotational and one linear axis system does not allow for rapid movements, so for industries that revolve around high-volume packaging or sorting, cylindrical robots might not be the most optimal choice.

**Dependency on external systems:**

Cylindrical robots, especially those tailored to specific applications, may require additional external components like specialized sensors or supplementary software systems. This need might complicate the integration process and drive up operational expenses, despite that the initial cost may be lower than other robot types.

**Maintenance concerns:**

Although these robots might require less maintenance overall compared to other robot types, as mentioned in the cost-effectiveness pro, certain parts of cylindrical robots, especially the rotary joints might experience occasional wear and degradation. Over time, they may require recalibration and periodic software updates to optimize performance and ensure desired precision levels as well.

**Reach:**

Since the robot has the opportunity of having a long linear arm the reach in the horizontal plane is exceptional compared to other types.

**Applicability:**

These types of robots are usually used in application like machine tool loading, material movement, component stacking, forging, casting and welding.

<p><b>Flexible.</b> Their small size makes them very flexible and easy to assemble. It can easily be disassembled and moved around to different locations without changing the layout.</p> <p><b>Safe:</b> Cobots don't need a dedicated work space with fences, since the robot's joints are force limited. This means that every joint is equipped with force sensors which adds a fast retractable reaction in case of contact and making the robot stop. It can also be fitted with laser sensors that slows down or makes the robot stop when approached by an individual.</p> <p><b>Collaborative robot</b></p>  <p><b>Articulated robots.</b> These are designed to safely interact with humans within a shared workspace.</p> <p><b>Can be two-armed:</b> A second arm can be added and this can be very useful in more delicate tasks like assembly of small parts or tightening screws. The result is an increase in productivity and flexibility.</p> <p><b>Applicability:</b> It can be applicable to various operations like screwing, assembly of smaller parts, polishing, pick and place, and more.</p>	<p><b>Loads:</b> The flexibility comes at a cost, so the cobots can not do heavy-duty tasks.</p> <p><b>Speed:</b> The speed is generally limited due to the safety procedures main priority.</p> <p><b>Independent:</b> A draw back to being a helping hand is that the robot is not fully independent in its work.</p> <p><b>Safety approval:</b> The safety approval of cobots can be very troublesome due to different regulations. In addition, if moved or changed tool the robot needs a new safety certification with new documentation and CE-marking.</p>	

## **Appendix E**

### **Rapid Risk Analysis - reporting form**

Risk ID	Category	Description of risk	Cause of scenario	Mitigation measures	Level of consequence				Level of risk				Level of modified consequence				Level of modified risk			
					Other people	Environment	Materialistic property	Project group	Tronrud Engineering	Level of probability	People	Environment	Materialistic property	Project group	Tronrud Engineering	Other people	Environment	Materialistic property	Project group	Tronrud Engineering
R11.0	Societal	Not reaching milestones	Underestimating the time needed to conduct specific work efforts.	Using daily scrum and adhere to the project timeline as best as possible.	N/A	N/A	N/A	3	2	C	N/A	N/A	H	M	N/A	N/A	2	1	B	N/A
R12.0	Technical	Not satisfying "A" priority requirements	Not fulfilling every verification method for the specific requirement, due to lack of time, effort and/or not designing and developing in accordance with the most fundamental requirements.	Take time to properly schedule tests, document verifications, and design and develop in compliance with the fundamental requirements.	N/A	N/A	N/A	4	2	C	N/A	N/A	H	M	N/A	N/A	2	1	A	N/A
R13.0	Societal	Loss of work and documentation	Encountering sudden storage corruption and/or accidentally deleting files.	Perform continuous backups of work efforts, be careful and considerate when pushing and deleting anyones previous work, and make sure the latest changes to project files are correctly saved.	N/A	N/A	N/A	4	3	B	N/A	N/A	H	H	N/A	N/A	2	1	A	N/A
R14.0	Societal	Misunderstanding the project framework	Insufficient effort to familiarize oneself with the chosen project model or establishing a framework that is overly complex.	Establish a framework that maintains simplicity while upholding the desired principles, and ensure that all members understand the reasons for why the particular model was chosen and how it will be enacted in practice.	N/A	N/A	N/A	2	N/A	B	N/A	N/A	L	N/A	N/A	N/A	1	N/A	A	N/A
R15.0	Societal	Severe disagreement or arguments within the group	Having different perspectives, goals, objectives and/or past conflicts.	Clarify potential misunderstandings among group members immediately or with the specific person concerned.	N/A	N/A	N/A	3	N/A	B	N/A	N/A	M	N/A	N/A	N/A	2	N/A	A	N/A
R16.0	Societal	Unsatisfactory fulfillment of the expected documentation level	None efficient prioritizing, or neglecting the importance of documentation.	Work to ensure that each member meet documentation deadlines and internal requirements, and contributes with descriptions of their technical efforts.	N/A	N/A	N/A	3	3	C	N/A	N/A	M	M	N/A	N/A	1	N/A	B	N/A
R17.0	Technical	Not satisfying the task description	Not working towards the overarching goal or wrongfully prioritizing of technical efforts.	Ensure that every discipline conduct technical efforts to work towards the common overarching goal.	N/A	N/A	N/A	3	3	B	N/A	N/A	M	M	N/A	N/A	2	1	A	N/A
R18.0	Societal	Using too much time on discussions	Not handling arguments or discussions properly, and inefficient time use on irrelevant matters.	End discussions quickly using joint voting, and/or put members back on the right track.	N/A	N/A	N/A	1	N/A	E	N/A	N/A	M	N/A	N/A	N/A	2	1	A	N/A
R19.0	Technical	Not taking into account the galvanic voltage series when choosing materials	Time-constraints, inadequate knowledge or a lack of dedicated effort.	Actively utilize the galvanic voltage series during mechanical design.	N/A	N/A	N/A	4	3	A	N/A	N/A	M	M	N/A	N/A	1	1	A	N/A
R10.0	Technical	Motor failure	Unlucky damage or wrongful operation of the motors.	Take preliminary measures to safely operate the motors, and construct mounting option that ensures stable operation.	N/A	N/A	N/A	4	2	A	N/A	N/A	M	L	N/A	N/A	2	1	A	N/A
R11.0	Technical	Incorrect assembly of parts and components	Misalignment between members' understanding of the project.	Ensure that every member remain on the same page.	N/A	N/A	N/A	3	2	B	N/A	N/A	M	L	N/A	N/A	1	1	A	N/A
R12.0	Technical	To weak electrical engines	Inadequate research into the choices of hardware.	Implement alternative solutions, such as gearing and/or downscaling to handle the issue.	N/A	N/A	N/A	4	2	A	N/A	N/A	M	L	N/A	N/A	2	1	A	N/A

## **Appendix F**

### **Individual pugh matrix - Delta robot**

Individual evaluation				
Delta robot				
Criteria	Weight	Score	Rating	Comment
Cost	3	2	6	Three motors, Carbon fiber and complex design(more moving parts and many small parts)
Assembly	2	1	2	Many smaller parts. This makes it hard to produce and assemble
Robust	2	2	4	Locked in place, good stability and low wear due to motors, but have a lot of joints
Accuracy	1	3	3	Delta robots are known for their high accuracy. It can move freely in three axes.
Maintenance	2	2	4	Due to many small and moving parts, performing maintenance can be a hassle. The delta arm is also a delicate construction.
Workspace	1	2	2	Workspace is restricted as the arms consume a big area while moving effector.
System feasibility (team)	3	2	6	Coding 3 motors working in 3 axis is quite challenging. Mechanically the delta arm is complex, these two points make it less feasible.
Speed	3	3	9	Because the motors are connected at the fixed base, the delta has lightweight, thin arms that can maneuver at high speeds.
Reliability	3	2	6	Due to lower weight on the arms, there will be less wear during operation, which increases reliability. There are many small parts that could fail, this decreases reliability.
Lifting capacity	2	1	2	The delta arm does not have the best lifting capacity due to the length and robustness of the arms.
Footprint	3	2	6	The delta arm takes up a lot of space, as the arms have to move further up than the mounting point when lifting to the highest point.
		Total	50	

## **Appendix G**

### **Individual pugh matrix - Duopod on rails**

## Individual evaluation

### Duopod on rails

Criteria	Weight	Score	Rating	Comment
<b>Cost</b>	3	2	6	2 motors for the arm, 1 motor for the rail.(noe mere og skrive her?). The rail motor shouldn't need to be as powerful as the two on the arm.
<b>Assembly</b>	2	2	4	Assembling the total system could be time-demanding due to the mounting of rails, and then assembling the robot itself.
<b>Robust</b>	2	3	6	Duopod-like design is sturdy due to two arms joining at one point. The arm works biaxially, making it more robust. It is manageable to make a solid connection interface rail/arm, to prevent sway.
<b>Accuracy</b>	1	3	3	Having only two axis makes it relatively simple to achieve high accuracy with a duopod.
<b>Maintenance</b>	2	1	2	Maintenance on a duopod could be simple, due to the biaxial design of the arm. The rail connection could cause higher maintenance, but this depends on the design.
<b>Workspace</b>	1	3	3	The duopod on rails should be able to work in the defined space whilst having a simple and effective design.
<b>System feasibility (team)</b>	3	2	6	The arm only works in two axis with two motors, this simplifies development and design for software, electrical and mechanical. This makes the system more feasible.
<b>Speed</b>	3	3	9	Motors are placed at the top of the arm, this makes the actual arms have low mass, this translates to fast and snappy movement. This design is known for providing a frame capable of high speeds.
<b>Reliability</b>	3	3	9	The arm has few moving parts making it more reliable. The rail is very reliable of performing the linear task from A to B. In addition, only occasionally performing this task will result in mild wear on the rails.
<b>Lifting capacity</b>	2	2	4	The duopod does not have the best lifting capacity due to the length and robustness of the arms.
<b>Footprint</b>	3	1	3	The duopod has a large footprint as it takes up quite some space with the rail. The two arms have to move above and on the sides of the mounting point of the robot.
<b>Total</b>		55		

## **Appendix H**

**Individual pugh matrix - Two  
directional articulated robot arm on  
rails**

## Individual evaluation

### Two directional articulated robot arm on rails

Criteria	Weight	Score	Rating	Comment
Cost	3	1	3	This concept need 3 motors in the arm and 1 for the rail, making it quite expensive.
Assembly	2	1	2	Assembly would be complex due to high motor count and having three joints and rails.
Robust	2	3	6	Due to the mechanical sturdy design and use of one single strong arm, articulated robots are highly robust.
Accuracy	1	3	3	The arm is comprised of many joints giving it 3 degrees of freedom, theoreticly this allows for high accuracy, but from a coding perspective its harder to achieve.
Maintenance	2	1	2	More motors means more frequent maintenance. High torque and load on the top joint will result in more maintenance.
Workspace	1	3	3	The concept can perform movement in the desired workspace. The arm is responsible for X- and Z-directions and the rail is responsible for the Y-direction.
System feasibility (team)	3	1	3	By no means impossible to design and produce, but it will be more challenging due to the number of active joints.
Speed	3	2	6	Slower speed as the top motor has to do lots of the motion.
Reliability	3	2	6	The mechanical and electrical components are encapsulated within the movable parts of the arm resulting in wear, decreasing the reliability over time. The rail is very reliable of performing the linear task from A to B in Y-direction. In addition, only occasionally performing this task will result in mild wear on the rails.
Lifting capacity	2	3	6	Articulated robots have a mechanical advantage due to their jointed arms which makes them sturdy, stable and allows them to handle high loads.
Footprint	3	2	6	The robot has a low footprint with one arm, but due to the rails it takes up more space.
		Total	46	

## **Appendix I**

### **Individual pugh matrix - Criteria description**

# Criteria explanation

## Description

How much does it cost to produce, the cheaper gives better score.

The simplicity of assembly process.

The robustness of the mechanical construction. Strength, stability, fatigue.

The accuracy and precision of movements.

The amount of maintenance needed and how easy is it performed.

The reach of the tools center point.

The feasibility of the designing and producing the robot within the given timeframe.

The speed of which the robot can operate.

The certainty of it to do a task without failure.

The weight able to be lifted by the concept.

The space taken up by the machine during the operation. The less the better

## **Appendix J**

**Results from the initial dynamic  
payload capacity tests without gearing**

		Color coding	Description
			The Junior model is able to comfortably complete the motion with the belonging additional mass at the specific arm length
			The Junior model is able to complete the motion with the belonging additional mass at the specific arm length, but visually the motion is not continuous
			The Junior model is either not able to complete the motion with the belonging additional mass at the specific arm length, or it manages a set of repetitions before failing and entering emergency mode
Range of degrees interval	[0, 50>-><50, 0]		<input type="checkbox"/> Check mark to specify the comfortable lifting capacity at arm lengths: 122 and 220 mm

Mass (m) [Kg]	Arm (L) [m]	Gravity (g) [m/s^2]	Force (F) [N]	Torque (T) [Nm]	Torque (T) [Kgcm]	Comment (when needed)
0,05	0,083	9,81	0,4905	0,0407115	0,415141679	
0,05	0,095	9,81	0,4905	0,0465975	0,475162163	
0,05	0,108	9,81	0,4905	0,052974	0,540184354	
0,05	0,122	9,81	0,4905	0,059841	0,610208252	
0,05	0,13	9,81	0,4905	0,063765	0,650221907	
0,051	0,22	9,81	0,50031	0,1100682	1,122383046	

0,103	0,083	9,81	1,01043	0,08386569	0,855191859	
0,103	0,095	9,81	1,01043	0,09599085	0,978834056	
0,103	0,108	9,81	1,01043	0,10912644	1,112779769	
0,103	0,122	9,81	1,01043	0,12327246	1,257028998	
0,103	0,13	9,81	1,01043	0,1313559	1,339457129	
0,101	0,22	9,81	0,99081	0,2179782	2,222758582	

0,151	0,083	9,81	1,48131	0,12294873	1,253727872	
0,151	0,095	9,81	1,48131	0,14072445	1,434989733	
0,151	0,108	9,81	1,48131	0,15998148	1,631356749	
0,151	0,122	9,81	1,48131	0,18071982	1,84282892	
0,151	0,13	9,81	1,48131	0,1925703	1,96367016	
0,1502	0,22	9,81	1,473462	0,32416164	3,305528109	

0,203	0,083	9,81	1,99143	0,16528869	1,685475218	
0,203	0,095	9,81	1,99143	0,18918585	1,929158382	
0,203	0,108	9,81	1,99143	0,21507444	2,193148477	
0,203	0,122	9,81	1,99143	0,24295446	2,477445501	
0,203	0,13	9,81	1,99143	0,2588859	2,639900944	
0,202	0,22	9,81	1,98162	0,4359564	4,445517164	The additional mass was lifted to a degree equivalent to the middle ground between yellow and red grading

0,253	0,083	9,81	2,48193	0,20600019	2,100616897	
0,253	0,095	9,81	2,48193	0,23578335	2,404320545	
0,253	0,108	9,81	2,48193	0,26804844	2,73333283	
0,253	0,122	9,81	2,48193	0,30279546	3,087653753	
0,253	0,13	9,81	2,48193	0,3226509	3,290122851	
0,255	0,22	9,81	2,50155	0,550341	5,611915232	The additional mass was lifted to a degree equivalent to the borderline between yellow and red grading

0,306	0,083	9,81	3,00186	0,24915438	2,540667078	
0,306	0,095	9,81	3,00186	0,2851767	2,907992438	
0,306	0,108	9,81	3,00186	0,32420088	3,305928246	
0,306	0,122	9,81	3,00186	0,36622692	3,7344745	
0,306	0,13	9,81	3,00186	0,3902418	3,979358073	
0,296	0,22	9,81	2,90376	0,6388272	6,514223171	Instant failure

0,276	0,083	9,81	2,70756	0,22472748	2,29158207	
0,276	0,095	9,81	2,70756	0,2572182	2,62289514	
0,276	0,108	9,81	2,70756	0,29241648	2,981817633	
0,276	0,122	9,81	2,70756	0,33032232	3,368349549	
0,276	0,13	9,81	2,70756	0,3519828	3,589224929	
0,276	0,22	9,81	2,70756	0,5956632	6,074072957	Instant failure

0,266	0,083	9,81	2,60946	0,21658518	2,208553734	
0,266	0,095	9,81	2,60946	0,2478987	2,527862708	
0,266	0,108	9,81	2,60946	0,28182168	2,873780762	
0,266	0,122	9,81	2,60946	0,31835412	3,246307898	
0,266	0,13	9,81	2,60946	0,3392298	3,459180547	
	0,22					The Junior model managed 4-5 repetitions of the motion, but failed shortly after
0,266	0,22	9,81	2,60946	0,5740812	5,853997849	

0,356	0,083	9,81	3,49236	0,28986588	2,955808757	
0,356	0,095	9,81	3,49236	0,3317742	3,383154601	
0,356	0,108	9,81	3,49236	0,37717488	3,846112599	
0,356	0,122	9,81	3,49236	0,42606792	4,344682751	
0,356	0,13	9,81	3,49236	0,4540068	4,629579981	
0,356	0,22	9,81	3,49236	0,7683192	7,834673813	

0,404	0,083	9,81	3,96324	0,32894892	3,354344769	
0,404	0,095	9,81	3,96324	0,3765078	3,839310278	
0,404	0,108	9,81	3,96324	0,42802992	4,364689579	
0,404	0,122	9,81	3,96324	0,48351528	4,930482673	
0,404	0,13	9,81	3,96324	0,5152212	5,253793012	
0,404	0,22	9,81	3,96324	0,8719128	8,891034328	

0,457	0,083	9,81	4,48317	0,37210311	3,794394949	
0,457	0,095	9,81	4,48317	0,42590115	4,342982171	
0,457	0,108	9,81	4,48317	0,48418236	4,937284994	
0,457	0,122	9,81	4,48317	0,54694674	5,577303419	The junior model lifted the additional mass to a degree just above the borderline between yellow and red grading. It managed 10 repetitions but failed shortly after
0,457	0,13	9,81	4,48317	0,5828121	5,943028234	
0,457	0,22	9,81	4,48317	0,9862974	10,0574324	

0,507	0,083	9,81	4,97367	0,41281461	4,209536629	
0,507	0,095	9,81	4,97367	0,47249865	4,818144334	
0,507	0,108	9,81	4,97367	0,53715636	5,477469348	The additional mass was lifted to a degree equivalent to the borderline between yellow and red grading
0,507	0,122	9,81	4,97367	0,60678774	6,187511671	Instant failure
0,507	0,13	9,81	4,97367	0,64657771	6,593250141	
0,507	0,22	9,81	4,97367	1,0942074	11,15780793	

0,544	0,083	9,81	5,33664	0,44294112	4,516741471	
0,544	0,095	9,81	5,33664	0,5069808	5,168764335	The additional mass was lifted to a degree equivalent to the borderline between yellow and red grading
0,544	0,108	9,81	5,33664	0,57635712	5,87720577	
0,544	0,122	9,81	5,33664	0,65107008	6,639065777	Instant failure
0,544	0,13	9,81	5,33664	0,6937632	7,074414353	
0,544	0,22	9,81	5,33664	1,1740608	11,97208583	

LSS HS1 - dynamic payload capacity test without gearing		Range of degrees intervals				Verification (Code)	Verification (Physical)
Additional mass (m) [Kg]	Maximum speed RPM	[(90,-90),(0,0)]	[(0,0),(63,750,9)]	[(63,750,9),(0,0)]	[(0,0),(90,-90)]		
0,32345	5					No	No
	10					Yes	Yes
0,4219	Default					Yes	Yes
	5					No	No
0,3994	10					No	No
	Default					No	No
	5					No	No
	10					Yes	Yes
	25 (Visually better than default speed)					Yes	Yes
	Default					Yes	Yes

## Total mass during test

Description	Mass (gram)
One active and passive arm with nuts, bolts and connections	112
One active and passive arm including parallel arm with toolhub, nuts, bolts and connections	165
Large metalplate nr 1	138
Large metalplate nr 2	161
Two small metalplates with nuts and bolts	94
<b>Total</b>	<b><u>670</u></b>

Friction between the arms and PLA components is neglected. Mounted to ensure as low friction as possible.

## Appendix K

### Results from the dynamic payload capacity tests with gearing

Additional mass (m) [Kg]	LSS HS1 - dynamic payload capacity test with gearing				Movement pattern
	Speed (Engine RPM )	Verification (Code)	Verification (Physical)	Comment	[(90,-90), (0,0)]
					[(0,0), (63.7, 50.9)]
0	default			The system reached its desired positions in both code and physical motion	
0,224				The system reached its desired positions physically, but not in code due to the additional mass	
0,371				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
0,4693				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
0,5398				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
0,6159				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
0,7502				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
0,8399				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
0,9778				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
1,16363				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	
1,16363				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	Initially, it crashed during the sidelifit, but with manipulating the positions it was able to complete the desired movement. The zero point for the engines changes every time the engines crashes. Engine 1 managed to perform the sidelifit with a good margin. It did the movement for 5+ mins.
1,39973				The system performed its desired movements physically to a satisfactory level, but the exact positions where not reached in code or physically due to the additional mass	The system performed its desired movements but due to less robust framework the gears on engine 1 skipped teeth. We stop adding weight here because of the less robust framework and possibility of damaging the engines.

# Total mass during test (gram)

Description	Engine w/ ID: 1	Engine w/ ID: 0	Friction between the arms and PLA components is neglected. Mounted to ensure as low friction as possible.
Active and passive arm with toolhub, bolts, nuts and parallel arm	110,75	110,75	
One large metall plate	123,92	115,76	
Two large metall plates	259,74	261,7	
Three small metall plates with bolt and nut	166,4	172,38	
One extra metall plate	123,92	115,76	
<b>Total</b>	<b>784,73</b>	<b>776,35</b>	<b><u>1561,08</u></b>

## Gearing parameters (1:2 scale)

Scale	Lifting capacity (Kg)	Gearing	Lifting capacity with gears (Kg)	Torque (T)	Speed (V)
1:2	0,670	1:1	0,670	T	V
1:2	0,670	1:2	1,340	2*T	V/2
1:2	0,670	1:2,5	1,675	2,5*T	V/2,5
1:2	0,670	1:3	2.01	3*T	V/3

## **Appendix L**

### **Mechanical design**

# Appendix M

## Electrical designs

## 1.1 Controller iterations

The design underwent several iterations before becoming our current solution. This is due to changes in requirements throughout the project and the software and hardware challenges we have encountered. The figures below show the solutions we found throughout the different iterations of the project.

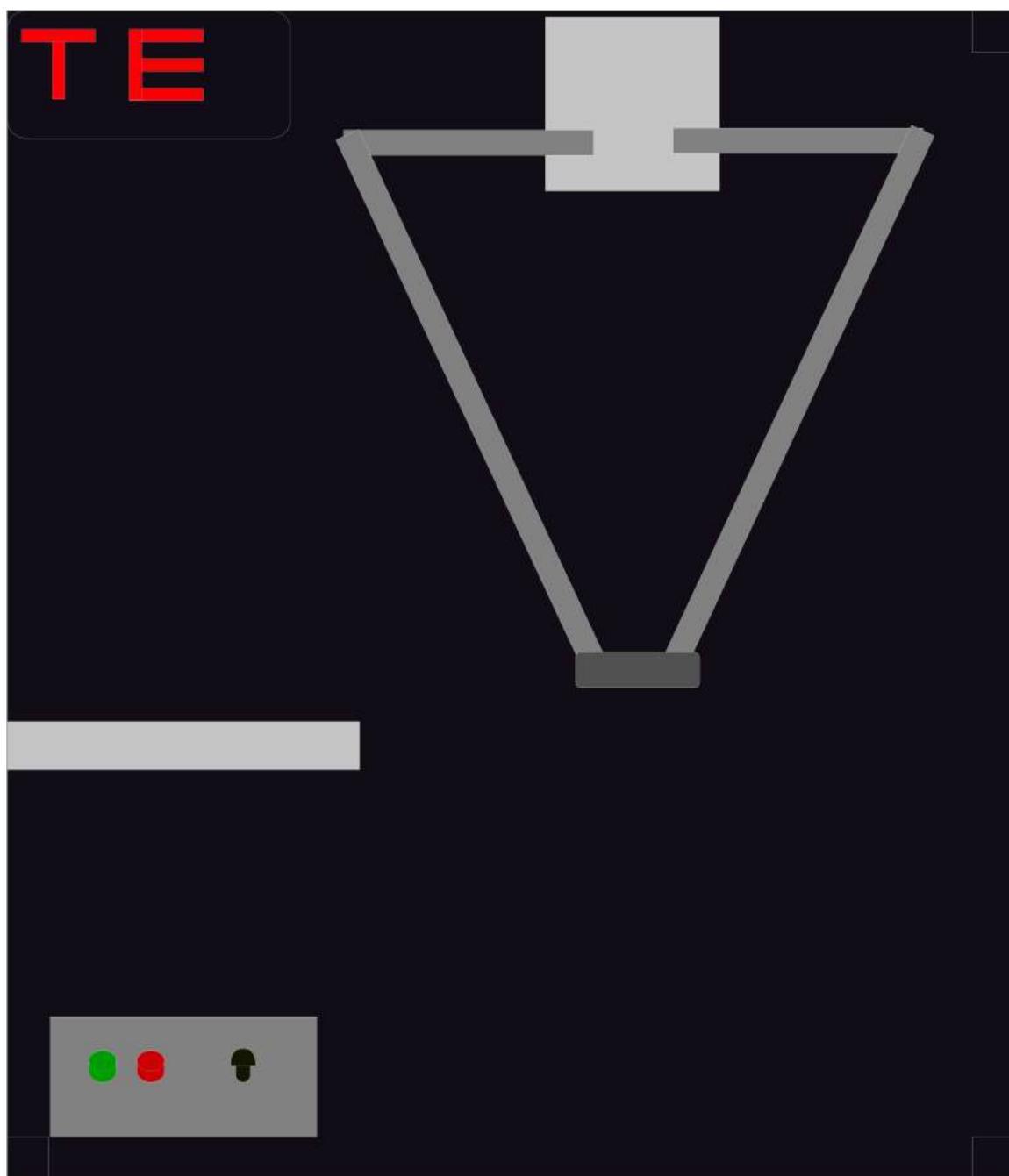


Figure M.1: First iteration control unit design with a integrated controller design within the model.

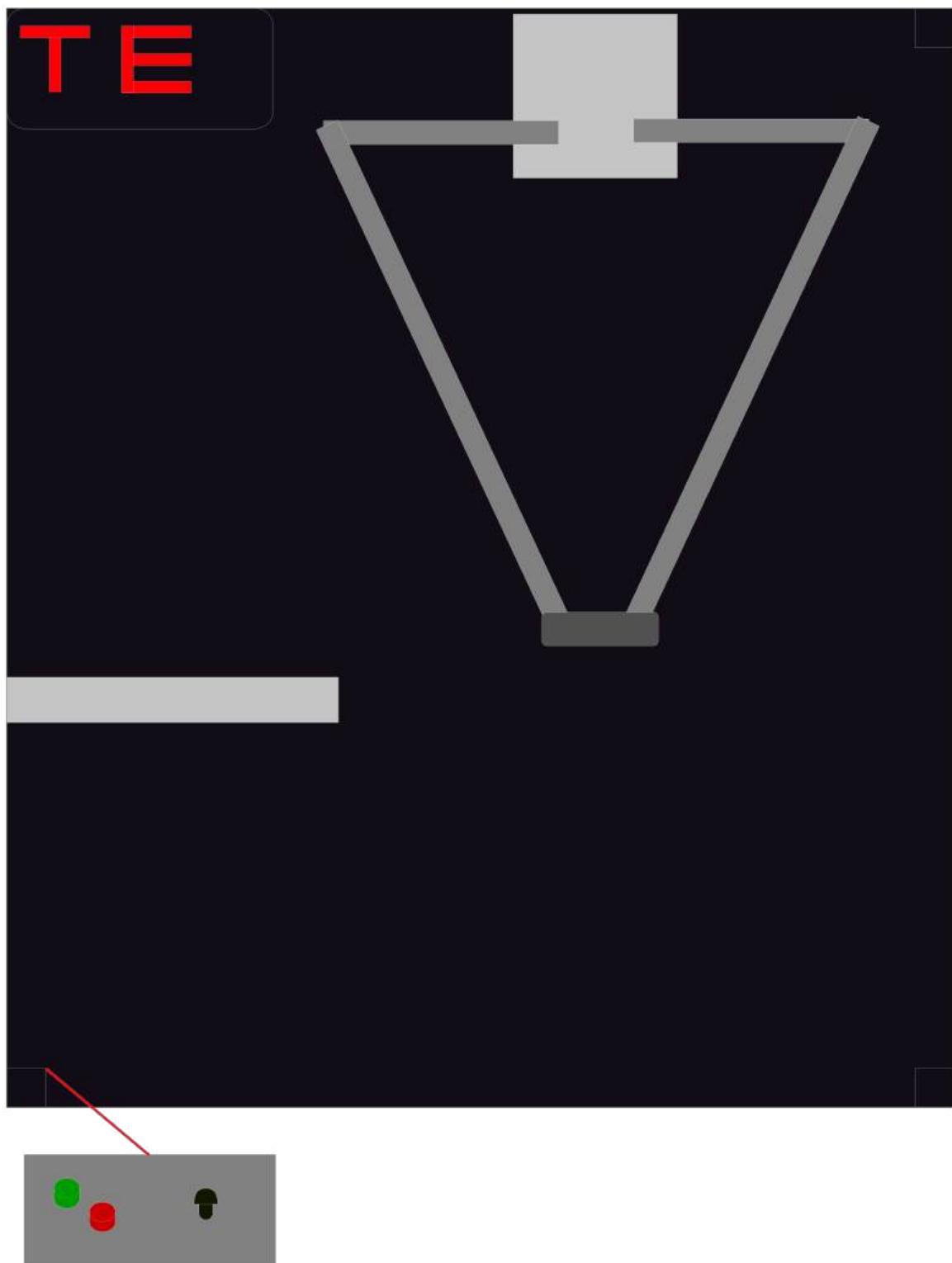


Figure M.2: Second iteration control unit design with an external controller with a connection in the front.

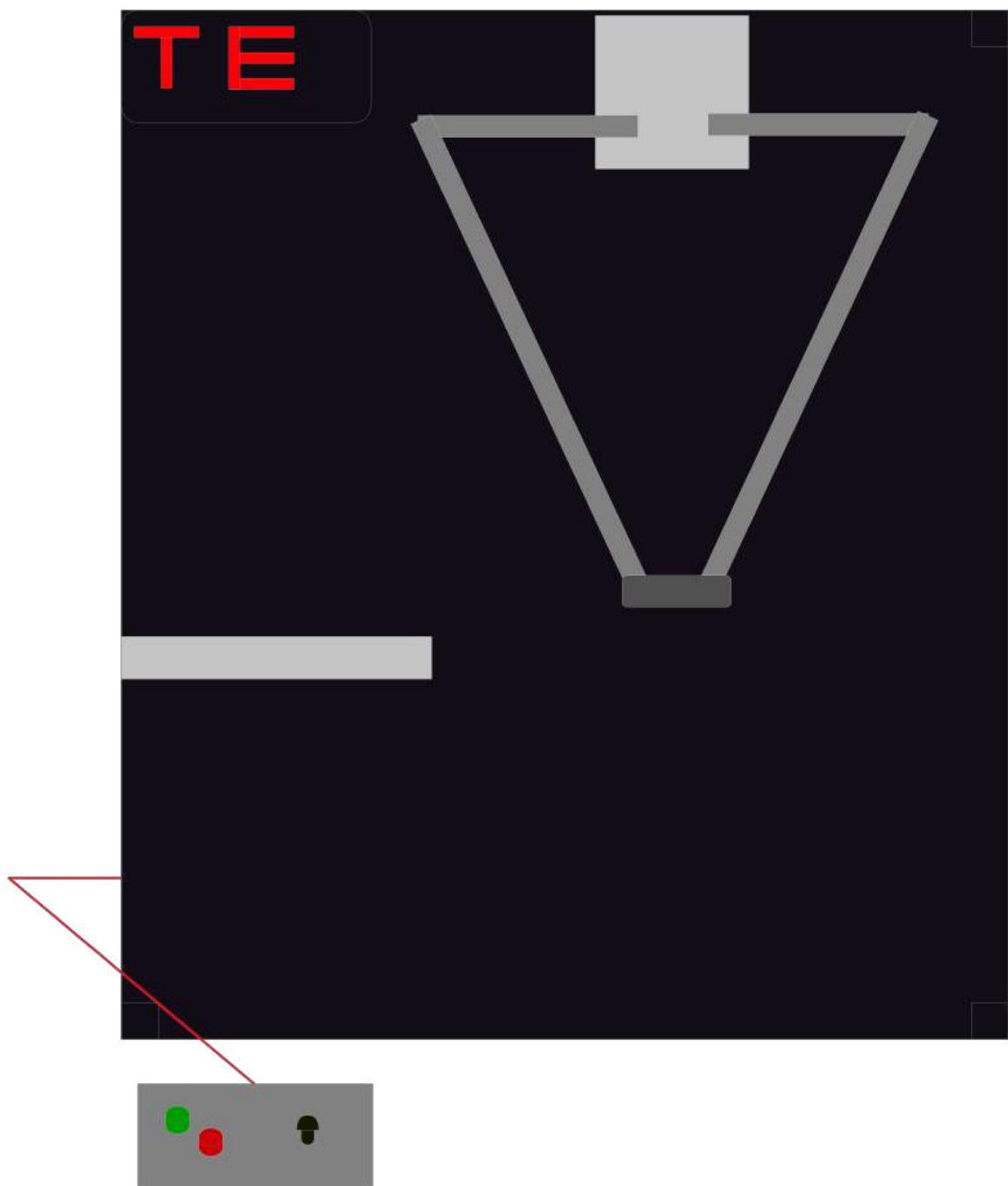


Figure M.3: Third iteration control unit design with an unremovable cable that goes behind the model inside the electrical cabinet.

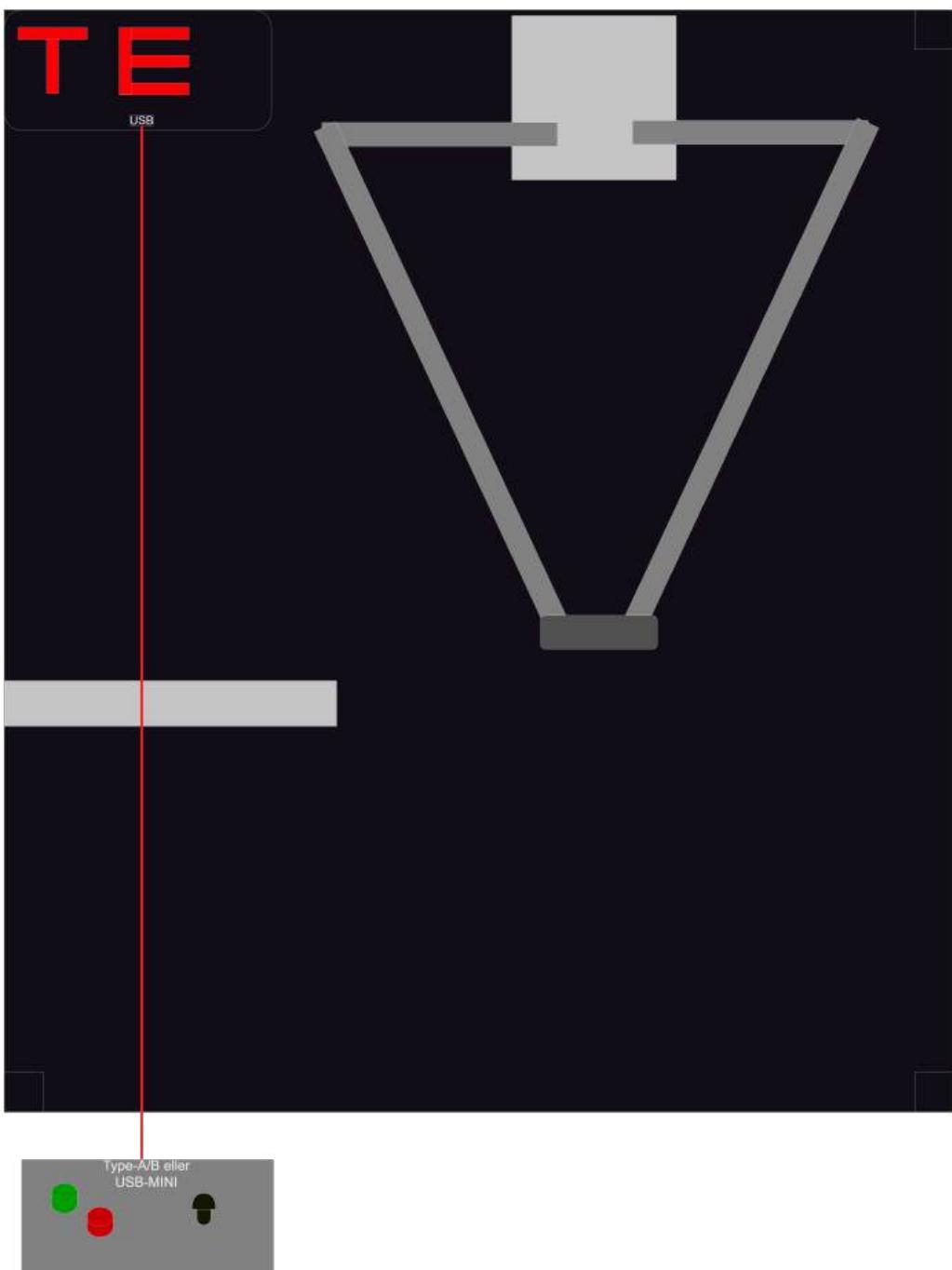


Figure M.4: Fourth iteration control unit design with a USB port at the top of the model by the led logo using either Type-A/B or USB-mini port for the controller.

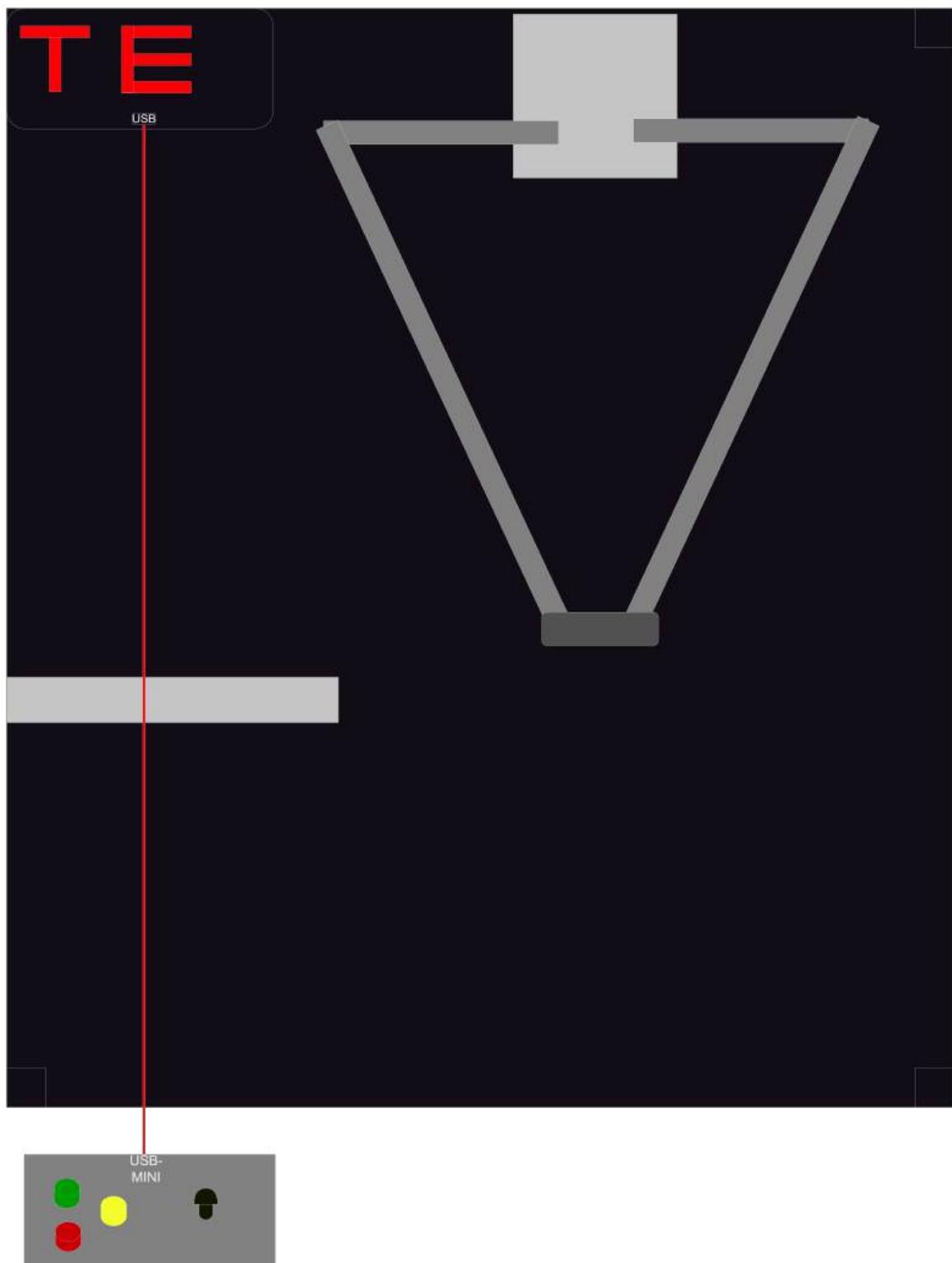


Figure M.5: Fifth iteration control unit design using USB-Mini to USB connection with three buttons.

## 1.2 Controller paths

LTR | AL

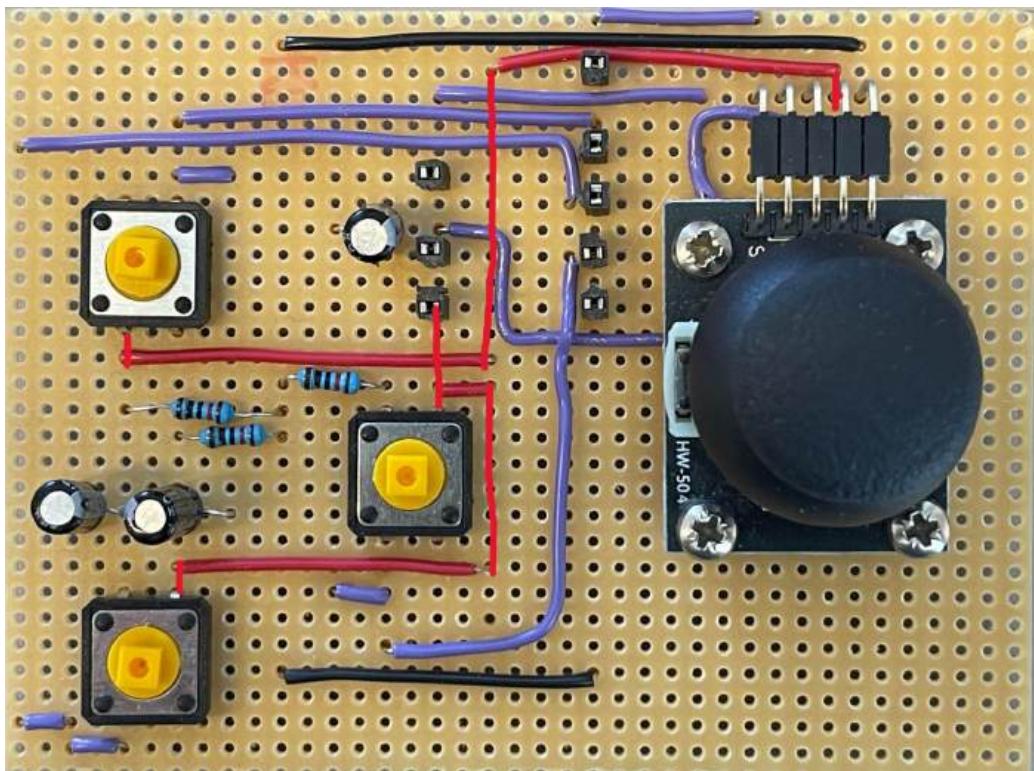


Figure M.6: The red lines show the route the voltage travels between the microcontroller to the components.

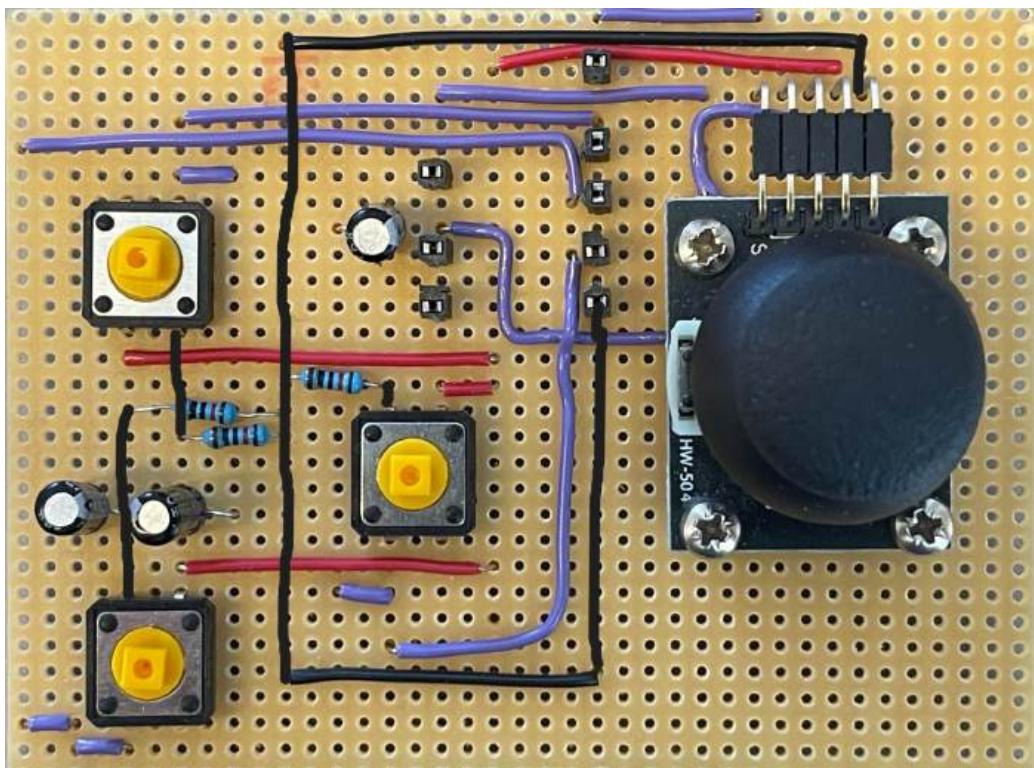


Figure M.7: The black lines show the route the ground goes between the microcontroller and the components.

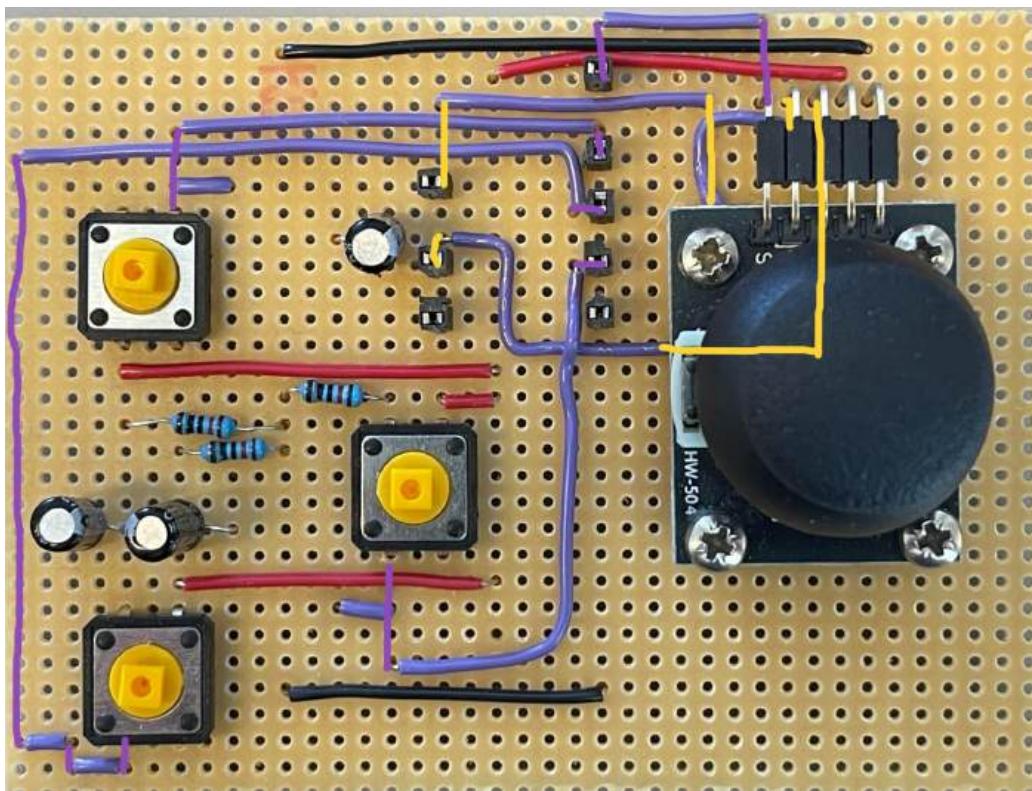


Figure M.8: The purple lines show the digital paths the signal take to the microcontroller whereas the yellow lines show the analog paths.

## 1.3 Led light testing

LTR | AL

### 1.3.1 First iteration led light code

LTR | AL

```
1 #define BUTTON_PIN1 3
2 #define BUTTON_PIN1 4
3 #define GREEN_LED 9
4 #define RED_LED 10
5 #define BLUE_LED 11
6
7
8 int brightness = 255;
9 int gBright = 0;
10 int rBright = 0;
11 int bBright = 0;
12
13
14
15 void setup()
```

```
16 {
17     Serial.begin(9600);
18     pinMode(BUTTON_PIN1, INPUT_PULLUP);
19     pinMode(BUTTON_PIN2, INPUT_PULLUP);
20     pinMode(GREEN_LED, OUTPUT);
21     pinMode(RED_LED, OUTPUT);
22     pinMode(BLUE_LED, OUTPUT);
23 }
24
25 void BlueLight () {
26     gBright = 0;
27     rBright = 0;
28     bBright = 255;
29     analogWrite(GREEN_LED, gBright);
30     analogWrite(RED_LED, rBright);
31     analogWrite(BLUE_LED, bBright);
32         delay(5000);
33 }
34
35 void GreenLight () {
36     gBright = 255;
37     rBright = 0;
38     bBright = 0;
39     analogWrite(BLUE_LED, bBright);
40     analogWrite(GREEN_LED, gBright);
41     analogWrite(RED_LED, rBright);
42         delay(5000);
43 }
44
45 void RedLight () {
46     gBright = 0;
47     rBright = 255;
48     bBright = 0;
49     analogWrite(BLUE_LED, bBright);
50     analogWrite(GREEN_LED, gBright);
51     analogWrite(RED_LED, rBright);
52 }
53
54 void loop()
55 {
```

---

```

56 if (gBright== 0 && rBright == 0 && bBright == 0) {
57     RedLight();
58 }
59     int buttonState1 = digitalRead(BUTTON_PIN1);
60     if (buttonState1 == HIGH) {
61         BlueLight();
62         delay(10);
63         RedLight();
64     }
65     int buttonState2 = digitalRead(BUTTON_PIN2);
66     if (buttonState2 == HIGH) {
67         GreenLight();
68         delay(10);
69         RedLight();
70     }
71 }
```

### 1.3.2 Second iteration led light code with smooth transitions

LTR | AL

```

1 #define BUTTON_PIN1 3
2 #define BUTTON_PIN2 4
3 #define GREEN_LED 10
4 #define RED_LED 8
5 #define BLUE_LED 9
6
7 int gBright = 0; //initial green color
8 int rBright = 0; //initial red color
9 int bBright = 0; //initial blue color
10 int fadeSpeed = 4;
11
12 void setup()
13 {
14     Serial.begin(9600);
15     pinMode(BUTTON_PIN1, INPUT);
16     pinMode(BUTTON_PIN2, INPUT);
17     pinMode(GREEN_LED, OUTPUT);
18     pinMode(RED_LED, OUTPUT);
19     pinMode(BLUE_LED, OUTPUT);
20 }
21 //dims the color to zero value. another variable can be exchanged
```

---

```
    with "255" for different intensity lighting.
22 void dimLight(int pin, int currentBrightness)
23 {
24     for (int currentBrightness = 255; currentBrightness >= 0;
25         currentBrightness--) {
26         analogWrite(pin, currentBrightness);
27         delay(fadeSpeed);
28     }
29 //brightens the color to max value. another variable can be
30 //exchanged with "255" for different intensity lighting.
30 void brightenLights (int pin, int currentBrightness)
31 {
32     for (int currentBrightness = 0; currentBrightness <= 255;
33         currentBrightness++) {
34         analogWrite(pin, currentBrightness);
35         delay(fadeSpeed);
36     }
37
38 void BlueLight () {
39     //since the red color always is the previous value before blue,
40     //its dimmed
40     dimLight(RED_LED, rBright);
41     //function wont return the last brightness value, so its set
42     //manually
42     rBright = 0;
43     brightenLights(BLUE_LED, bBright);
44     //function wont return the last brightness value, so its set
45     //manually
45     bBright = 255;
46     //serves to demonstrate the state activation period
47     delay(5000);
48 }
49
50 void GreenLight ()
51 {
52     //since the red color always is the previous value before green,
53     //its dimmed
53     dimLight(RED_LED, rBright);
```

```
54 //function wont return the last brightness value, so its set
55 //manually
56 rBright = 0;
57 //brightens the color to max value. another variable can be
58 //exchanged with "255" for different intensity lighting.
59 brightenLights(GREEN_LED, gBright);
60 //function wont return the last brightness value, so its set
61 //manually
62 gBright = 255;
63 //serves to demonstrate the state activation period
64 delay(5000);
65 }
66
67 void RedLight ()
68 {
69 //dims the previous color
70 if (gBright == 255){
71 dimLight(GREEN_LED, gBright);
72 //function wont return the last brightness value, so its set
73 //manually
74 gBright = 0;
75 }
76 else if (bBright == 255){
77 dimLight(BLUE_LED, bBright);
78 //function wont return the last brightness value, so its set
79 //manually
80 bBright = 0;
81 }
82
83 void loop()
84 {
85 //turns on red light at startup.
86 if (gBright== 0 && rBright == 0 && bBright == 0){
```

```

87  RedLight();
88 }
89 //reads status of first button
90 int buttonState1 = digitalRead(BUTTON_PIN1);
91 Serial.print(buttonState1);
92 //button press starts "program" and lights respectable color
93 if (buttonState1 == HIGH) {
94   BlueLight();
95   delay(1);
96   RedLight();
97 }
98 //reads status of button 2
99 int buttonState2 = digitalRead(BUTTON_PIN2);
100 //button press starts "program" and lights respectable color.
101 //will only activate if the other program is not active
102 if (buttonState2 == HIGH && buttonState1 == LOW) {
103   GreenLight();
104   delay(1);
105   RedLight();
106 }
```

## 2 Motors

**LTR | AL**

### 2.1 Torque requirement

**LTR | AL**

As a potential solution for the lack of torque the HS1 motors had on the Senior, we considered overvolting the motors. Adhering to the voltage specifications in Fig. ef fig:lss voltage specs, we noted the maximum operating voltage being 12.6 V and the absolute maximum operating voltage being 15.2 V. This means that the voltage could theoretically be increased to 12.6 V, which will gain a higher torque and rpm in a relatively safe manner. If we further push the voltage, we take a higher risk but may get a higher performance. However, it's important to note that either way, the motors can take more amperage and perform better, but their lifetime will be reduced. The servos are also smart, which gives them the function of stopping when the motors meet too much resistance and cut off all power consumption. Though it is smart, it could prove to be a problem for the overvolting due to the motors now using a higher current, which may be read as resistance for the software, causing it to go into the emergency hold.

Specification	HS1
<b>Voltage</b>	
Absolute minimum operating voltage	6V
Minimum operating voltage	6.5V
Recommended operating voltage	12V
Maximum operating voltage	12.6V
Absolute maximum operating voltage	15.2V

Figure M.9: LSS voltage specifications.[10]

### 3 Bill of materials

LTR | AL

Qty.	Producent	Vendor	Article number	Description	Cost
3	Mouser Electronics	RS-Online	IRF520	N-type MOSFET	9-10 NOK
3	Panasonic	RS-Online	ECA2EHG010B	1 $\mu$ F Capacitor	3,71 NOK
3	Omron	RS-Online	B3F-4055	Button	0.240 GBP
1	Mean-well	RS-Online	RD-85A	SMPS	490,27 NOK
2	Arduino	Arduino	A000067	Arduino Mega	42,00 EUR
3	Lynxmotion	Roboshop	RB-Lyn-988	HS1 Smart servo	67.99 USD
1	Mace industries	Mace Industries	STP001	Stop button	450 NOK
1	INF NO	Elkjøp	732308	LED strip	139 NOK

#### Vendors

Name	URL	Location
RS-Online	<a href="https://no.rs-online.com/web/">https://no.rs-online.com/web/</a>	Norway
Roboshop	<a href="http://www.robotshop.com/">www.robotshop.com/</a>	Germany
Arduino	<a href="http://www.store.arduino.cc/">www.store.arduino.cc/</a>	Germany
Mace industries	<a href="http://www.maceindustries.co.uk/">www.maceindustries.co.uk/</a>	Corby, England
Elkjøp	<a href="http://www.elkjop.no/">www.elkjop.no/</a>	Norway

Table M.1: Bill of materials for electrical components

# Appendix N

## Software design

---

# 1 Final lower-level architecture

AL | SG

## 1.1 UC2-E1 - An operator presses the reset or stop the active process button when the system is in any of the other states besides *joystick\_arm\_control*

AL | TM

### 1.1.1 Sequence diagram

AL | TM

The system flow in Fig. N.1 can be described as follows: The operator presses the reset/"Stop the active process" button, which initiates a request to set the system state to *standby* by triggering the `send_system_state_request()` method with the following message data: "standby". This function calls the `publish()` method of the `set_system_state_by_request` publisher, which forwards the message to the topic: "system\_state\_request". The arrival of the message will trigger a callback method in the *State\_Manager* node, which outputs a logger message to the screen, indicating that the reset button press was ignored, since the system is in a state that cannot be changed.

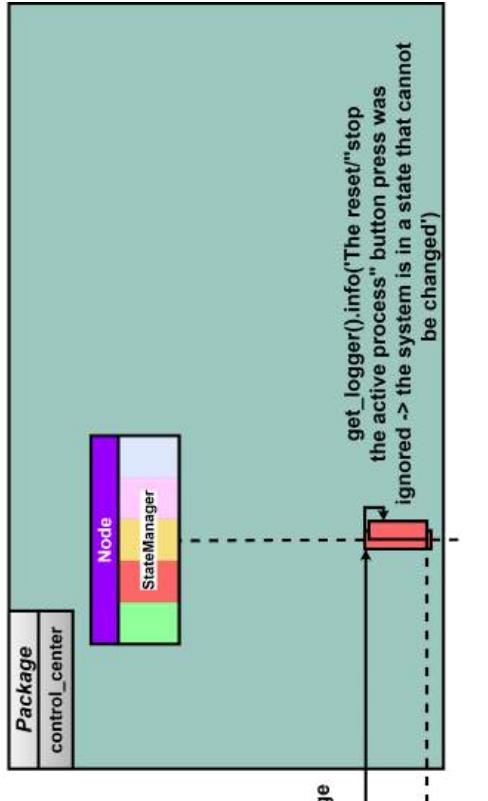
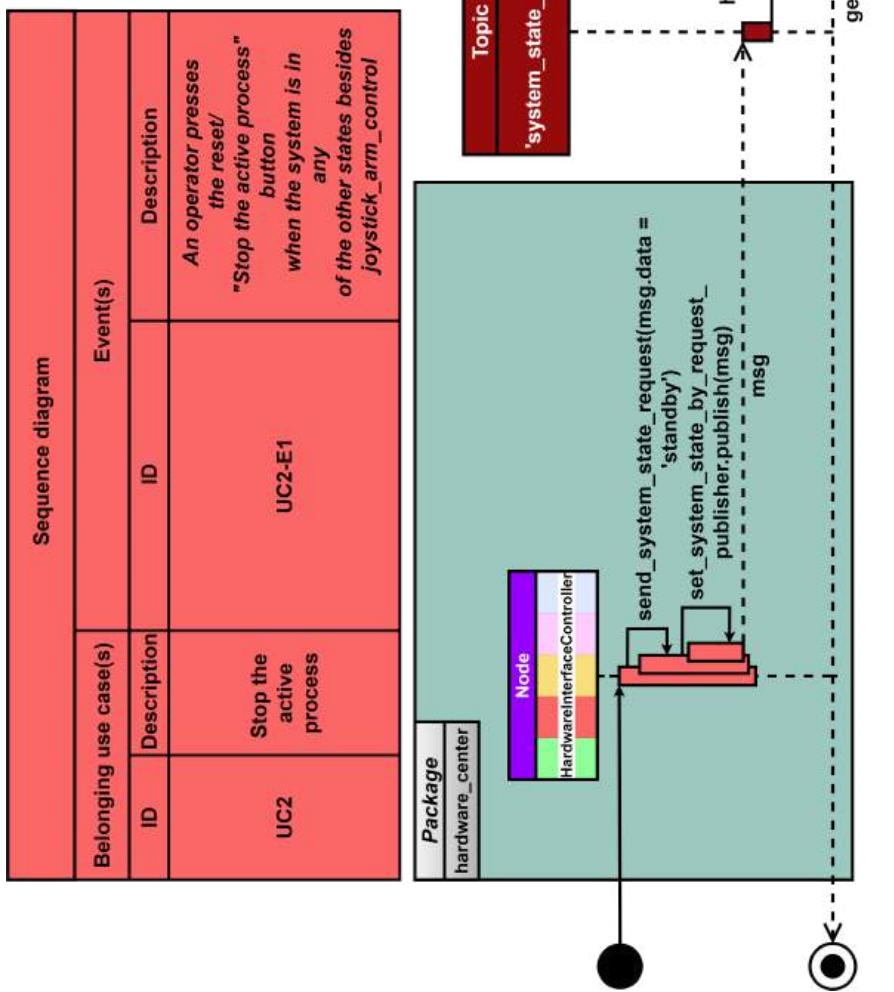


Figure N.1: Sequence diagram - An operator presses the reset or stop the active process button when the system is in any of the other states besides *joystick\_arm\_control*

---

## 1.2 UC2-E2 - An operator presses the reset/"Stop the active process" button when the system state is equal to *joystick\_arm\_control*

AL | TM

### 1.2.1 Sequence diagram

AL | TM

The system flow in Fig. N.2 can be described as follows: The operator presses the reset/"Stop the active process" button, which initiates a request to set the system state to *standby*. This request trigger the `send_system_state_request()` method, which calls the `publish()` method of the `set_system_state_by_request` publisher. The `publish()` method forwards a message with the data: "standby", onto the topic: "system\_state\_request". The arrival of the message triggers a callback method of the "system\_state\_request" subscriber in the *State\_Manager* node. Since the system state is equal to *joystick\_arm\_control* the system state will be reset to *standby*, and the `publish_state()` method will activated. This method, subsequently triggers the `publish()` method of the `state_publisher`, which forwards the current state onto the topic: "system\_state". The arrival of this message triggers the callback method of the "system\_state" topic subscriber in the *Hardware\_Interface\_Controller* which sets the `current_system_state` variable of the node to *standby* and the `previous_system_state` variable to *joystick\_arm\_control*.

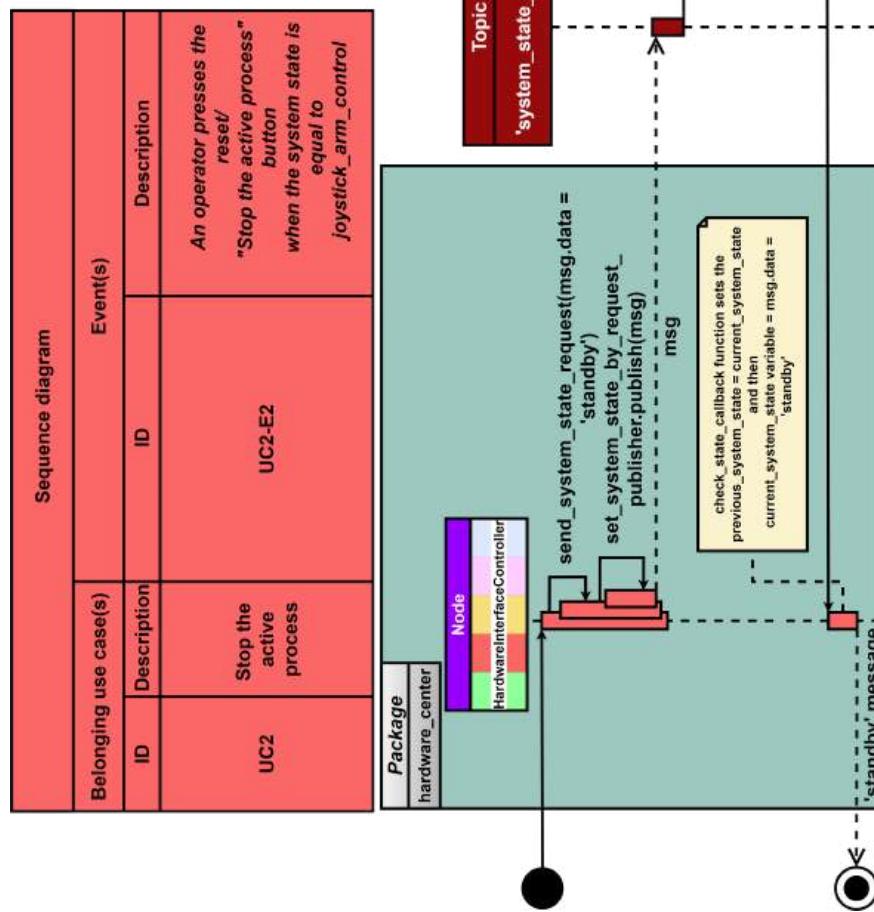


Figure N.2: Sequence diagram - An operator presses the reset or stop the active process button when the system state is joystick\_arm\_control

---

## 1.3 UC3&4-E1 - An operator moves the joystick towards the north or northeast

AL | TM

### 1.3.1 Sequence diagram

AL | TM

The system flow in Fig. N.3 can be described as follows:

- If the system state is either *joystick\_arm\_control* or *map* the *control\_arm\_with\_joystick()* method of the *Hardware\_Interface\_Controller* node will trigger, and the incoming analog values will be evaluated. For this sequence the latter evaluation has two outcomes:
  - If both the analog value representing the x - axis and the analog value representing the z - axis is within the interval: [0, 257], it indicates the prescription of a northeast movement. This outcome has two alternatives based on the system state:
    - \* If the system state is *map* the *move\_arm\_northeast()* method will trigger without any boundary checks and move the prototype (or tool hub) towards the northeast. The method executes until the operator changes the movement of the joystick, which indicates the initiation of another cardinal or ordinal movement sequence, or emergency mode is reached in one or both of the LSS-HS1 motors, due to structural boundaries. The latter causes a motor error which requires a reboot of the system.
    - \* If the system state is *joystick\_arm\_control* the system will continuously check if the LSS-HS1 with ID: 0, has reached its top boundary. The latter has two outcomes:
      - If the condition check evaluates to false, the *move\_arm\_northeast()* will be continuously called.
      - If the condition check evaluates to true, movement of the arms would be paused, and a logger message, indicating that the LSS-HS1 with ID: 0 has reached its top boundary. This ends the sequence, and the operator stands free to move the arms towards an allowed section of the workspace.
  - If the analog value representing the x - axis is within the interval: [0, 100], and the analog value representing the z - axis is within the interval: <257, 767>, it indicates the prescription of a northward movement. This outcome has two alternatives based on the system state:
    - \* If the system state is *map* the *move\_arm\_north()* method will trigger without any boundary checks and move the prototype (or tool hub) towards the north. The method executes until the operator changes the movement of the joystick or emergency mode is reached in one or both of the LSS-HS1 motors.

- \* If the system state is *joystick\_arm\_control* the system will continuously check if both of the LSS-HS1 motors, connected to the arms, has reached their top boundary. The latter has two outcomes:
  - If the condition check evaluates to false, the *move\_arm\_north()* will be continuously called.
  - If the condition check evaluates to true, movement of the arms would be paused, and a logger message, indicating that one or both of the LSS-HS1 motors have reached their top boundary. This ends the sequence, and the operator stands free to move the arms towards an allowed section of the workspace.
- If the system is in any other state, the movement is ignored.

Tab. N.1 visualizes the use cases and their respective colors, which are associated with event or pre-condition that initiates the sequence.

Table N.1: Sequence diagram table overview - An operator moves the joystick towards the north or northeast

Sequence diagram			
Belonging use case(s)		Event(s)	
ID	Description	ID	Description
UC3	Initiate a mapping sequence	UC3&4-E1	An operator moves the joystick towards the north or northeast
UC4	Control the system with a joystick		

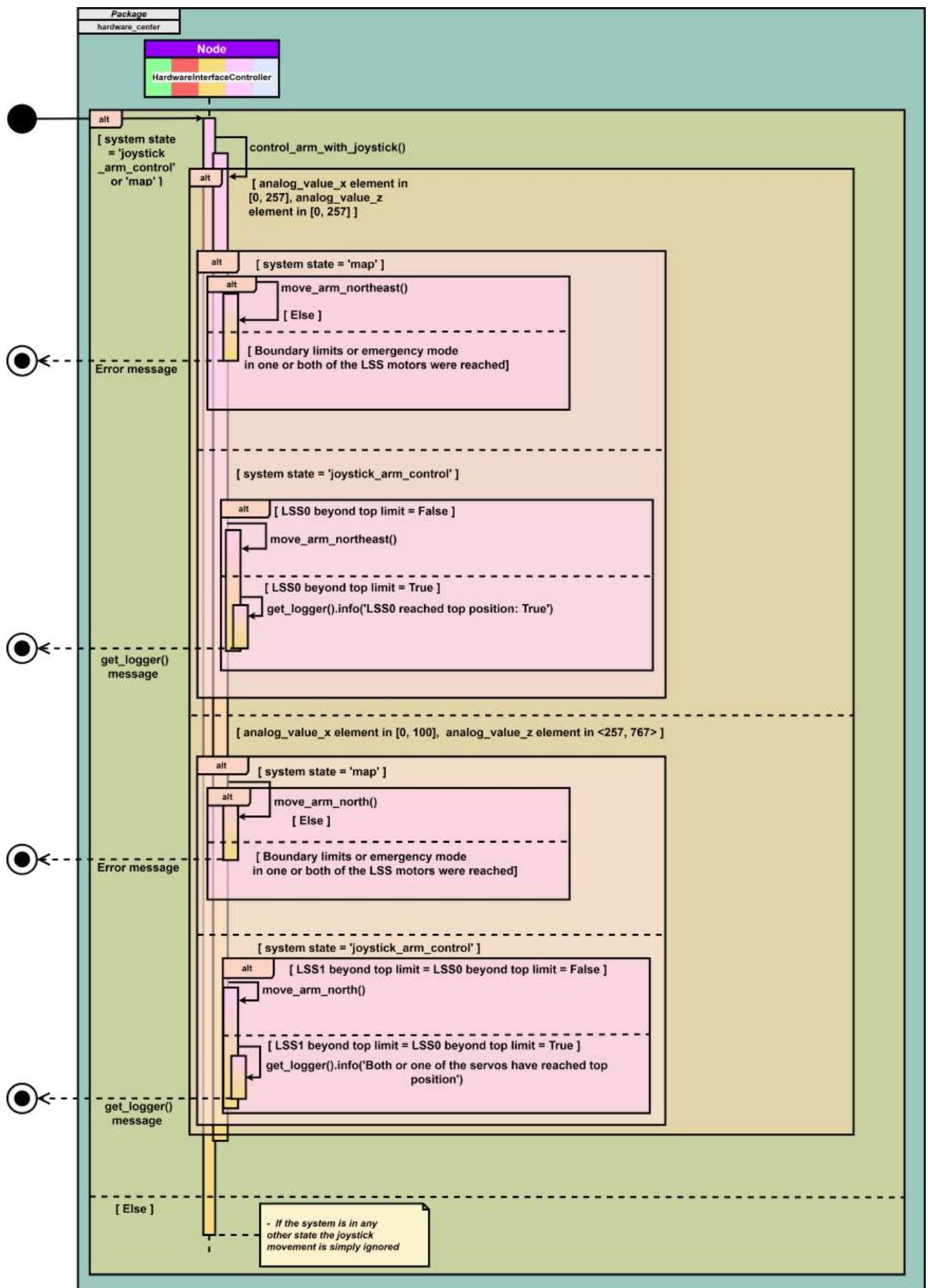


Figure N.3: Sequence diagram - An operator moves the joystick towards the north or northeast

---

## 1.4 UC3&4-E2 - An operator moves the joystick towards the east or southeast

AL | TM

### 1.4.1 Activity diagram

AL | TM

See Fig. N.4.

Activity diagram			
Belonging use case(s)		Event(s)	
ID	Description	ID	Description
UC3	Initiate a mapping sequence	UC3&4-E2	An operator moves the joystick towards the east or southeast
UC4	Control the system with a joystick		

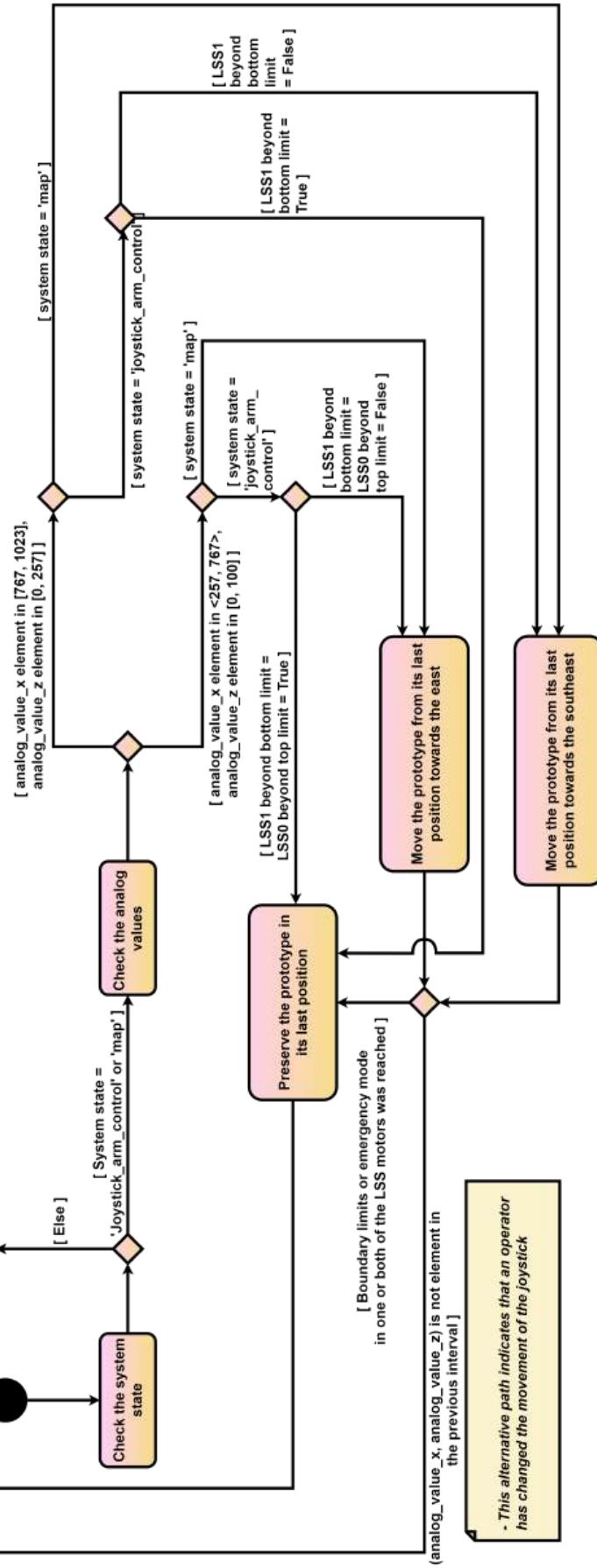


Figure N.4: Activity diagram - An operator moves the joystick towards the east or southeast

---

#### 1.4.2 Sequence diagram

**AL | TM**

See Fig. N.5.

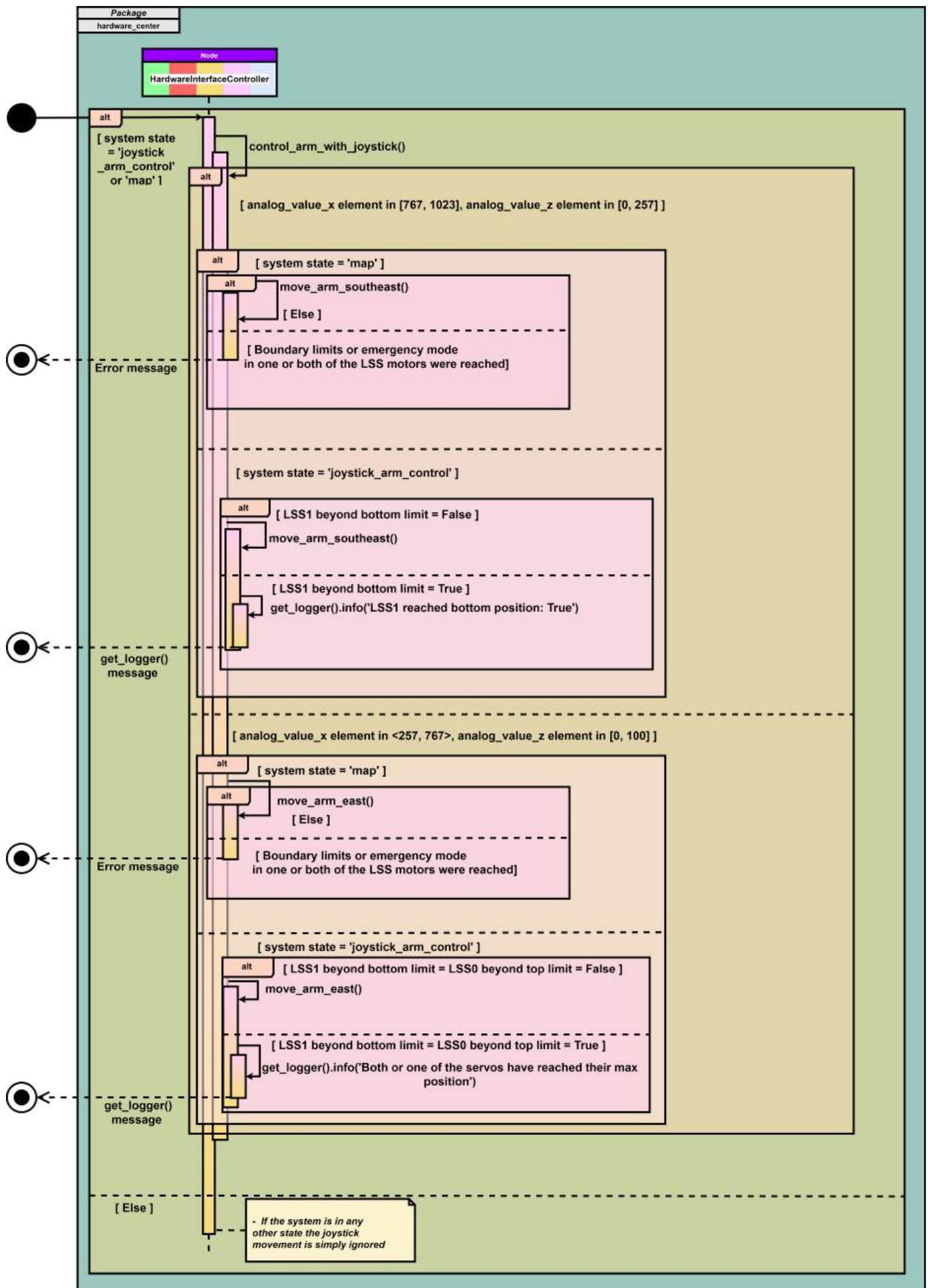


Figure N.5: Sequence diagram - An operator moves the joystick towards the east or southeast

---

**1.5 UC3&4-E3 - An operator moves the joystick towards the south or southwest**

**AL | TM**

**1.5.1 Activity diagram**

**AL | TM**

See Fig. N.6.

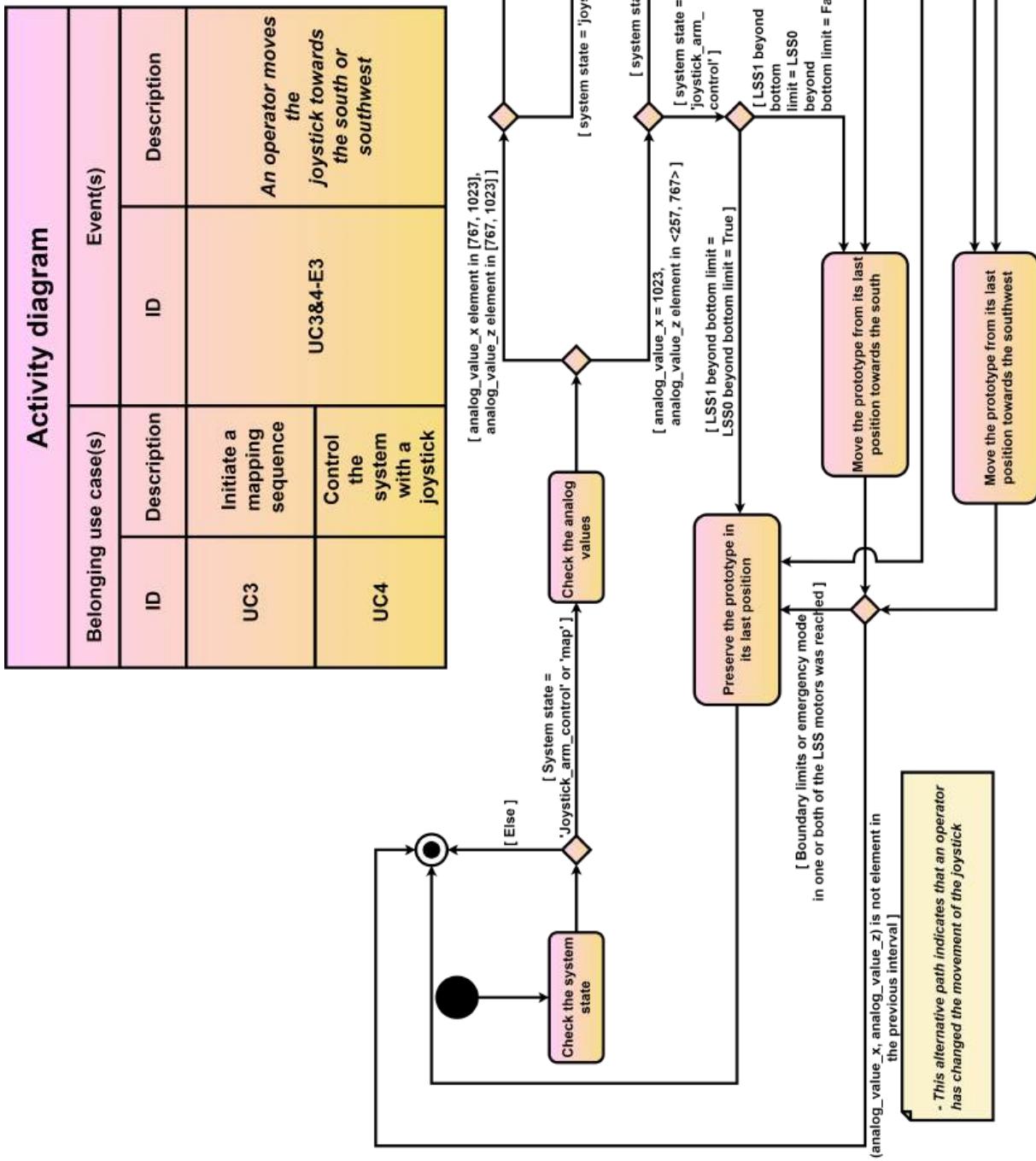


Figure N.6: Activity diagram - An operator moves the joystick towards the south or southwest

---

### 1.5.2 Sequence diagram

**AL | TM**

See Fig. N.7.

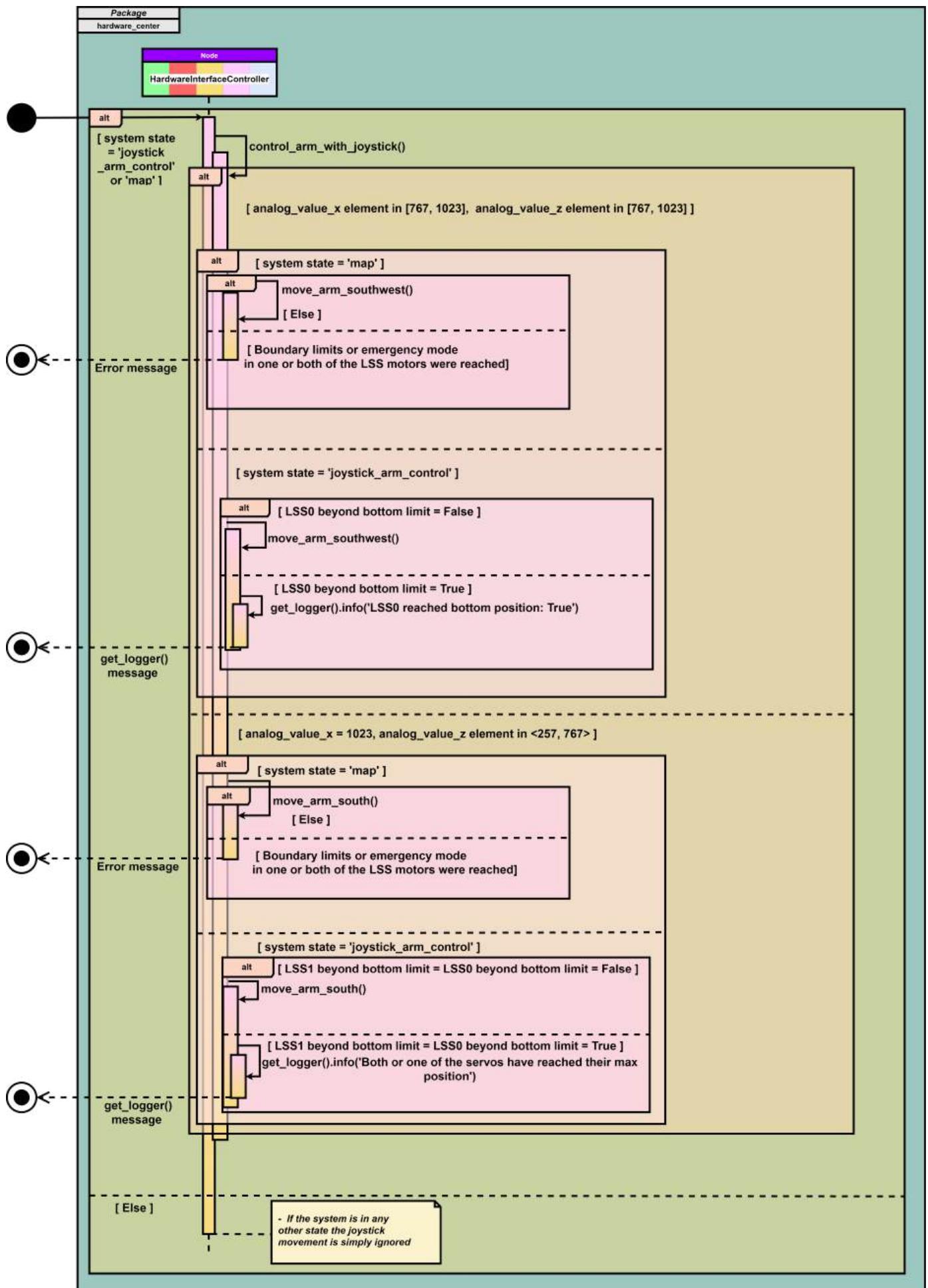


Figure N.7: Sequence diagram - An operator moves the joystick towards the south or southwest

---

**1.6 UC3&4-E4 - An operator moves the joystick towards the west or northwest**

**AL | TM**

**1.6.1 Activity diagram**

**AL | TM**

See Fig. N.8.

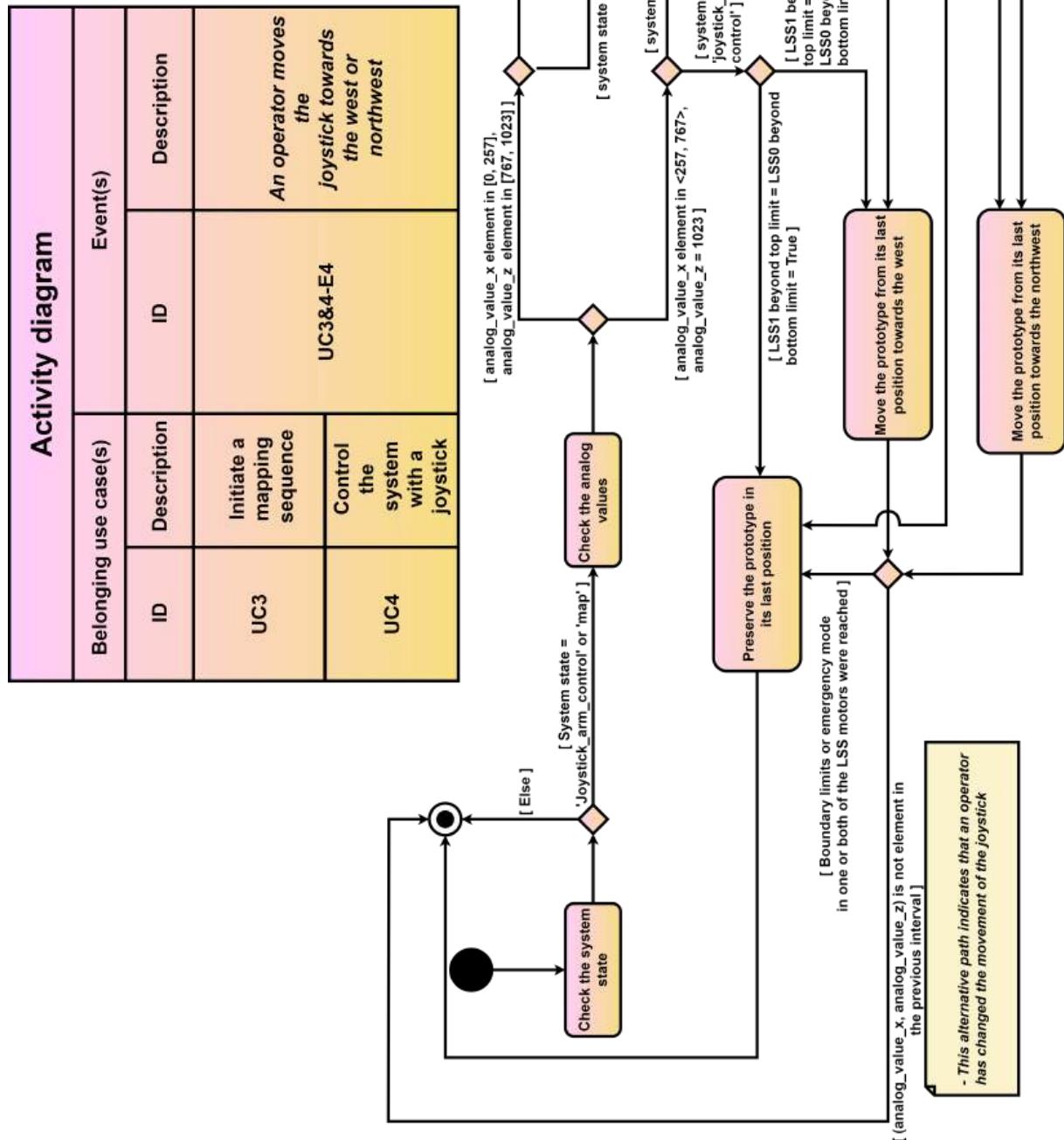


Figure N.8: Activity diagram - An operator moves the joystick towards the west or northwest

---

### 1.6.2 Sequence diagram

**AL | TM**

See Fig. N.9.

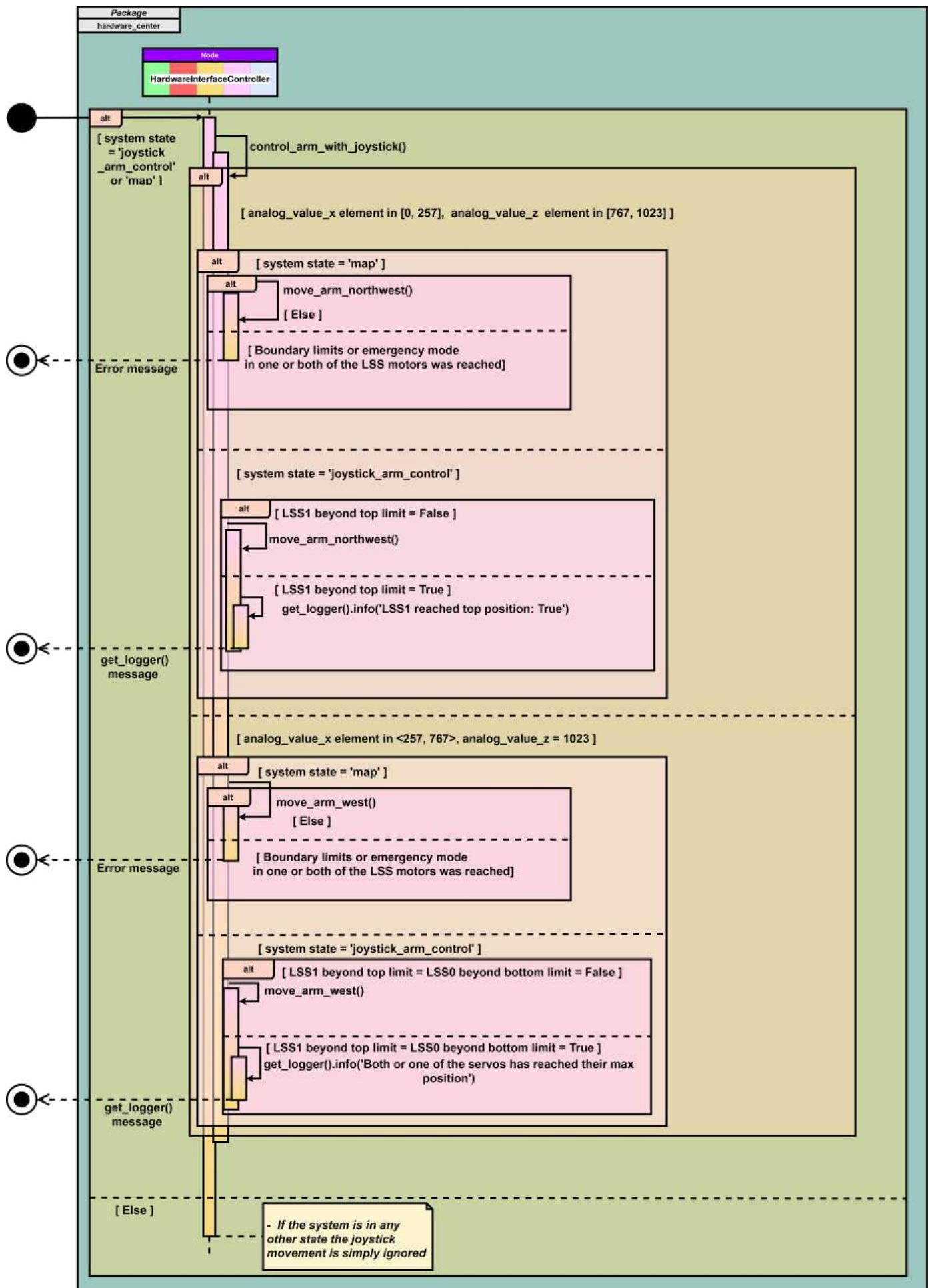


Figure N.9: Sequence diagram - An operator moves the joystick towards the west or northwest

---

## 1.7 UC3&4-E5 - An operator pauses the movement of the joystick

AL | TM

### 1.7.1 Flowchart

AL | TM

See Fig. N.10 and Tab. N.2.

Table N.2: Flowchart table overview - An operator pauses the movement of the joystick

Flowchart			
Belonging use case(s)		Event(s)	
ID	Description	ID	Description
UC3	Initiate a mapping sequence	UC3&4-E5	<i>An operator pauses the movement of the joystick</i>
UC4	Control the system with a joystick		

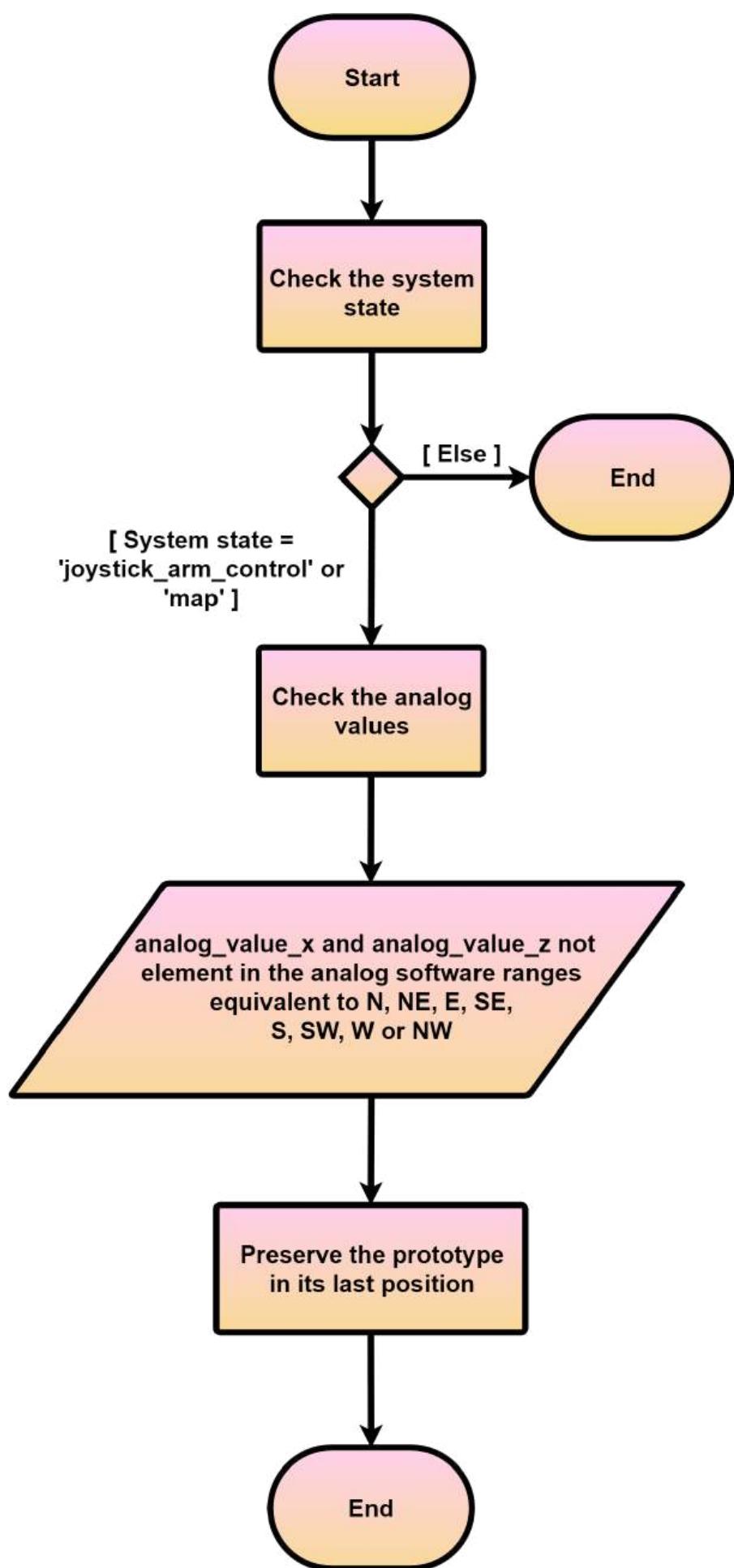


Figure N.10: Flowchart - An operator pauses the movement of the joystick

---

### 1.7.2 Sequence diagram

**AL | TM**

See Fig. N.11.

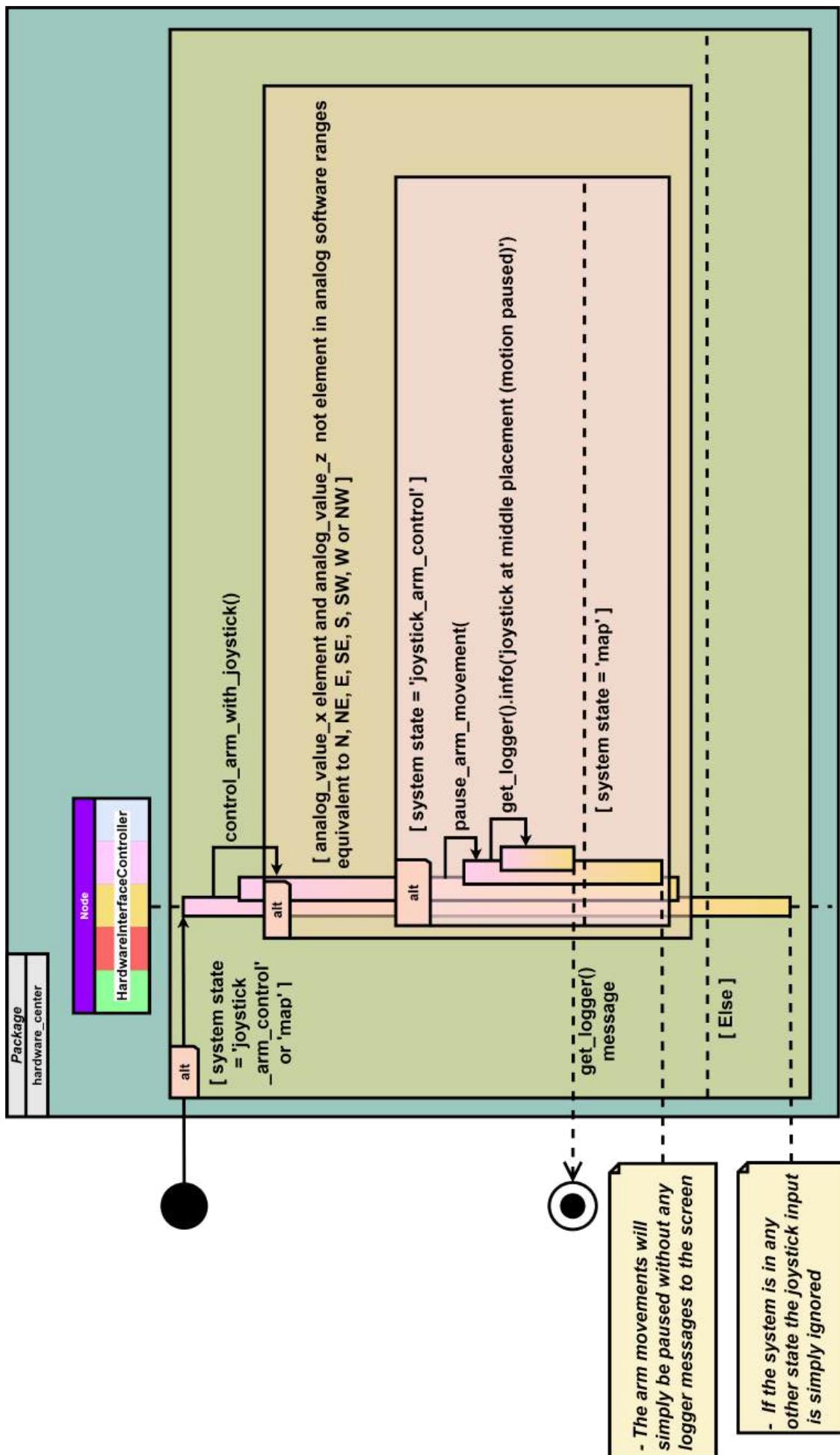


Figure N.11: Sequence diagram - An operator pauses the movement of the joystick

---

## 1.8 UC4&5-E1 - An operator presses the incorporated button of the joystick

AL | TM

### 1.8.1 Sequence diagram

AL | TM

The system flow in Fig. N.12 can be described as follows:

- When an operator or user presses the joystick button the *Hardware\_Interface\_Controller* node will check if there is valid data stored in a JSON file. This condition has two outcomes:
  - If valid data is not found the operator will be prompted with a message to initiate a mapping process, which ends the sequence.
  - If valid data is found the node will publish a message on the "system\_state\_request" topic with the content: "joystick\_control". This publish will, subsequently, trigger a callback method in the *State\_Manager* node, which evaluates the message request in accordance with the value of the state variable stored in the node. This evaluation has three outcomes:
    - \* If the state variable is equal to *standby* the state variable will be set to *joystick\_arm\_control* and a message with the current state will be published to the topic: "system\_state". This message triggers a callback method in the *Hardware\_Interface\_Controller* node, which sets the "current state variable" of that respective node to *joystick\_arm\_control* and the "previous system state" variable to *standby*. This process will then activate the appropriate functionality.
    - \* If the state variable is equal to *joystick\_arm\_control* the state variable will be set to *joystick\_rail\_control* and a message with the current state will be published to the topic: "system\_state". This message triggers a callback method in the *Hardware\_Interface\_Controller* node, which sets the current state variable of that respective node to *joystick\_rail\_control* and the "previous system state" variable to *joystick\_arm\_control*. This process will trigger the *activate\_rail\_system()* method, which triggers the *move\_rail\_system()* function. This method utilizes the last rail position of the JSON file to move the LSS-HS1 connected to the rail to the opposite position (either A or B, depending on the last position). Once the LSS-HS1 has reached the target position, the new rail position is stored in the JSON file, and a request to set the state back to *joystick\_arm\_control* is published on the topic: "system\_state\_request". This triggers the callback method of the *State\_Manager* node, which sets the state variable back to *joystick\_arm\_control*. The state update is then published to the topic: "system\_state", which triggers the callback method in the

---

*Hardware\_Interface\_Controller* node, successfully updating the current state variable of the node to *joystick\_arm\_control* and the previous system state variable to *joystick\_rail\_control*.

- \* If the state variable is equal to any other state beside those previously mentioned, the operator will be prompted with an ignored message, which ends the sequence.

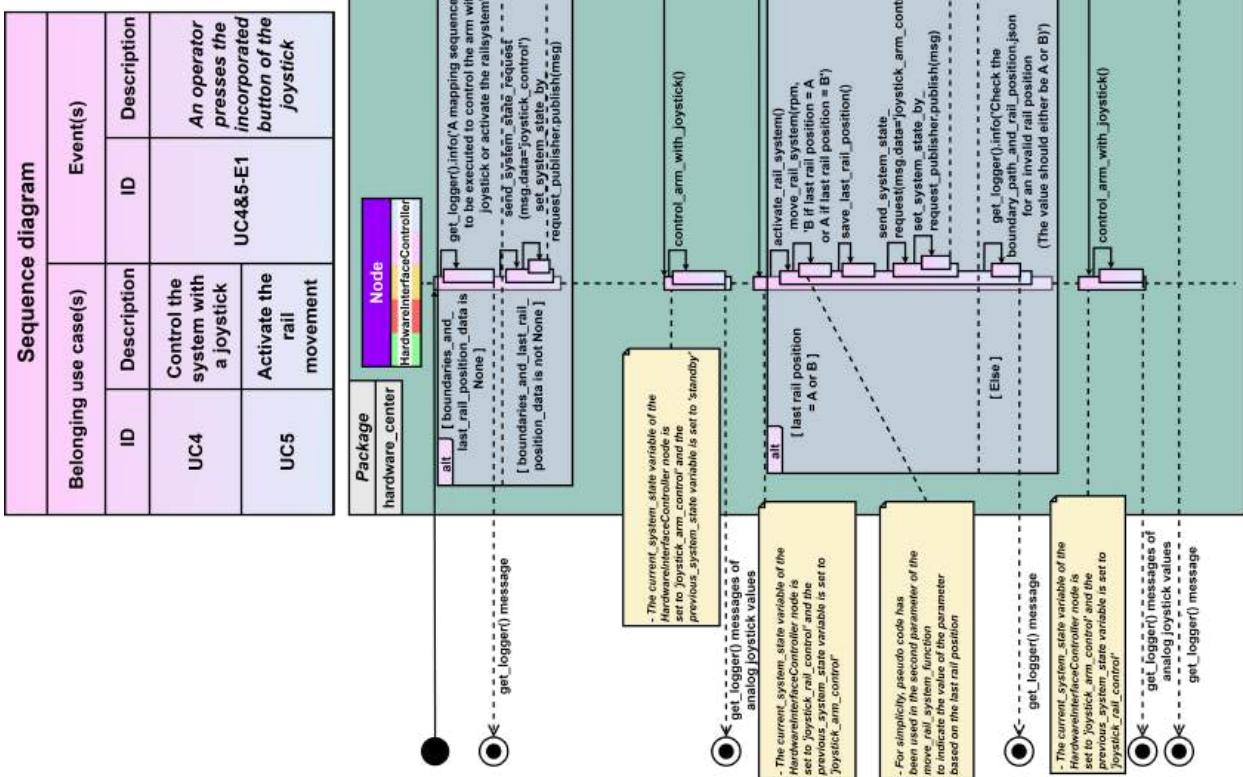


Figure N.12: Sequence diagram - An operator presses the incorporated button of the joystick

---

## 2 Code documentation

TM, AL |

### 2.1 Arduino software

TM, AL |

#### 2.1.1 Transmission software for the Arduino handling the controller

AL | TM

```
1 #define map_button_pin 3
2 #define joy_button_pin 10
3 #define rst_button_pin 5
4 #define rpp_button_pin 7
5 #define x_pin A6
6 #define z_pin A3
7
8 // Variables related to the analog values of the joystick:
9 int x_analog_value;
10 int z_analog_value;
11
12 // Variables related to the interval of data transmission and
13 // readings:
13 unsigned long prev_millis = 0;
14 unsigned long curr_millis;
15 const long interval = 100;
16 const unsigned long debounce_duration = 500;
17
18 // Variables related to the joystick button:
19 unsigned long last_joy_button_press_time = 0;
20 bool prev_joy_button_state = HIGH;
21 int joy_button_pressed = 0;
22 bool current_joy_button_state = HIGH;
23
24 // Variables related to the reset/"stop the active process", the "
25 // run a predefined path", and mapping button:
25 unsigned long last_RST_button_press_time = 0;
26 unsigned long last_RPP_button_press_time = 0;
27 unsigned long last_MAP_button_press_time = 0;
28 bool prev_RST_button_state = LOW;
```

```
29 bool prev_rpp_button_state = LOW;
30 bool prev_map_button_state = LOW;
31 int rst_button_pressed = 0;
32 int rpp_button_pressed = 0;
33 int map_button_pressed = 0;
34 bool current_RST_button_state = LOW;
35 bool current_rpp_button_state = LOW;
36 bool current_map_button_state = LOW;
37
38 // Method to read button presses on the rising or falling edge
39 // depending on state parameter a and b:
40 void read_button_pin(int button_pin, bool &prev_button_state, bool
41 &current_button_state, unsigned long &last_button_press_time,
42 int &button_pressed, int state_value_a, int state_value_b) {
43
44     current_button_state = digitalRead(button_pin);
45
46     if (current_button_state == state_value_a && prev_button_state ==
47         state_value_b) {
48         if (millis() - last_button_press_time >= debounce_duration) {
49             button_pressed = 1;
50             last_button_press_time = millis();
51         }
52     } else {
53         button_pressed = 0;
54     }
55     prev_button_state = current_button_state;
56 }
57
58 void setup() {
59     Serial.begin(115200);
60     pinMode(x_pin, INPUT);
61     pinMode(z_pin, INPUT);
62     pinMode(joy_button_pin, INPUT_PULLUP);
63     pinMode(rst_button_pin, INPUT);
64     pinMode(rpp_button_pin, INPUT);
65     pinMode(map_button_pin, INPUT);
66 }
```

```
65
66 curr_millis = millis();
67
68 // Millis() interval used to control the rate of data
   transmission and readings:
69 if (curr_millis - prev_millis >= interval){
70
71   prev_millis = curr_millis;
72
73   // Read analog values from the correct pins:
74   x_analog_value = analogRead(x_pin);
75   z_analog_value = analogRead(z_pin);
76
77   // Read the joystick button:
78   read_button_pin(joy_button_pin, prev_joy_button_state,
    current_joy_button_state, last_joy_button_press_time,
    joy_button_pressed, LOW, HIGH);
79
80   // Read the reset/"Stop the active process" button:
81   read_button_pin(rst_button_pin, prev_RST_button_state,
    current_RST_button_state, last_RST_button_press_time,
    rst_button_pressed, HIGH, LOW);
82
83   // Read the "Run a predefined path" button:
84   read_button_pin(rpp_button_pin, prev_rpp_button_state,
    current_rpp_button_state, last_rpp_button_press_time,
    rpp_button_pressed, HIGH, LOW);
85
86   // Read the "Mapping" button:
87   read_button_pin(map_button_pin, prev_map_button_state,
    current_map_button_state, last_map_button_press_time,
    map_button_pressed, HIGH, LOW);
88
89   Serial.print(x_analog_value);
90   Serial.print(",");
91   Serial.print(z_analog_value);
92   Serial.print(",");
93   Serial.print(joy_button_pressed);
94   Serial.print(",");
95   Serial.print(rst_button_pressed);
```

```
96     Serial.print(",");
97     Serial.print(rpp_button_pressed);
98     Serial.print(",");
99     Serial.print(map_button_pressed);
100    Serial.println();
101 }
102 }
```

## 2.1.2 Receiver software for the Arduino controlling the LED lights

TM | AL

```
1 #define GREEN_LED 10
2 #define RED_LED 8
3 #define BLUE_LED 9
4 #define raspi_red_signal 52
5 #define raspi_blue_signal 50
6 #define raspi_green_signal 48
7
8 void setup() {
9     Serial.begin(115200);
10    pinMode(raspi_red_signal, INPUT);
11    pinMode(raspi_blue_signal, INPUT);
12    pinMode(raspi_green_signal, INPUT);
13    pinMode(GREEN_LED, OUTPUT);
14    pinMode(RED_LED, OUTPUT);
15    pinMode(BLUE_LED, OUTPUT);
16 }
17
18 void loop() {
19
20     static int lastState = 0; // Keeps track of the last state
21
22     int redState = digitalRead(raspi_red_signal);
23     int greenState = digitalRead(raspi_green_signal);
24     int blueState = digitalRead(raspi_blue_signal);
25
26     // Calculate the new state based on combination of inputs:
27     int newState = 0;
28     if (redState && !greenState && !blueState) newState = 1; // Red
color: standby state
```

```
29 else if (redState && !greenState && blueState) newState = 2; //  
    Pink color: joystick_arm_control state  
30 else if (!redState && greenState && blueState) newState = 3; //  
    Blue: color: joystick_rail_control state  
31 else if (redState && greenState && !blueState) newState = 4; //  
    Yellow color: map state  
32 else if (!redState && greenState && !blueState) newState = 5; //  
    Green color: run_predefined_path state  
33  
34 // Only change LED states if the new state is different from the  
last state:  
35 if (newState != lastState) {  
36     switch (newState) {  
37  
38         case 1:  
39             analogWrite(RED_LED, 255);  
40             analogWrite(GREEN_LED, 0);  
41             analogWrite(BLUE_LED, 0);  
42             break;  
43  
44         case 2:  
45             analogWrite(RED_LED, 255);  
46             analogWrite(GREEN_LED, 0);  
47             analogWrite(BLUE_LED, 125);  
48             break;  
49  
50         case 3:  
51             analogWrite(RED_LED, 0);  
52             analogWrite(GREEN_LED, 0);  
53             analogWrite(BLUE_LED, 255);  
54             break;  
55  
56         case 4:  
57             analogWrite(RED_LED, 255);  
58             analogWrite(GREEN_LED, 255);  
59             analogWrite(BLUE_LED, 0);  
60             break;  
61  
62         case 5:  
63             analogWrite(RED_LED, 0);
```

---

```
64     analogWrite(GREEN_LED, 255);
65     analogWrite(BLUE_LED, 0);
66     break;
67 }
68 lastState = newState; // Update last state
69 }
70 }
71 }
```

Both arduino codes in these sub-subsections were formatted using code from the following GitHub repository: [133].

## 2.2 The first iteration of the mapper node

TM | AL

The first iteration of the Mapper node was designed to use inverse kinematics together with some reference points where angles are used to calculate the desired motor angles at each point in a large grid. It would calculate the angles for both motors one set at the time, then send a request for the *MotorController* to move to these angles, read the actual angles and send them back to the *Mapper*. The angles would then be validated within a tolerance and stored, which continued until all points had been mapped. Further, at each point the *Mapper* node used boundary angles and a circle placed in the middle of the workarea to determine whether the point is inside or outside the workarea. We did get this functionality to work, but it did not deliver the results we looked for: the robot arm was not able to move all the way to the bottom of the workarea. One of the possible reasons for this was faults in the inverse kinematics, therefore we made a test script for visualizing the robot arm at different positions. This was helpful, as it gave us a better understanding of the relationship between the digital and physical arms. It also helped us tweak different variables making the calculation more accurate, yet we still did not get the desired result. Due to time constraints we decided to try a simpler solution, which ended up working great.

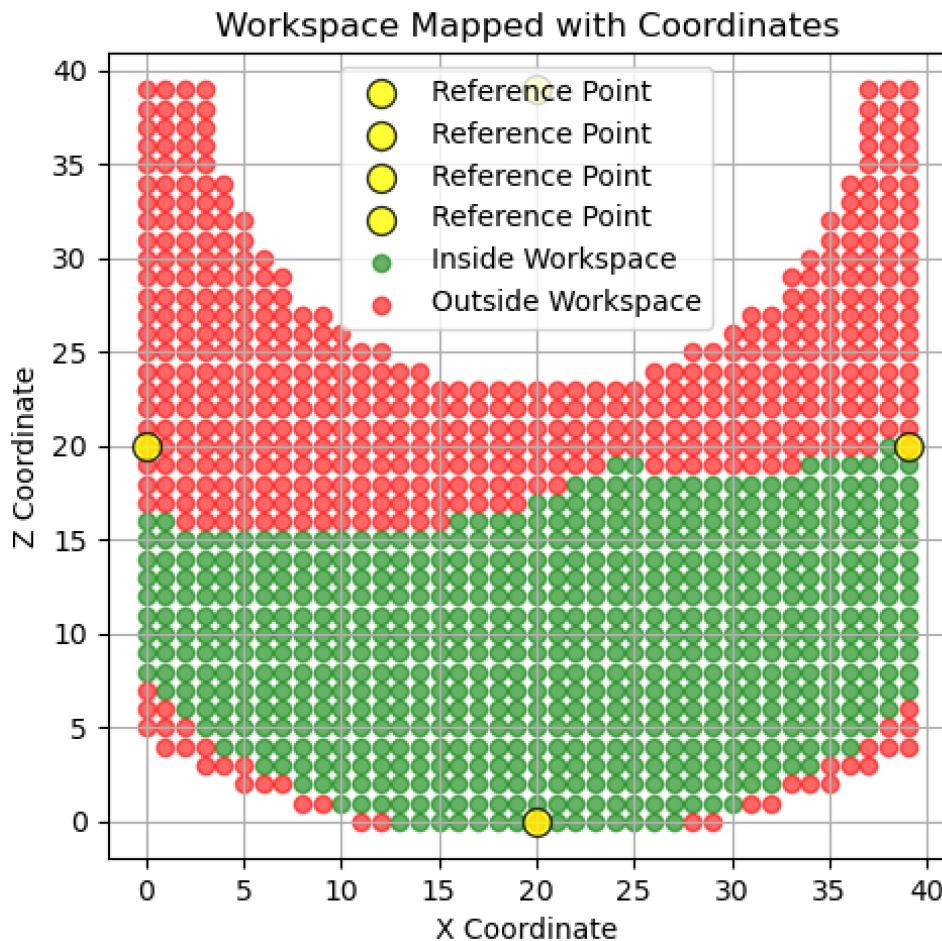


Figure N.13: Mapped workspace visualized with matplotlib

---

Inverse kinematics code used to calculate angles given a coordinate. This was used in both the mapping sequence and the IK test script. It was made with the help of ChatGTP.

```
1 def IK_equation(p, x_target, z_target, D, W, grid_size_x, grid_size_z, offset=1)
2     :
3     theta1_left, theta2_left, theta1_right, theta2_right = p
4
5     # Base position calculations
6     x_center = grid_size_x / 2
7     z_base = grid_size_z
8     x_base_left = x_center - W / 2
9     x_base_right = x_center + W / 2
10
11    # Left arm calculations
12    x_elbow_left = x_base_left + LL1 * np.cos(theta1_left)
13    z_elbow_left = z_base + LL1 * np.sin(theta1_left)
14    x_hand_left = x_elbow_left + LL2 * np.cos(theta1_left + theta2_left)
15    z_hand_left = z_elbow_left + LL2 * np.sin(theta1_left + theta2_left)
16
17    # Right arm calculations
18    x_elbow_right = x_base_right + LR1 * np.cos(theta1_right)
19    z_elbow_right = z_base + LR1 * np.sin(theta1_right)
20    x_hand_right = x_elbow_right + LR2 * np.cos(theta1_right + theta2_right)
21    z_hand_right = z_elbow_right + LR2 * np.sin(theta1_right + theta2_right)
22
23    # Hands positioned slightly left and right of the target
24    eq1 = x_hand_left - (x_target - offset) # Left hand x-coordinate offset to
25    # the left of target
26    eq2 = z_hand_left - z_target # Left hand z-coordinate to be at target
27    height
28    eq3 = x_hand_right - (x_target + offset) # Right hand x-coordinate offset to
29    # the right of target
30    eq4 = z_hand_right - z_target # Right hand z-coordinate to be at target
31    height
32
33    return (eq1, eq2, eq3, eq4)
```

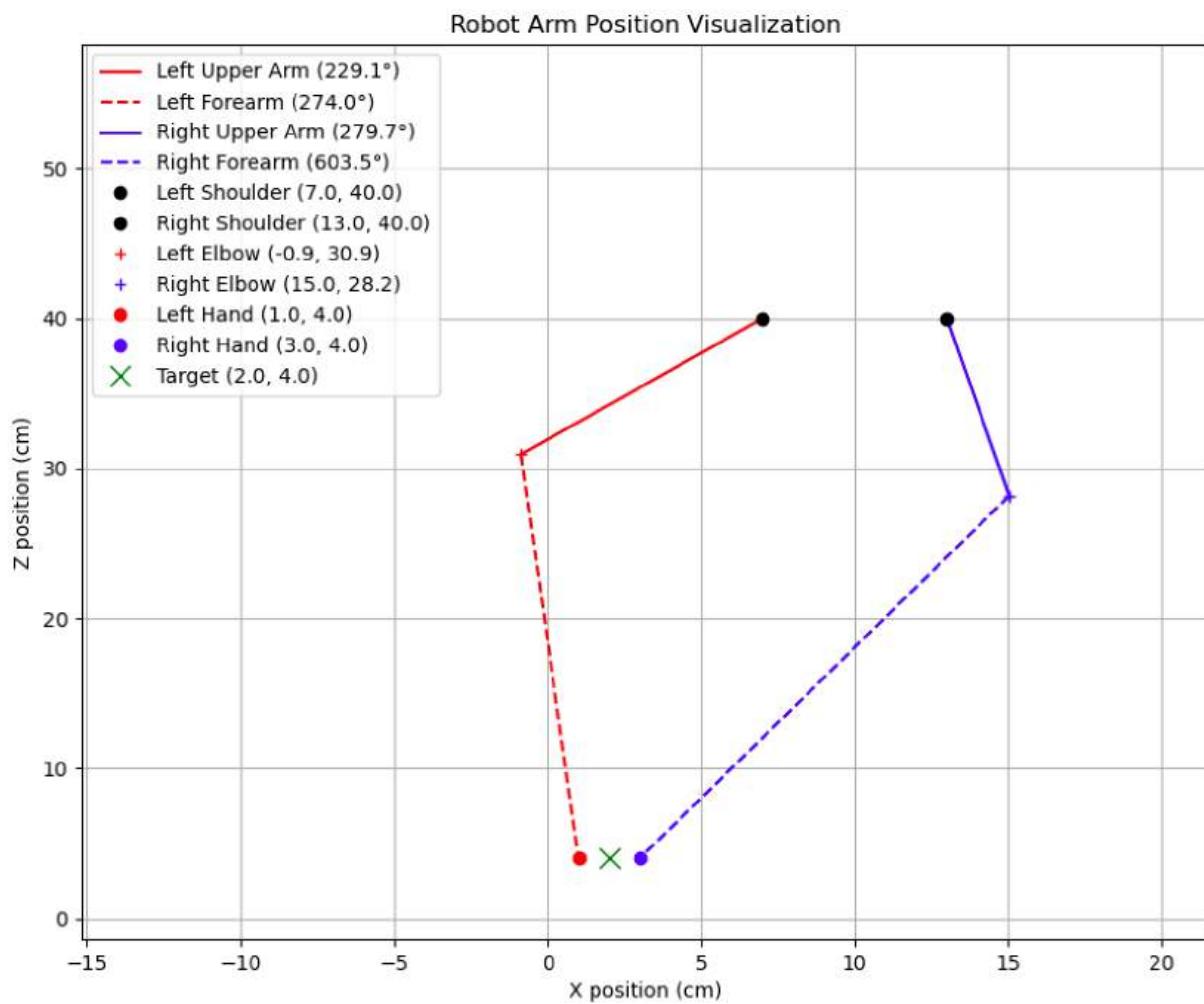


Figure N.14: Visualization of the robot or with the script made for testing the inverse kinematics equation.

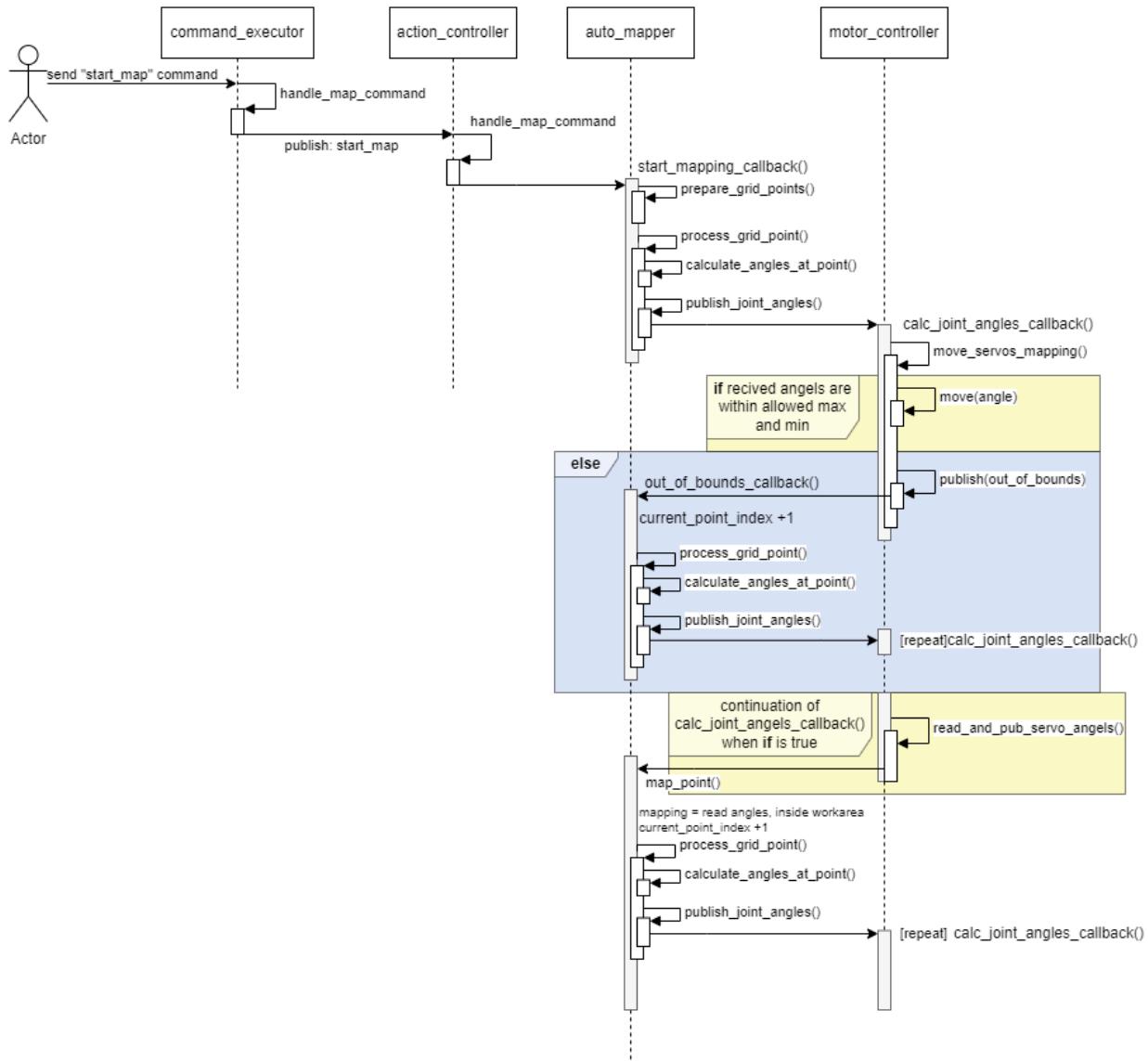


Figure N.15: Sequence diagram describing the first iteration mapping sequence.

## 2.3 Command system

TM | AL

At the start of development we made a command system for initiating sequences like mapping, reading of angles, setting nullpoints and more. This was a great addition and starting point for development as it made testing and running functionality an easy process. The functionality was split in two nodes, one for configuring and displaying the window, and one for executing the commands.

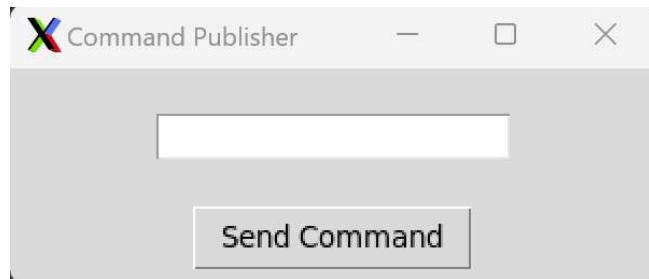


Figure N.16: Command window

## 2.4 GUI buttons

TM | AL

The GUI buttons were used during testing, and made development easier. As the GUI button is pressed, a message is sent to initiate a function. The message is sent to the same topic that a physical buttons would send to, making the integration of physical buttons straight forward. When using SSH to develop on a remote PI we had to use something called X11 forwarding to make the UI visible on our local machines. This is simple to setup, as it only requires enabling X11 in the SSH config on both sides.



Figure N.17: GUI buttons

## 2.5 Documentation generated by Doxygen

TM, AL | AL, TM



Documentation  
ROS2 Python  
Generated by Doxygen

---

<b>1 Class Index</b>	1
1.1 Class List . . . . .	1
<b>2 Class Documentation</b>	3
2.1 control_center.command_executor.CommandExecutor . . . . .	3
2.1.1 Detailed Description . . . . .	3
2.1.2 Constructor & Destructor Documentation . . . . .	3
2.1.2.1 <code>__init__()</code> . . . . .	4
2.1.3 Member Function Documentation . . . . .	4
2.1.3.1 <code>command_callback()</code> . . . . .	4
2.1.3.2 <code>process_command()</code> . . . . .	4
2.2 control_center.command_window.CommandPublisher . . . . .	4
2.2.1 Detailed Description . . . . .	5
2.2.2 Constructor & Destructor Documentation . . . . .	5
2.2.2.1 <code>__init__()</code> . . . . .	5
2.2.3 Member Function Documentation . . . . .	5
2.2.3.1 <code>send_command()</code> . . . . .	5
2.3 control_center.display.DisplayNode . . . . .	6
2.3.1 Detailed Description . . . . .	6
2.3.2 Constructor & Destructor Documentation . . . . .	6
2.3.2.1 <code>__init__()</code> . . . . .	6
2.3.3 Member Function Documentation . . . . .	7
2.3.3.1 <code>animate()</code> . . . . .	7
2.3.3.2 <code>display_callback()</code> . . . . .	7
2.3.3.3 <code>read_mapping()</code> . . . . .	7
2.3.3.4 <code>target_point_callback()</code> . . . . .	8
2.4 hardware_center.gpio_controller.GPIOController . . . . .	8
2.4.1 Detailed Description . . . . .	8
2.4.2 Constructor & Destructor Documentation . . . . .	8
2.4.2.1 <code>__init__()</code> . . . . .	9
2.4.3 Member Function Documentation . . . . .	9
2.4.3.1 <code>handle_state_update()</code> . . . . .	9
2.4.3.2 <code>on_shutdown()</code> . . . . .	9
2.4.3.3 <code>set_led_state()</code> . . . . .	10
2.4.3.4 <code>setup_gpio()</code> . . . . .	10
2.5 hardware_center.hardware_interface_controller.HardwareInterfaceController . . . . .	10
2.5.1 Detailed Description . . . . .	12
2.5.2 Member Function Documentation . . . . .	12
2.5.2.1 <code>activate_rail_system()</code> . . . . .	12
2.5.2.2 <code>calculate_time_to_run_rail_system()</code> . . . . .	12
2.5.2.3 <code>check_state_callback()</code> . . . . .	13
2.5.2.4 <code>cleanup_serial()</code> . . . . .	13

---

2.5.2.5 clear_button_press_flags()	13
2.5.2.6 control_arm_with_joystick()	14
2.5.2.7 lss1_beyond_bottom_limit()	14
2.5.2.8 lss1_beyond_top_limit()	15
2.5.2.9 move_arm_north()	15
2.5.2.10 move_rail_system()	15
2.5.2.11 read_values_from_serial()	16
2.5.2.12 save_last_rail_position()	16
2.5.2.13 send_map_button_presses()	16
2.5.2.14 send_system_state_request()	17
2.5.2.15 set_boundaries_and_last_rail_position_data()	17
2.5.2.16 update_simultaneous_button_press_conditions()	17
2.6 map_and_path_center.mapper.Mapper	18
2.6.1 Detailed Description	18
2.6.2 Constructor & Destructor Documentation	18
2.6.2.1 __init__()	18
2.6.3 Member Function Documentation	18
2.6.3.1 button_press_callback()	19
2.6.3.2 init_communication()	19
2.6.3.3 joint_angles_callback()	19
2.6.3.4 pub_mapping_done()	20
2.6.3.5 save_to_file()	20
2.6.3.6 set_mapping_data()	20
2.6.3.7 start_mapping()	20
2.6.3.8 state_callback()	21
2.7 hardware_center.motor_control.motorControl	21
2.7.1 Detailed Description	22
2.7.2 Constructor & Destructor Documentation	22
2.7.2.1 __init__()	23
2.7.3 Member Function Documentation	23
2.7.3.1 calc_angle()	23
2.7.3.2 calc_position()	23
2.7.3.3 check_state_and_stop()	23
2.7.3.4 control_servo_speed()	24
2.7.3.5 follow_path()	24
2.7.3.6 follow_path_callback()	24
2.7.3.7 init_communication()	25
2.7.3.8 init_servo_threads()	25
2.7.3.9 limp_and_set_origin()	25
2.7.3.10 load_and_set_boundaries()	25
2.7.3.11 manauly_callback()	26
2.7.3.12 publish_joint_angles()	26

---

2.7.3.13 <code>read_angles_callback()</code>	26
2.7.3.14 <code>servo_control_loop()</code>	26
2.7.3.15 <code>setup_servos_and_queues()</code>	27
2.7.3.16 <code>start_read_callback()</code>	27
2.7.3.17 <code>state_callback()</code>	27
2.7.4 Member Data Documentation	27
2.7.4.1 <code>completed_movements</code>	28
2.7.4.2 <code>lock</code>	28
2.7.4.3 <code>threads</code>	28
2.8 <code>control_center.state_manager.StateManager</code>	29
2.8.1 Detailed Description	29
2.8.2 Member Function Documentation	29
2.8.2.1 <code>handle_state_change()</code>	29
2.8.2.2 <code>publish_state()</code>	29

# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">control_center.command_executor.CommandExecutor</a>	A ROS2 Node for executing received commands . . . . .	3
<a href="#">control_center.command_window.CommandPublisher</a>	A ROS2 Node that publishes commands from a Tkinter GUI . . . . .	4
<a href="#">control_center.display.DisplayNode</a>	A ROS2 Node for visualizing robot arm mappings . . . . .	6
<a href="#">hardware_center.gpio_controller.GPIOController</a>	Used to send state to arduino that then powers LED lights with correct state colour . . . . .	8
<a href="#">hardware_center.hardware_interface_controller.HardwareInterfaceController</a>	The <a href="#">HardwareInterfaceController</a> node is designed to handle user inputs from the controller, publish state requests to the state_manager node and map button presses to the mapper node, activate the rail system servo, and move the others based on the received joystick inputs . . . . .	10
<a href="#">map_and_path_center.mapper.Mapper</a>	Used to set nullpoints, boundarys and a path that the arm can follow . . . . .	18
<a href="#">hardware_center.motor_control.motorControl</a>	Used to controll and read values form LSS servo motor during mapping and path execution . . . . .	21
<a href="#">control_center.state_manager.StateManager</a>	The <a href="#">StateManager</a> node is designed to handle state requests from the HardwareInterfaceController node and confirmation messages from the Mapper, motorControl and Path node, and publish the current system state on the ROS topic: system_state . . . . .	29

# Chapter 2

## Class Documentation

### 2.1 control\_center.command\_executor.CommandExecutor Class Reference

A ROS2 Node for executing received commands.

#### Public Member Functions

- def `__init__` (self)
- def `command_callback` (self, msg)  
*Callback function for the command subscription.*
- def `process_command` (self, command)  
*Process the received command and execute the corresponding action.*

#### Public Attributes

- `subscription`  
*Subscription to the 'command\_topic'.*
- `start_mapping_pub`  
*Publisher for starting mapping.*
- `start_test_pub`  
*Publisher for starting test.*

#### 2.1.1 Detailed Description

A ROS2 Node for executing received commands.

#### 2.1.2 Constructor & Destructor Documentation

### 2.1.2.1 `__init__()`

```
def control_center.command_executor.CommandExecutor.__init__ (
    self )
```

Initialize the CommandExecutor node

## 2.1.3 Member Function Documentation

### 2.1.3.1 `command_callback()`

```
def control_center.command_executor.CommandExecutor.command_callback (
    self,
    msg )
```

Callback function for the command subscription.

Parameters

<code>msg</code>	The received command message
------------------	------------------------------

### 2.1.3.2 `process_command()`

```
def control_center.command_executor.CommandExecutor.process_command (
    self,
    command )
```

Process the received command and execute the corresponding action.

Parameters

<code>command</code>	The command to process
----------------------	------------------------

Referenced by `control_center.command_executor.CommandExecutor.command_callback()`.

## 2.2 `control_center.command_window.CommandPublisher` Class Reference

A ROS2 Node that publishes commands from a Tkinter GUI.

## Public Member Functions

- def [\\_\\_init\\_\\_](#) (*self*)
- def [send\\_command](#) (*self*, *command*)  
*Send a command to the 'command\_topic'.*

## Public Attributes

- [publisher](#)  
*Publisher for the 'command\_topic'.*

### 2.2.1 Detailed Description

A ROS2 Node that publishes commands from a Tkinter GUI.

### 2.2.2 Constructor & Destructor Documentation

#### 2.2.2.1 [\\_\\_init\\_\\_\(\)](#)

```
def control_center.command_window.CommandPublisher.__init__ (
    self )
```

Initialize the CommandPublisher node

### 2.2.3 Member Function Documentation

#### 2.2.3.1 [send\\_command\(\)](#)

```
def control_center.command_window.CommandPublisher.send_command (
    self,
    command )
```

Send a command to the 'command\_topic'.

#### Parameters

<i>command</i>	The command to publish
----------------	------------------------

## 2.3 control\_center.display.DisplayNode Class Reference

A ROS2 Node for visualizing robot arm mappings.

### Public Member Functions

- def `__init__` (self)
- def `read_mapping` (self, filename)  
*Read the mapping from a JSON file.*
- def `target_point_callback` (self, msg)  
*Callback function for the target point subscription.*
- def `display_callback` (self, msg)  
*Callback function for the display data subscription.*
- def `animate` (self, i)  
*Animate the plot with the updated data.*

### Public Attributes

- `mapping`  
*Load the initial work area mapping from the JSON file.*
- `subscription`  
*Subscription to the 'display\_data' topic.*
- `target_point_subscription`  
*Subscription to the 'target\_point' topic.*
- `ax`  
*Matplotlib figure and axis.*
- `inside_scatter`  
*Initialize the plot elements with None, they will be created in `animate()`*

### 2.3.1 Detailed Description

A ROS2 Node for visualizing robot arm mappings.

### 2.3.2 Constructor & Destructor Documentation

#### 2.3.2.1 `__init__()`

```
def control_center.display.DisplayNode.__init__ (
    self )
```

Initialize the DisplayNode

### 2.3.3 Member Function Documentation

#### 2.3.3.1 animate()

```
def control_center.display.DisplayNode.animate (
    self,
    i )
```

Animate the plot with the updated data.

##### Parameters

<i>i</i>	The frame index
----------	-----------------

#### 2.3.3.2 display\_callback()

```
def control_center.display.DisplayNode.display_callback (
    self,
    msg )
```

Callback function for the display data subscription.

##### Parameters

<i>msg</i>	The received message containing the display data
------------	--

#### 2.3.3.3 read\_mapping()

```
def control_center.display.DisplayNode.read_mapping (
    self,
    filename )
```

Read the mapping from a JSON file.

##### Parameters

<i>filename</i>	The filename of the JSON file
-----------------	-------------------------------

##### Returns

The mapping data

### 2.3.3.4 target\_point\_callback()

```
def control_center.display.DisplayNode.target_point_callback (
    self,
    msg )
```

Callback function for the target point subscription.

#### Parameters

<i>msg</i>	The received message containing the target point
------------	--

## 2.4 hardware\_center.gpio\_controller.GPIOController Class Reference

Used to send state to arduino that then powers LED lights with correct state colour.

### Public Member Functions

- def [\\_\\_init\\_\\_](#) (self)
- def [setup\\_gpio](#) (self)
- def [handle\\_state\\_update](#) (self, msg)
- def [set\\_led\\_state](#) (self, red, green, blue)
- def [on\\_shutdown](#) (self)

### 2.4.1 Detailed Description

Used to send state to arduino that then powers LED lights with correct state colour.

### 2.4.2 Constructor & Destructor Documentation

#### 2.4.2.1 \_\_init\_\_()

```
def hardware_center.gpio_controller.GPIOController.__init__ (
    self )
```

Initialize the GPIOController node. Set up a subscription to listen for state updates from the action controller.

### 2.4.3 Member Function Documentation

#### 2.4.3.1 handle\_state\_update()

```
def hardware_center.gpio_controller.GPIOController.handle_state_update (
    self,
    msg )
```

Change LED states based on the received state from stateManager.

Referenced by hardware\_center.gpio\_controller.GPIOController.\_\_init\_\_().

#### 2.4.3.2 on\_shutdown()

```
def hardware_center.gpio_controller.GPIOController.on_shutdown (
    self )
```

Clean up the GPIO settings to ensure the GPIO pins are left in a safe state when the program terminates.

#### 2.4.3.3 set\_led\_state()

```
def hardware_center.gpio_controller.GPIOController.set_led_state (
    self,
    red,
    green,
    blue )
```

output signal through the GPIO pins based on the provided boolean values for red, green, and blue. The arduino reads these signals and shows the correct colour.

Referenced by hardware\_center.gpio\_controller.GPIOController.handle\_state\_update(), and hardware\_center.gpio\_controller.GPIOController.setup\_gpio().

#### 2.4.3.4 setup\_gpio()

```
def hardware_center.gpio_controller.GPIOController.setup_gpio (
    self )
```

Configure the GPIO settings for the Raspberry Pi. Set up three LED pins (red, green, blue) for output and initialize them to a standby state.

Referenced by hardware\_center.gpio\_controller.GPIOController.\_\_init\_\_().

## 2.5 hardware\_center.hardware\_interface\_controller.HardwareInterfaceController Class Reference

The `HardwareInterfaceController` node is designed to handle user inputs from the controller, publish state requests to the state\_manager node and map button presses to the mapper node, activate the rail system servo, and move the others based on the received joystick inputs.

### Public Member Functions

- def `set_boundaries_and_last_rail_position_data` (self)
- def `save_last_rail_position` (self)
- def `lss1_beyond_bottom_limit` (self)
- def `lss1_beyond_top_limit` (self)
- def `clear_button_press_flags` (self)
- def `update_simultaneous_button_press_conditions` (self)
- def `read_values_from_serial` (self)
- def `move_arm_north` (self)
- def `control_arm_with_joystick` (self)
- def `check_state_callback` (self, msg)
- def `cleanup_serial` (self)
- def `send_map_button_presses` (self)
- def `send_system_state_request` (self, request)
- def `calculate_time_to_run_rail_system` (self, rpm)
- def `move_rail_system` (self, rpm, target\_pos)
- def `activate_rail_system` (self)

### Public Attributes

- `current_system_state`  
*State variable.*
- `previous_system_state`  
*State variable.*
- `state_subscriber`  
*Subscriber.*
- `send_map_button_press_publisher`  
*Publisher.*
- `set_system_state_by_request_publisher`  
*Publisher.*
- `joystick_button_pressed`  
*Attribute related to joystick button pressing.*
- `reset_button_pressed`  
*Attribute related to reset/"stop the active process" button pressing.*
- `run_predefined_path_button_pressed`  
*Attribute related to the "run a predefined path" button pressing.*
- `map_button_pressed`  
*Attribute related to "mapping" button pressing.*
- `CST_LSS_Port`  
*Port variable for the communication between the raspi, the LSS adapter board and the servo engines.*
- `ser_obj_controller`  
*Serial object to establish communication with the controller.*

- [CST\\_LSS\\_Baud](#)  
*Baudrate for the communication between the raspi, the LSS adapter board and the servo engines.*
- [diagonal\\_threshold](#)  
*Threshold to create ranges for the received analog values.*
- [zero\\_value\\_deadzone](#)  
*Deadzone to account for imprecise analog values, when they should hypothetically be 0*
- [diameter\\_of\\_wheel](#)  
*Diameter of the smart servo wheel, or the bigger gear with gearing, to be used in the calculation of the time to activate the rail system.*
- [distance\\_to\\_travel](#)  
*The distance to travel on the rail system.*
- [position\\_multiplier](#)  
*Variable to multiply with the angles in degrees stored in the mapping data of the JSON file.*
- [Iss0](#)  
*Object for the LSS engine with ID: 0.*
- [Iss1](#)  
*Object for the LSS engine with ID: 1.*
- [Iss2](#)  
*Object for the LSS engine with ID: 2.*
- [rail\\_position](#)  
*Variable to hold the last rail position.*
- [boundaries\\_and\\_last\\_rail\\_position\\_data](#)  
*Variable to store the mapping and last rail position data from the JSON file.*
- [simultaneous\\_button\\_press\\_conditions](#)  
*Array that contain every simultaneous button press condition.*
- [standby\\_logger\\_printed](#)  
*Variable to ensure the standby message is only printed once during the standby state.*

## 2.5.1 Detailed Description

The [HardwareInterfaceController](#) node is designed to handle user inputs from the controller, publish state requests to the state\_manager node and map button presses to the mapper node, activate the rail system servo, and move the others based on the received joystick inputs.

## 2.5.2 Member Function Documentation

### 2.5.2.1 activate\_rail\_system()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.activate_rail_system (
    self )
```

Activates the rail system and sets the appropriate parameters for the move\_rail\_system() function based on the value of the rail\_position variable.

After reaching its target position, the function calls the send\_system\_state\_request() method with the string: joystick\_arm\_control to set the system state back to joystick\_arm\_control.

### 2.5.2.2 calculate\_time\_to\_run\_rail\_system()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.calculate_time_to_run_rail_system (
    self,
    rpm )
```

Calculates and returns the period in seconds to run the rail system, based on a given rpm and the values of the circumference\_of\_wheel and the distance\_to\_travel variable.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.move\_rail\_system().

### 2.5.2.3 check\_state\_callback()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.check_state_callback (
    self,
    msg )
```

Callback function of the state subscriber.  
Assigns the value of the current\_system\_state to the previous\_system\_state variable before assigning the value of the message to the current\_system\_state variable

### 2.5.2.4 cleanup\_serial()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.cleanup_serial (
    self )
```

Closes the communication with the controller to clean up resources. Called at the except of the try-except block.

### 2.5.2.5 clear\_button\_press\_flags()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.clear_button_press_flags (
    self )
```

Resets button press flags.  
Utilized as a safety check, after multiple buttons were pressed, to ensure that no high signals are stored.  
Used in favor of the reset input buffer to keep the analog value readings in the buffer.

### 2.5.2.6 control\_arm\_with\_joystick()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.control_arm_←
with_joystick (
    self )
```

Function utilized to receive analog values and call the correct directional functions, while handling various button presses and perform the appropriate actions.

### 2.5.2.7 lss1\_beyond\_bottom\_limit()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.lss1_beyond_←
bottom_limit (
    self )
```

Compares the smart servo's position with the assigned bottom limit. If the active position is lower than the bottom limit it returns true, else it returns false.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.move\_arm\_north().

### 2.5.2.8 lss1\_beyond\_top\_limit()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.lss1_beyond_←
top_limit (
    self )
```

Compares the smart servo's position with the assigned top limit. If the active position is greater than the bottom limit it returns true, else it returns false.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.move\_arm\_north().

### 2.5.2.9 move\_arm\_north()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.move_arm_north (
    self )
```

Moves the robot towards the north.  
Evaluates the current system state and performs the appropriate actions accordingly.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.control\_arm\_with\_joystick().

### 2.5.2.10 move\_rail\_system()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.move_rail_system
(
    self,
    rpm,
    target_pos )
```

Utilizes a given rpm and a target position, and calls the calculate\_time\_to\_run\_rail\_system() method and the wheelRPM() function to physically move the rail system for the correct period.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.activate\_rail\_system().

### 2.5.2.11 read\_values\_from\_serial()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.read_values_from_serial (
    self )
```

Reads values from the serial port connected to the controller.  
Adds the appropriate values to their designated variables.

### 2.5.2.12 save\_last\_rail\_position()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.save_last_rail_position (
    self )
```

Save the last rail position to the correct field of the variable, in which the JSON file was originally stored, to ensure that no other field gets overwritten. Dump the new data variable to the original JSON file.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.activate\_rail\_system().

### 2.5.2.13 send\_map\_button\_presses()

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.send_map_button_presses (
    self )
```

Creates a message indicating that the map button was pressed, and calls the publish() method of the send\_map\_button\_press publisher to forward the message onto the ROS topic: map\_button\_pressed

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.control\_arm\_with\_joystick().

**2.5.2.14 send\_system\_state\_request()**

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.send_system_state_request (
    self,
    request )
```

Creates a message with the request parameter, and calls the publish() method of the set\_system\_state\_by\_request\_publisher to forward the message onto the ROS topic: system\_state\_request

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.activate\_rail\_system().

**2.5.2.15 set\_boundaries\_and\_last\_rail\_position\_data()**

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.set_boundaries_and_last_rail_position_data (
    self )
```

Load the boundaries and last rail position data from the stored JSON file to a data variable.  
Use this variable to store the boundaries and last rail position to their appropriate variables for further use.

**2.5.2.16 update\_simultaneous\_button\_press\_conditions()**

```
def hardware_center.hardware_interface_controller.HardwareInterfaceController.update_simultaneous_button_press_conditions (
    self )
```

Updates the simultaneous button press conditions array with every possible button combination. Utilized in the read\_values\_from\_serial(self) function to updated each individual case according to the user inputs.

Referenced by hardware\_center.hardware\_interface\_controller.HardwareInterfaceController.read\_values\_from\_serial().

**2.6 map\_and\_path\_center.mapper.Mapper Class Reference**

Used to set nullpoints, boundarys and a path that the arm can follow.

## Public Member Functions

- def `__init__` (self)
- def `init_communication` (self)
- def `state_callback` (self, msg)
- def `start_mapping` (self)
- def `button_press_callback` (self)
- def `joint_angles_callback` (self, msg)
- def `pub_mapping_done` (self)
- def `set_mapping_data` (self)
- def `save_to_file` (self)

### 2.6.1 Detailed Description

Used to set nullpoints, boundarays and a path that the arm can follow.

The [Mapper](#) works communicates with MotorController to achive this funcitonality, as the MotorController can read angels of servos

The data generated wil be stored in a JSON file that is used during joystick controll and path execution.

### 2.6.2 Constructor & Destructor Documentation

#### 2.6.2.1 `__init__()`

```
def map_and_path_center.mapper.Mapper.__init__ (
    self )
```

Initialize the Mapper node with mapping settings.

### 2.6.3 Member Function Documentation

#### 2.6.3.1 `button_press_callback()`

```
def map_and_path_center.mapper.Mapper.button_press_callback (
    self )
```

Respond to button press events for mapping.  
Button press index is incresed by 1 and reading angles is requested.

Referenced by `map_and_path_center.mapper.Mapper.init_communication()`.

### 2.6.3.2 init\_communication()

```
def map_and_path_center.mapper.Mapper.init_communication (
    self )
```

Initialize communication for the node.  
Setup all necessary publishers and subscribers for sending and receiving commands  
and data to and from other components of ROS system.

Referenced by map\_and\_path\_center.mapper.Mapper.\_\_init\_\_().

### 2.6.3.3 joint\_angles\_callback()

```
def map_and_path_center.mapper.Mapper.joint_angles_callback (
    self,
    msg )
```

When the read joint angles are sent back, this function runs,  
saving the angles as either boundaries or path points,  
depending on what button\_press\_index is.

Referenced by map\_and\_path\_center.mapper.Mapper.init\_communication().

### 2.6.3.4 pub\_mapping\_done()

```
def map_and_path_center.mapper.Mapper.pub_mapping_done (
    self )
```

This will publish to stateManager that the mapping is done

Referenced by map\_and\_path\_center.mapper.Mapper.joint\_angles\_callback().

### 2.6.3.5 save\_to\_file()

```
def map_and_path_center.mapper.Mapper.save_to_file (
    self )
```

Save the collected mapping data to a JSON file.

Referenced by map\_and\_path\_center.mapper.Mapper.joint\_angles\_callback().

### 2.6.3.6 set\_mapping\_data()

```
def map_and_path_center.mapper.Mapper.set_mapping_data (
    self )
```

Load existing mapping data from JSON file if it exist

Referenced by map\_and\_path\_center.mapper.Mapper.\_\_init\_\_().

### 2.6.3.7 start\_mapping()

```
def map_and_path_center.mapper.Mapper.start_mapping (
    self )
```

Begin the mapping process by making the motors limp, this is done by sending a message to MotorController. After 25 sec the MotorController has set nullpoints, and the operator is prompted to move the arm to first set

Referenced by map\_and\_path\_center.mapper.Mapper.state\_callback().

### 2.6.3.8 state\_callback()

```
def map_and_path_center.mapper.Mapper.state_callback (
    self,
    msg )
```

Handle updates to the state received from the system\_state topic.

Referenced by map\_and\_path\_center.mapper.Mapper.init\_communication().

## 2.7 hardware\_center.motor\_control.motorControl Class Reference

Used to controll and read values form LSS servo motor during mapping and path execution.

## Public Member Functions

- def `__init__` (self)
- def `init_communication` (self)
- def `load_and_set_boundaries` (self)
- def `setup_servos_and_queues` (self)
- def `init_servo_threads` (self)
- def `servo_control_loop` (self, servo\_key)
- def `control_servo_speed` (self, servo\_key, target\_angle)
- def `follow_path_callback` (self, msg)
- def `follow_path` (self, angles, num\_cycles)
- def `limp_and_set_origin` (self, msg)
- def `publish_joint_angles` (self, servo\_key, angle)
- def `calc_position` (self, angle)
- def `calc_angle` (self, position)
- def `read_angles_callback` (self)
- def `manually_callback` (self)
- def `start_read_callback` (self, msg)
- def `check_state_and_stop` (self)
- def `state_callback` (self, msg)

## Public Attributes

- `real_time_qos`  
*Define a QoS profile for real-time updates.*
- `boundaries`  
*Initialize attributes and subscriptions.*
- `threads`  
*Init empty boundary dictionary.*
- `lock`  
*Initialize array for servo thread.*
- `state`  
*create varialbe for locking threads*
- `total_movements`  
*Initialize system state for this node as 'standby'.*
- `completed_movements`  
*Initialize variable.*
- `current_movement_nr`  
*Initialize variable.*
- `actual_joint_angles_publisher`  
*publish the read angels, used during mapping*
- `joint_angles_publisher`  
*Topic to publish the joint angles to display node, its not used as we didnt get the display node to work properly.*
- `start_read_sub`  
*Subscriber for receiving start test command.*
- `limp_and_reset_origin_sub`  
*Subscriber for making the arm limp and setting nullpoints, used during mapping.*
- `read_angels_sub`  
*Subscriber for initaiting reading of angles.*
- `joint_angles_subscription`  
*Subscriber used for reciving angle array from PathExecutor, aslo initiates path execution.*
- `path_done_pub`  
*Publisher for updating the stateManager when the path is finished.*
- `state_subscription`  
*Subscriber for getting system state updates.*

## 2.7.1 Detailed Description

Used to control and read values from LSS servo motor during mapping and path execution.

The `motorControl` node is designed to interface with LSS servos and manage their operations through various topics. This node handles real-time control, state management, and boundary safety for motor operations within this ROS system.

## 2.7.2 Constructor & Destructor Documentation

### 2.7.2.1 `__init__()`

```
def hardware_center.motor_control.motorControl.__init__ (
    self )
```

Initialize the motor control node, setting up publishers, subscribers, and servo control threads.

## 2.7.3 Member Function Documentation

### 2.7.3.1 `calc_angle()`

```
def hardware_center.motor_control.motorControl.calc_angle (
    self,
    position )
```

Calculate the joint angle based on the servo position reading. `position` is LSS servomotors way of describing a joint's position. Converts raw position data from the servo to a human-readable angle format.

Referenced by `hardware_center.motor_control.motorControl.control_servo_speed()`, `hardware_center.motor_control.motorControl.limp_and_set_origin()`, and `hardware_center.motor_control.motorControl.read_angles_callback()`.

### 2.7.3.2 `calc_position()`

```
def hardware_center.motor_control.motorControl.calc_position (
    self,
    angle )
```

Calculate the joint position. `position` is LSS servomotors way of describing angle, `position = angle*10` this is used when moving the servos

### 2.7.3.3 check\_state\_and\_stop()

```
def hardware_center.motor_control.motorControl.check_state_and_stop (
    self )
```

Check the system's state and stop all servo movements if necessary.

sadly we couldnt get the node to recive messages whils operating the servos, and therefor this funciton is use

### 2.7.3.4 control\_servo\_speed()

```
def hardware_center.motor_control.motorControl.control_servo_speed (
    self,
    servo_key,
    target_angle )
```

Control the speed of a servo to reach a specified target angle.

Implements PID control to reach the target angle by comparing current and target angle.

Referenced by hardware\_center.motor\_control.motorControl.servo\_control\_loop().

### 2.7.3.5 follow\_path()

```
def hardware_center.motor_control.motorControl.follow_path (
    self,
    angles,
    num_cycles )
```

Reads the target angles and puts them in the queue for independent servo threads to read  
Keeps track of movement iteration and correctly ending a path execution

- explanation of reverse for loop -

len(angles) - 2: This starts the index from the second-to-last element in the angles list.  
It adjusts for Python's zero-based indexing, ensuring the arm starts reversing from the correct position.

-1: This is the end parameter of the range, which is exclusive. In Python's range, when counting backwards, you need to go one past the first index you want to include, hence -1 is used to ensure it includes the index

-2: This is the step parameter, which determines the increment between each step in the range.  
Here, -2 means the loop decrements the index by 2 each time,

effectively moving backwards through the angles list two elements at a time.  
This allows the system to address pairs of angles (assuming each servo position in the pair is spaced by one index in the list) in reverse order

Referenced by hardware\_center.motor\_control.motorControl.follow\_path\_callback().

### 2.7.3.6 follow\_path\_callback()

```
def hardware_center.motor_control.motorControl.follow_path_callback (
    self,
    msg )
```

Callback for moving servos according to a set of predefined angles.  
Handles the reception of an array of angles,  
setting the path for the servos to follow,  
and number of times to iterate over the path.

### 2.7.3.7 init\_communication()

```
def hardware_center.motor_control.motorControl.init_communication (
    self )
```

Initialize communication for the node.  
Setup all necessary publishers and subscribers for sending and receiving commands  
and data to and from other components of ROS system.

Referenced by map\_and\_path\_center.mapper.Mapper.\_\_init\_\_().

### 2.7.3.8 init\_servo\_threads()

```
def hardware_center.motor_control.motorControl.init_servo_threads (
    self )
```

Initialize threads for controlling servos independently.

### 2.7.3.9 limp\_and\_set\_origin()

```
def hardware_center.motor_control.motorControl.limp_and_set_origin (
    self,
    msg )
```

Makes the servos go limp and sets a new origin after a delay.  
Used at the start of mapping process

### 2.7.3.10 load\_and\_set\_boundaries()

```
def hardware_center.motor_control.motorControl.load_and_set_boundaries (
    self )
```

Load and set boundaries for servo movements from a JSON configuration file.  
Ensures that servos operate within safe operational limits to avoid mechanical damage.

### 2.7.3.11 manualy\_callback()

```
def hardware_center.motor_control.motorControl.manualy_callback (
    self )
```

Read the servos when during testing and debugging

Referenced by hardware\_center.motor\_control.motorControl.start\_read\_callback().

### 2.7.3.12 publish\_joint\_angles()

```
def hardware_center.motor_control.motorControl.publish_joint_angles (
    self,
    servo_key,
    angle )
```

Publish the current joint angles to display node for visualization, sadly didnt get it working.

### 2.7.3.13 read\_angles\_callback()

```
def hardware_center.motor_control.motorControl.read_angles_callback (
    self )
```

Read the servos when during mapping, is called when arm state='map' and map button is pressed

### 2.7.3.14 servo\_control\_loop()

```
def hardware_center.motor_control.motorControl.servo_control_loop (
    self,
    servo_key )
```

Control loop for servos.  
Handles target angles for a single servo from its angles queue.

Referenced by hardware\_center.motor\_control.motorControl.init\_servo\_threads().

### 2.7.3.15 setup\_servos\_and\_queues()

```
def hardware_center.motor_control.motorControl.setup_servos_and_queues (
    self )
```

Initializes servo objects and angles queues for asynchronous operation.

### 2.7.3.16 start\_read\_callback()

```
def hardware_center.motor_control.motorControl.start_read_callback (
    self,
    msg )
```

callback for starting reading angles used during testing

### 2.7.3.17 state\_callback()

```
def hardware_center.motor_control.motorControl.state_callback (
    self,
    msg )
```

Callback for updating the system's operational state.  
Receives state updates from the action controller.

Referenced by map\_and\_path\_center.mapper.Mapper.init\_communication().

## 2.7.4 Member Data Documentation

#### 2.7.4.1 completed\_movements

```
hardware_center.motor_control.motorControl.completed_movements
```

Initialize variable.

Clear the angles queue.

Forward direction.

Wait for **all** movements to complete before reversing

Reverse direction

Wait for **all** movements to complete before next cycle

Reset movements counters after completion of **all** cycles

Stop the servo

Referenced by hardware\_center.motor\_control.motorControl.check\_state\_and\_stop(), hardware\_center.motor\_control.motorControl.follow\_path(), and hardware\_center.motor\_control.motorControl.servo\_control\_loop().

#### 2.7.4.2 lock

```
hardware_center.motor_control.motorControl.lock
```

Initialize array for servo thread.

Initialize servo threads

Initialize communication

Referenced by hardware\_center.motor\_control.motorControl.control\_servo\_speed(), and hardware\_center.motor\_control.motorControl.servo\_control\_loop().

#### 2.7.4.3 threads

```
hardware_center.motor_control.motorControl.threads
```

Init empty boundary dictionary.

Initialize loading boundaries form JSON file

Initialize servos objects and quese for target angles

## 2.8 control\_center.state\_manager.StateManager Class Reference

The [StateManager](#) node is designed to handle state requests from the HardwareInterfaceController node and confirmation messages from the Mapper, motorControl and Path node, and publish the current system state on the ROS topic: `system_state`.

### Public Member Functions

- def [publish\\_state](#) (`self`)
- def [handle\\_state\\_change](#) (`self, msg`)

### Public Attributes

- [state](#)  
*Official state variable.*
- [state\\_publisher](#)  
*State publisher.*

#### 2.8.1 Detailed Description

The [StateManager](#) node is designed to handle state requests from the HardwareInterfaceController node and confirmation messages from the Mapper, motorControl and Path node, and publish the current system state on the ROS topic: `system_state`.

#### 2.8.2 Member Function Documentation

##### 2.8.2.1 [handle\\_state\\_change\(\)](#)

```
def control_center.state_manager.StateManager.handle_state_change (
    self,
    msg )
```

Handles the value of the messages received on the ROS topic: `system_state_request`, and performs the appropriate actions according to our desired architecture. The `publish_state()` method is called to notify other nodes of the current state change.

##### 2.8.2.2 [publish\\_state\(\)](#)

```
def control_center.state_manager.StateManager.publish_state (
    self )
```

Adds the content of the state variable to a message, which is forwarded onto the ROS topic: `system_state`.

Referenced by `control_center.state_manager.StateManager.handle_state_change()`.