Table of contents

1.0 Introduction

The MACHINETRON defends the known space against the threat of floral foam. By combining four submachines, the handler, mill, drill and lathe, the MACHINETRON becomes a powerful tool capable of processing raw floral foam into prescribed shapes.

This report details the engineering development process and analysis for the MACHINETRON's handler submachine, in addition to its scope, objectives and requirements.

1.1 Scope

The handler fulfils the following tasks:

- Power and data distribution
- Movement the foam block between the other submachines
- Rotation of the foam block
- Gripping the foam block
- Software and tool pathing implementation, including status control

1.2 Objectives and requirements

The objectives and requirements are broken down as follows:

- Linear actuation of the foam block along the y axis perpendicular to the other submachines
- Rotation along the z and y axes to provide submachines access to all five faces of the foam block (appendix K1 shows the handler's axes)
- Send power and data to the other submachines through two power lines and two data lines
- Grip the foam block in place for the other submachines to process the foam block
- Show its status through three status LEDs and start and stop operation as required
- Generate toolpaths and coordinate points, and distribute them to the other submachines
- Integrate with the other submachines, operating within the other submachine constraints
- Use at least one custom PCB and one machined part
- Meet an overall low compliance build to allow machining of piece within 0.5mm accuracy
- Meet all OHS requirements

Figure 1 in appendix A shows the casual dependency of the handler subsystem. The diagram shows the steps required from the handler to process a foam block. The handler implements software toolpaths which are sent to the PCB, which are then distributed to the other submachines, in addition to power. The handler shall then actuate the foam block to the other submachines and rotate and hold the block as required.

2.0 MACHINETRON handler design overview

The handler has a linear actuation system and two rotational axes (Figure 4). The linear motion moves the foam block between the other submachines, controlling all y axis linear actuation. The two rotational axes give each submachine access to all fives faces of the block without the submachines requiring any rotational axis (Aside from the drill and mill machining tools).

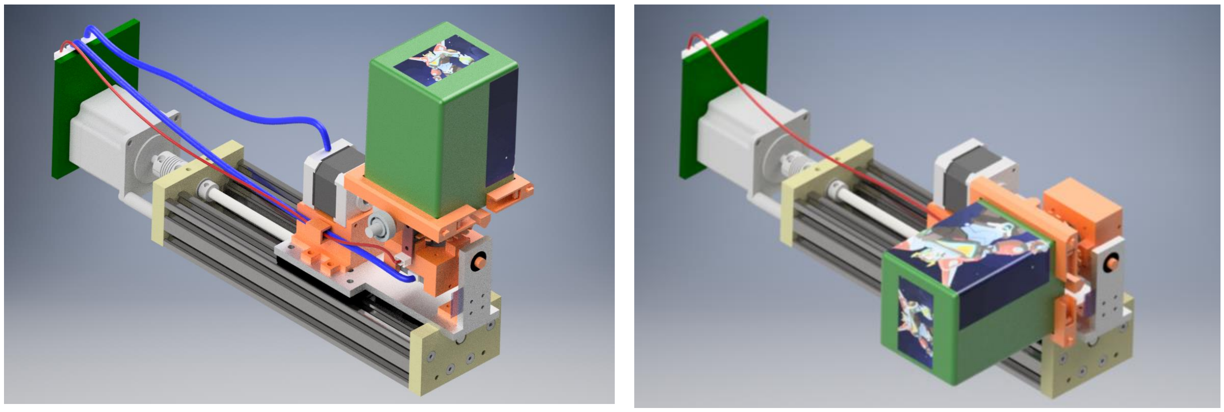


Figure 4: Handler assembly upright (left) and rotated (right)

The linear actuator uses a lead screw actuation system in which a gantry plate is mounted (Figure 5 appendix A). The gantry plate slides alongside a C-beam using four wheels, on which the rotational mechanisms of the handler are mounted. The lead screw was chosen for high accuracy and to give precise positioning with low backlash, compared to the main alternative considered: a belt system that has the advantage of speed and force [4], which were chosen to have lower priority than maintaining positional accuracy by the engineering team. The limitation of choosing a lead screw, over a similar ball screw mechanism is that greater torque is required, and greater friction present in the system [3]. Table 3 Appendix A shows the decision matrix between the linear actuation systems considered.

From equation 2 in appendix A, the accuracy of using the lead screw actuator was found to be ±0.01mm. This is better than the required machining accuracy of 0.5mm, thus a lead screw actuator was a suitable solution.

To confirm the linear actuator would be able to actuate the required load of the handler's rotational systems, a simulation was done in inventor to ensure there would be minimal compliance. Figures 6-8 Appendix B detail the simulation results on the gantry plate. The simulation shows that there is a high safety factor of 15, and acceptable stress levels (-0.1947MPa compressive max) located where the bolts for the wheels are mounted to the gantry plate, with no large deflection (1.318E-4mm max) or stresses represented by red zones.

The handler platform (Figure 9 Appendix C) acts as the base for the rotational axis, as well as a quick release mechanism that is attached to the gantry plate using wing nuts, allowing it to be disassembled quickly. To design the handler platform and support, a high strength material that would minimize deflection, that has low mass was required. For the application of the handler's platform, strength was prioritized for minimal deflection. The material needed to be obtainable, and within budget constraints. Using the diagram in Figure 10, appendix C, an appropriate material was selected. From the diagram, aluminium was chosen. Aluminium is durable, low density (low weight), high strength and corrosion resistant. This would allow maintenance free and of the part and meet required specifications.

The material was machined using waterjet cutting. The advantage of waterjet cutting allowed highly accurate (0.025 mm tolerance) [2] cutting of the material using CAD, without material distortion, no heat affected zone or additional finishing process required [8].

Using Young's modulus of the material selected the platform's deflection for the handler platform was found to be 0.017782mm (See appendix C for calculation). The value for deflection is minimal and will not have a significant impact on the accuracy, making the material a suitable choice. An alternative material considered was ABS plastic because it was readily available, low cost and low weight. However, the deflection calculation for ABS plastic in Appendix C prove the material does not have a high enough modulus to prevent large deflection, and thus meet accuracy requirements of 0.5mm.

Further simulation was also done for deflection and stresses on the gripper from an external force on the block from the submachines, in addition its own weight (See Figures 12-14 appendix D). The simulation results show minimal deflection (0.04mm max on gripper), except on the foam block due to its weak material properties. The principal stresses show small stresses (14.96 MPa max) at the connection to the motor shaft; it is an acceptable stress represented by the light blue that is below the tensile strength of stainless steel which is 215 MPa. Furthermore, the safety factor shows very high margin of 15.

The handler is comprised of two parts and uses two mechanisms to hold the foam block (See Figure 15 appendix D for CAD model). On two sides of the gripper, there are devices inside for attaching springs, which give continuous force on the foam block as well as help keep the two parts of the gripper aligned. Additionally, the gripper has two holds for rubber bands which allow extra continuous force. Testing of the springs and rubber bands demonstrated the force provided by the springs and rubber bands was enough in holding the block in the gripper. One part of the gripper slides in and out along a plastic shaft to place and secure the block, while the other part remains stationary. Thus, the block is secured in the stationary part, which will align the centre of the block with the z rotation axis, in addition to allowing the block to be placed in the same position each time.

To attach the gripper to a stepper motor[1] (through a shaft coupler), stainless steel shafts (Figure 16, appendix D) were placed attached to the gripper and motor holder using epoxy Loctite: a high strength adhesive to form bonds between stainless steel and ABS plastic. A stainless-steel shaft prevented wear on the shaft from the grub screw when attaching to the shaft coupler, as well as provide a central high inertia to keep rotation of the gripper on axis. Furthermore, the high

[1] See section 6.0 for actuation

young's modulus of stainless steel will prevent it from being elastically deformed, which would lead to compliance in the position of the foam block (Appendix L1).

3.0 Position sensing

To maintain positional accuracy, the handler has two sensors. The first sensor is a mechanical switch located at the zero position of the y axis linear actuator, which allows resetting of the handler position to the origin. The second sensor is an optical end stop for rotational accuracy in the z axis. The optical end stop is mounted to the stepper motor bracket on the z axis, and a disk attached to the shaft coupler rotates through the optical end stop (See figure 3, appendix E). When the disc passes through, the end stop detects it using a by reading its signal pin in a timer, and the handler detects it has reached the zero position of the z axis. This gives the handler the ability to reset its position and maintain accuracy after lathing if motor steps are skipped.

4.0 MACHINETRON system status

The MACHINETRON uses a 48 MHZ timer (default maximum speed) to constantly check for status input from the handler's GO button. Once operation has begun, the red status LED will light up, and if the GO button is pressed while in operation, the light will change to an amber light, and the operation will be suspended on standby for the operation to be continued from where it left off by a longer push from the GO button. The code for the timer and GO button can be found in appendix J, and the state diagram for operation in appendix M.

5.0 PCB placement and quick release mechanism

The PCB for the handler controls the power and data lines for the other submachines, and thus a stationary placement at the linear actuator motor was chosen. This prevents the power lines from moving if it was to be attached to any part of the gantry plate. A hook and loop fastener (Velcro) were used to quickly assemble the PCBs as a quick release mechanism. The advantage of Velcro is that it is quick, easy to use, safe and maintenance free [10]. The alternative was to use a 3D printed mechanical hook, which would slot into a 3D printed PCB holder. The decision matrix in appendix E compares the two options.

6.0 MACHINETRON handler actuator selection

For rotating the foam block 90 degrees, stepper motors and servos were predominately compared. A prominent servo considered was MG955 tower pro servo. Figure 2 in Appendix A shows the decision matrix for selecting the appropriate actuator. The servo specified low power draw, high speed, very high torque and metal gearing, (See appendix A for specifications or [14]) however, a quantitative assessment of user reviews[1] and previous experience[2] (empirical analysis) with servos within the budget constraint showed them to be unreliable and inaccurate for factors such as not holding position under a load, not accurately actuating and poor solder joints leading to damage and becoming unusable. Instead stepper motors were chosen. Stepper motors could be obtained for low cost ($7.53-$8.32 for selected stepper motors), allowing budget to be allocated elsewhere. In addition, stepper motors output high torque (torque values below) for the handler's low speed rotations, including stability at standstill to hold the block with precise positioning (1.8° per step) and repeatability due to low error (3-5%) for each step [1].

For the stepper motor in the z axis, the motor required holding torque high enough to hold its position while being machined. All calculations can be found in appendix F. The torque required to accelerate to 150 RPM in 500ms was found to be 0.04021Ncm, a minimal value, and the holding torque while a submachine machined the foam block to be 20Ncm. A large consideration in selecting the stepper motor was its weight, as the greater the weight, the more torque required for the stepper motor in the other rotation axis would need to rotate the block with respect to the submachines. The stepper selected was a 17HS13-0404S1 NEMA 17 with a 34mm long body and a 26Ncm holding torque, a value greater than the worst-case requirement of 20Ncm [11].

The speed of the linear actuator stepper motor was set to 150RPM, or 2.5 RPS. (The max speed without a timer) The torque required was calculated in appendix F2 and found to be 5.18Ncm, and thus a short body 17HS08-1004S NEMA 17 x 20mm body was chosen [12] with a torque of roughly 11.7Ncm at 150 RPM. The selected motor meets the torque requirements with a safety factor of 2.27, as well as being the smallest NEMA 17 length, giving the advantage of a shorter machine length (box constraint). Alternative stepper motors compared were NEMA 14 steppers due to their smaller face size, however, body width was considered a higher priority for size constraint since the width of the c-beam end plate determined the minimum width of the linear actuator.

For the actuator that rotates with respect to the submachines, a torque requirement of 16Ncm was calculated (See appendix F3). Thus, a 17HS15-1504S1 NEMA 17 stepper motor with a 45Ncm holding torque was selected that has a safety factor of 2.8125 [13]. Since this axis does not use a sensor to detect its position, a high torque stepper was required, for a lower cost than a lower torque specified motor ($8.11 compared to $8.32 for 20mm body NEMA 17). Inventor simulations showed that motor mass or size were not a large consideration, and high holding torque was the highest priority (Appendix A and D simulations). All stepper motors were obtained from Stepper Online, and a comparison of other motors considered can be found in appendix P.

7.0 Power requirements and stepper drivers

The power requirement for the handler predominantly comes from stepper motor draw. The linear actuator uses a lead screw; it does not need to be powered unless it is in operation. However, the rotational motors must be powered throughout operation to hold position, at 4.8 and 5.175 watts (load is direct to axle, no transmission). None of the motors run at greater than 1.5A, thus A4988 stepper drivers were selected, which are rated up to 2A with sufficient cooling, such as a heat sink [15]. See appendix G for power calculations.

[1] See reference [6] and [8] for user reviews.

[2]ENGG1100, METR2800 and METR4202 projects.

8.0 Toolpath and g-code generation

To setup g-code, the machine part file was opened in Fusion 360, and using the manufacture CAM process, the submachine tool bits could be selected (See Figure 21 Appendix I). Once selected, manufacturing processes could be selected to machine the part. Fusion 360 can generate the toolpath, and which can then be simulated to confirm desired behaviour (Figure 22 appendix I). The process is then posted, which produces a text file with the g-code required, that can be sent to the handler to be distributed to the submachines and inputted into the actuation functions. The file contains coordinates in x, y and z domain with respect to the foam block.

9.0 MACHINETRON software process and submachine integration

The software flow chart in appendix J describes the software process. The handler has two main functions, which are to move to each submachine, and rotate and hold the foam block. Appendix J outlines an example of the rotation function, which keeps track of the block's face through the integer pointers *current_face* and *current_z_face*, and when called rotates to the appropriate face. Additionally, the appendix contains the function to spin the block for the lathe to cut. Appendix J shows the linear actuation function, which takes a position in mm from the actuator's origin located at the mechanical switch and compares it to its current position. It then actuates to the input position, returning the float value of its new position: *current_position*. Additional functions (not included to save trees) were also made to initialise stepper drivers, update the status LEDs, reset the horizontal actuation to a zero-position using a limit switch, and to reset the z axis rotation using the optical end stop.

The MACHINETRON uses UART communication between the submachines between two data lines to the TX and RX pins. UART was used for its simplicity, reliability, and ability to function at distances between submachines (Decision matrix Appendix N). Wi-Fi and Bluetooth were not used due to the cost of buying their modules which would use budget that could be allocated elsewhere. The handler distributes commands through three UART ports to the submachines and begins machining operation at the press of the handler's GO button.

10.0 Conclusion

The handler's total cost for all mechanical and electrical components was $85.62. The budget's breakdown for each component can be found in Appendix O. Due to unexpected failure of the MACHINETRON's electrical system, the micro controllers on all subsystems were damaged, which inhibited further testing and implementation of the MACHINETRON's advanced features. Future work on the handler subsystem could include diagnosing the fault that caused the microcontrollers to be damaged, and thus allow further calibration and testing of implementing g-code and automation of the process.

11.0 References

[1] ElProCus - Electronic Projects for Engineering Students. (2019). *The Stepper Motor Basics: Types, Working Operation and Applications*. [online] Available at: https://www.elprocus.com/stepper-motor-types-advantages-applications/

[2] Flowwaterjet.com. (2019). *Benefits of Waterjet Technology for Precision Cutting - Flow Waterjet*. [online] Available at: https://www.flowwaterjet.com/Learn/Benefits-of-Waterjet.aspx.

[3] Linear, H. (2019). THE ENGINEER’S GUIDE: LEAD SCREWS VS. BALL SCREWS. Presentation.

[4] Myostat.ca. (2019). *Ball Screw vs Belt Driven Actuators | Articles | Myostat Motion Control*. [online] Available at: http://www.myostat.ca/ball-screw-versus-belt-driven-actuators.

[5] Orientalmotor.com.sg. (2019). *How much accuracy if I am using Stepping Motor with the ballscrew?*. [online] Available at: https://www.orientalmotor.com.sg/qa_det/qa_stmotor05/.

[6] Pbclinear.com. (2019). *What is Lead Screw Efficiency?*. [online] Available at: https://www.pbclinear.com/Blog/What-is-Leadscrew-Efficiency.

[7] Rcmodelreviews.com. (2019). *Review: Towerpro/Hextronic MG995 servo*. [online] Available at: https://www.rcmodelreviews.com/mg995review.shtml.

[8] Resato.com. (2019). *Five benefits of waterjet cutting - Resato*. [online] Available at: https://www.resato.com/en/knowledge-center/blogs/five-benefits-of-waterjet-cutting.

[9] Servodatabase.com. (2019). *TowerPro MG995 Servo Specifications and Reviews*. [online] Available at: https://servodatabase.com/servo/towerpro/mg995.

[10] velcro fastener advantages and disadvantages. (2019). Retrieved from http://www.paxstrap.com/new/velcro-fastener-advantages-and-disadvantages.html

12.0 Datasheets

[11] **Nema 17 34mm body 17HS13-0404S1:**

https://www.omc-stepperonline.com/download/17HS13-0404S1.pdf

[12] **Nema 17 20mm body 17HS08-0404S1:**

https://www.omc-stepperonline.com/download/17HS08-1004S.pdf

[13] **Nema 17 39mm body 17HS15-0404S1:**

https://www.omc-stepperonline.com/download/17HS15-1504S1.pdf

[14] **MG955 Tower Pro servo**:

https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

[15] **A4988 stepper driver:**

https://www.pololu.com/product/1182

Appendix A: Handler overview

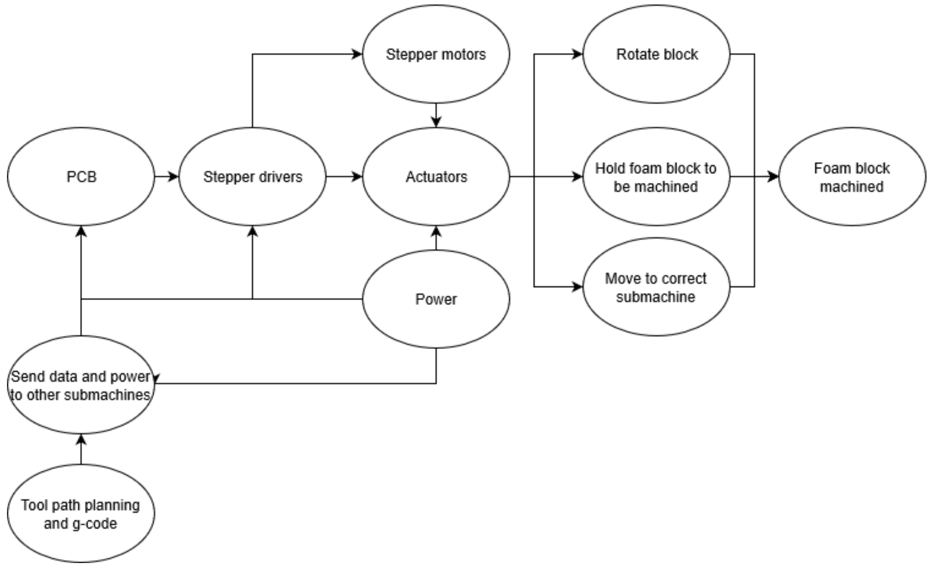


Figure 1: Handler's casual dependency

Linear Actuation and rotation motor decision matrix					
Motor	Precision	Torque	Speed	Cost	Total
Stepper	8	8	8	15	39
Servo	10*	10*	10*	1*	31
Brushless DC with encoder	10	10	10	3	33
Brushless DC Motor	1	5	10	10	26

Table 2: Linear actuation and rotation motor decision matrix

*For a high-quality servo that meets its advertised specifications

MG955 servo specifications:

Stall torque: 9.4kg/cm (4.8v); 11kg/cm (6v)

Operating speed: 0.20sec/60degree (4.8v); 0.16sec/60degree (6.0v)

Operating voltage: 4.8~ 6.6v

Gear Type: Metal gear

Temperature range: 0- 55deg

Dead band width: 1us

servo wire length: 32cm

Current draw at idle 10MA

No load operating current draw 170MA

Stall current draw 1200MA

Datasheet [12]:

https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

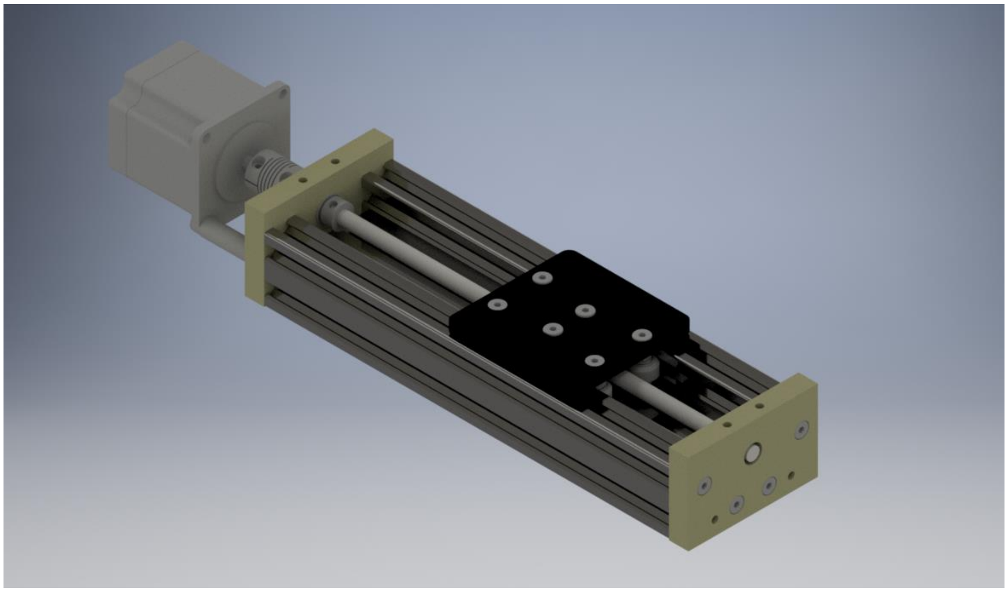


Figure 5: C-beam linear actuator

Linear actuation method					
Type	Accuracy	Size	Cost	Speed	Total
Lead screw	5	5	8	5	23
Ball screw	8	5	3	6	22
Belt drive	4	5	8	5	22

Table 3: Linear actuation decision matrix

lead pitch: 2mm (Chosen due to being easily attainable)
Accuracy at load side = (1.8 / 360°) x ballscrew pitch/lead
(±1.8°/360°) x 2mm = ±0.01 mm

Equation 2: lead screw accuracy equation [Oriental motors]

Appendix B: Linear actuation simulations

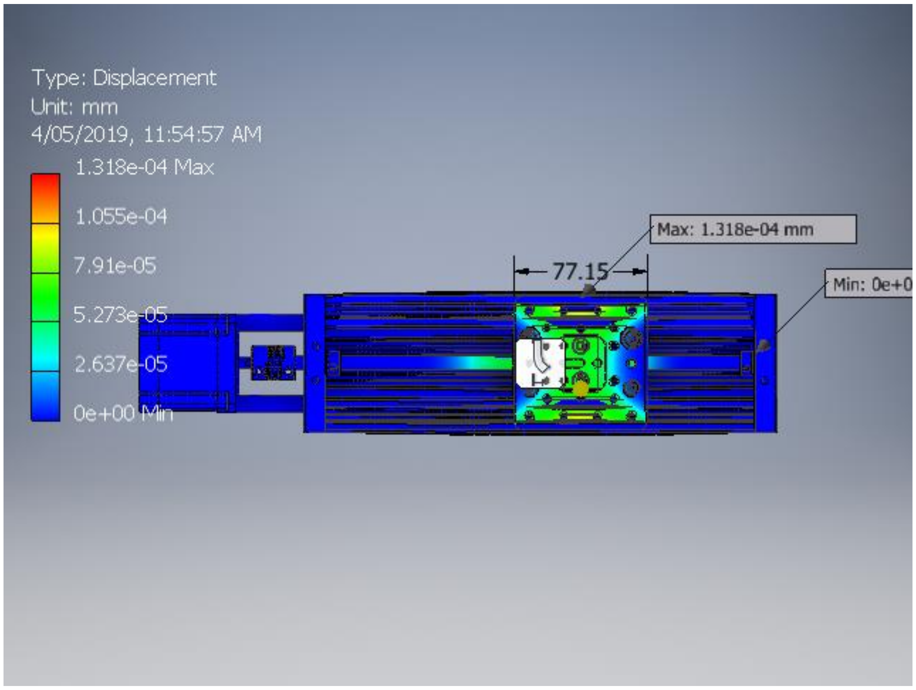


Figure 6: Displacement of C-beam simulation

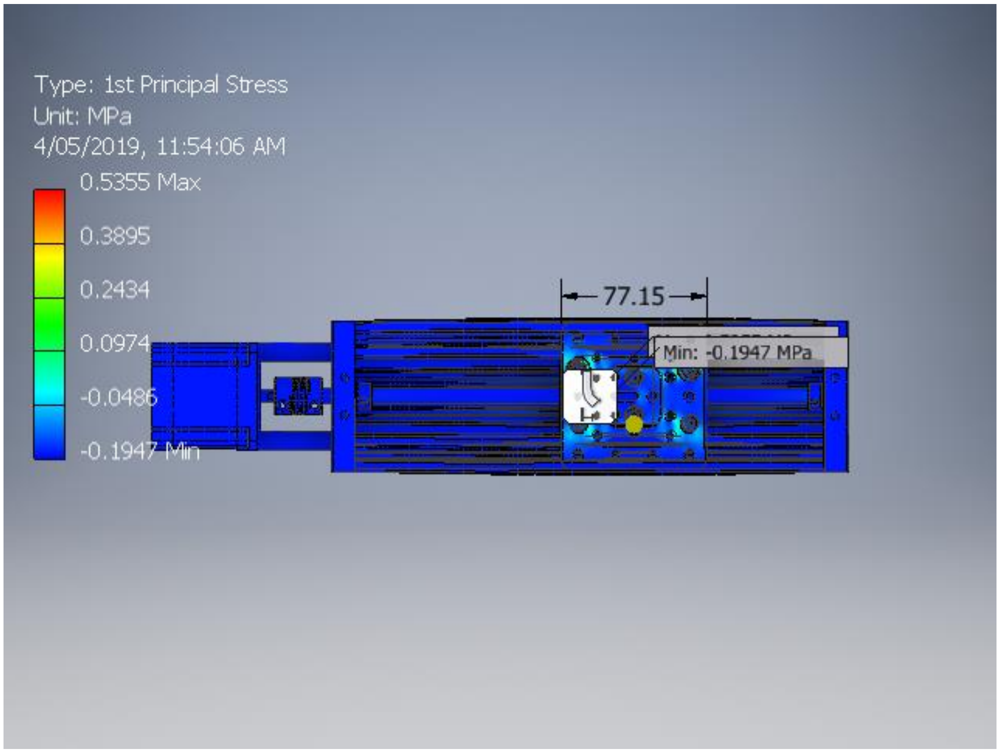


Figure 7: Stress simulation of linear actuation

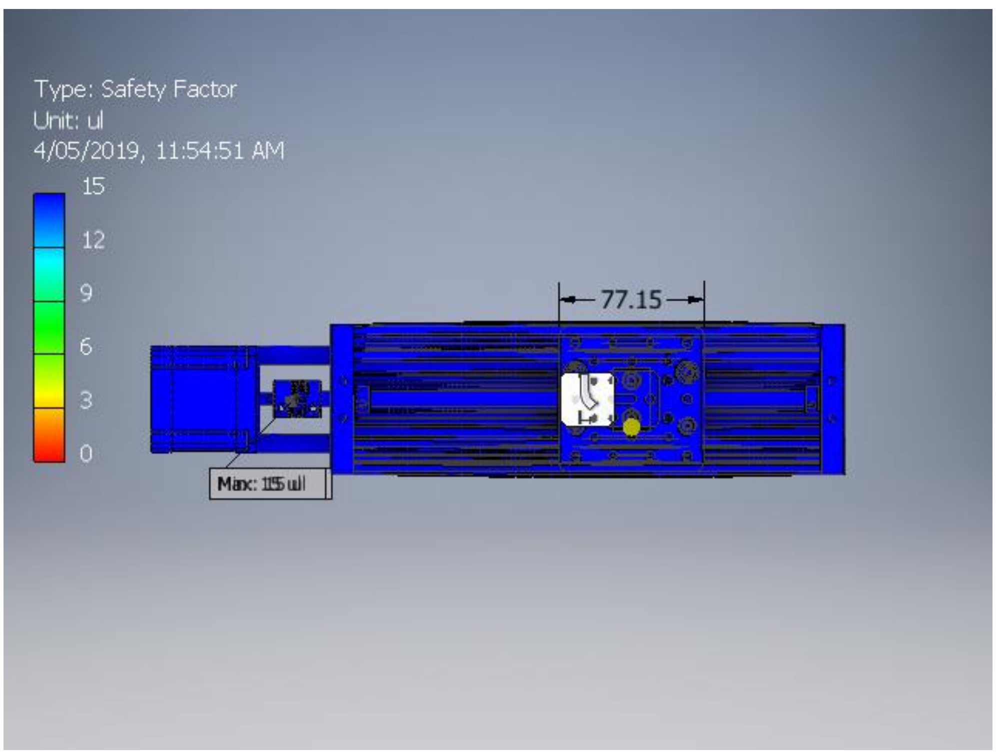


Figure 8: Safety factor of linear actuator

Appendix C: Platform analysis

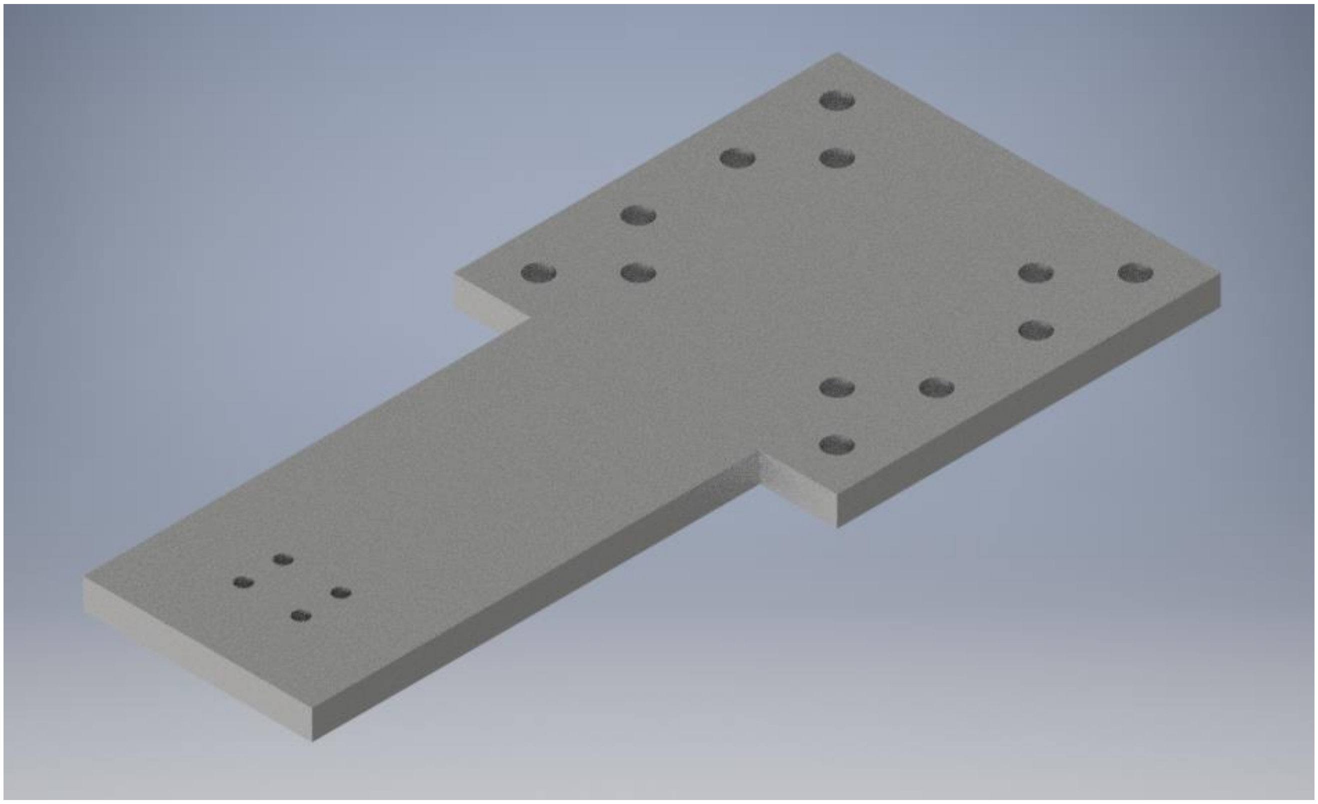


Figure 9.1: Handler machined platform

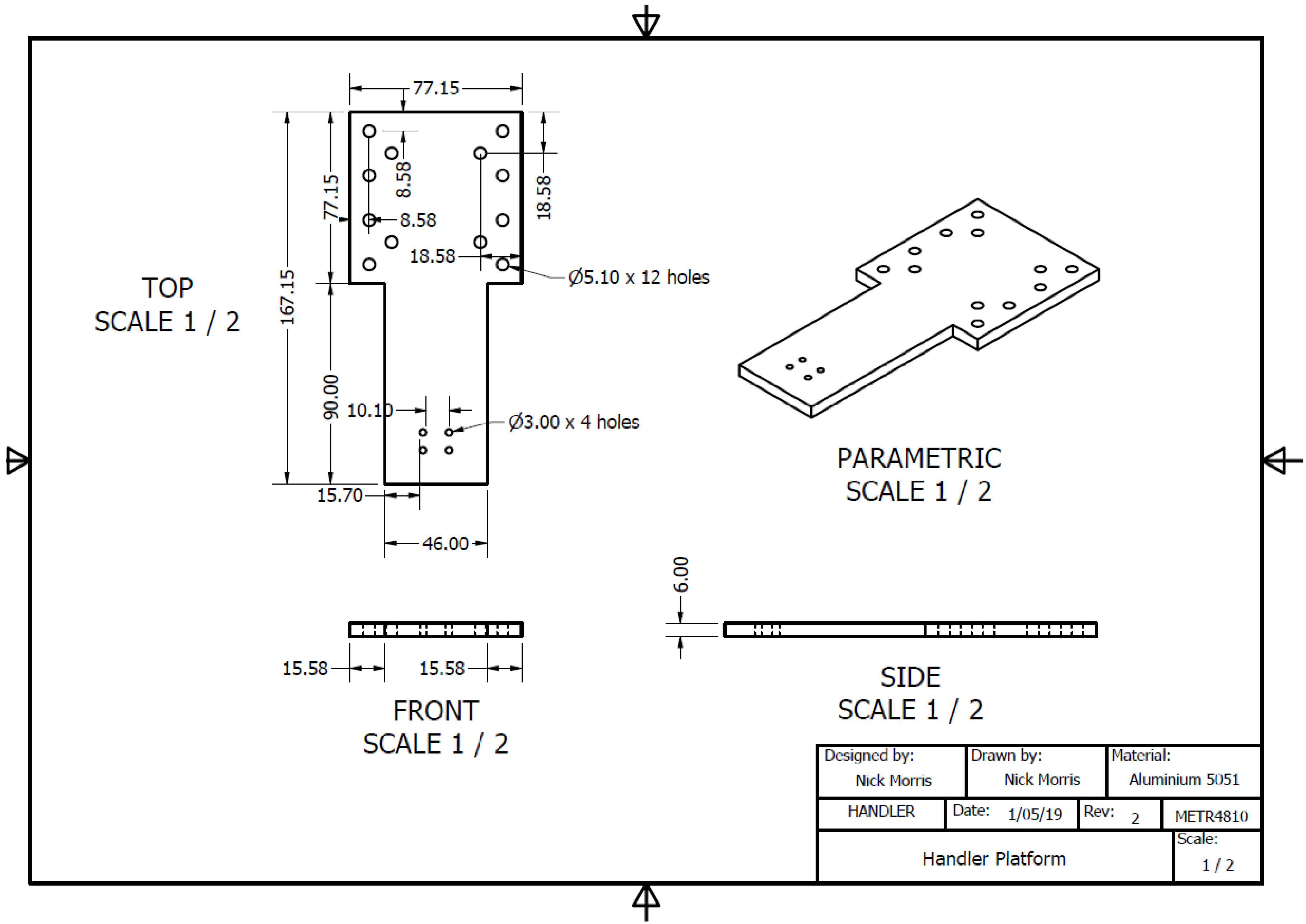


Figure 9.2 – Machine platform drawing

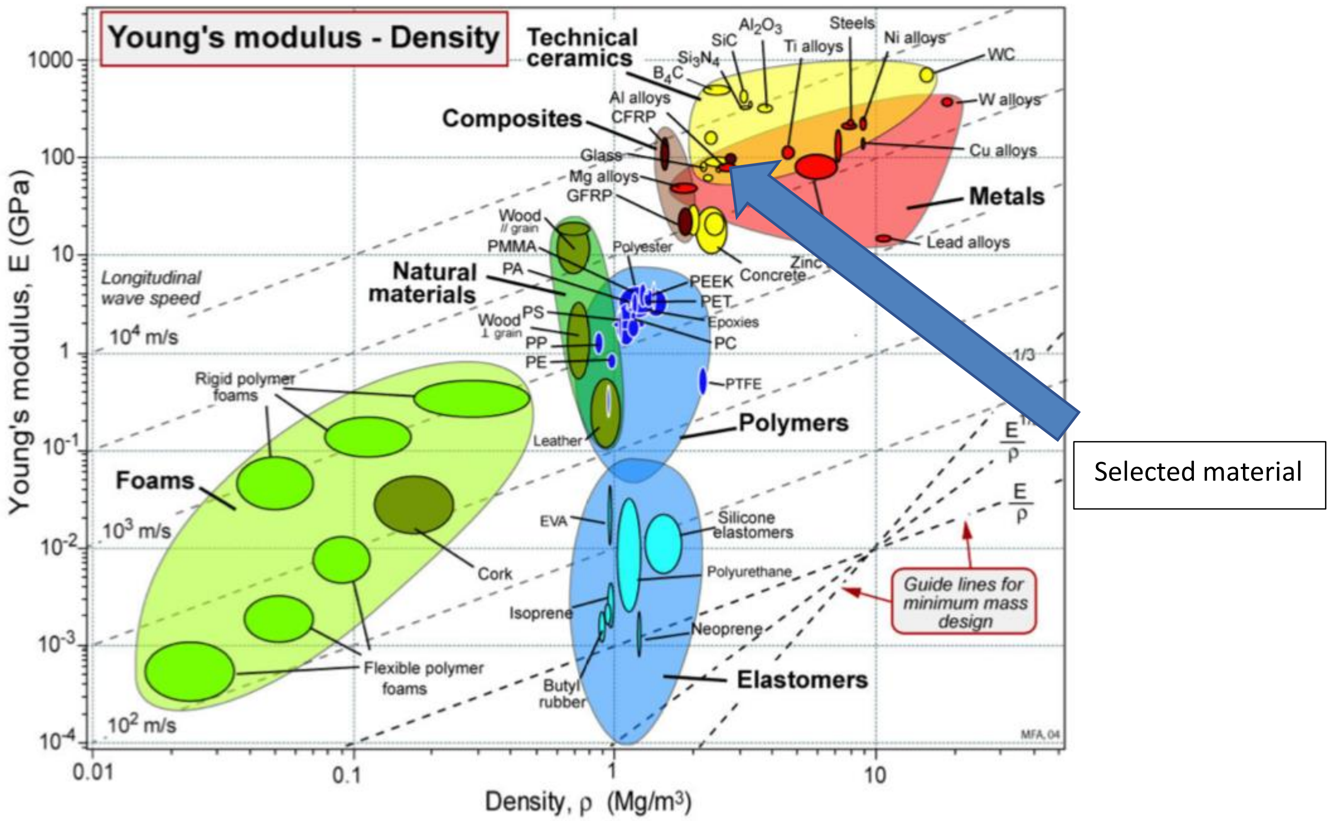


Figure 10: Ashby Diagram [Granta Design: Material selection 2010]

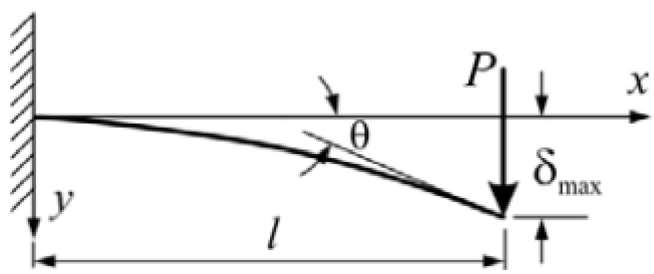


Figure 11: Cantilever deflection

$$\partial = \frac{Pl^3}{3EI}$$

Equation 1: Deflection of cantilever beam equation

Modulus of Aluminium = 68.9GPa

Second moment of area (I) = $\frac{bh^3}{12} = \frac{46 \times 6^3}{12} = 828mm^4$

Approximate force (P) = ma = 140g (motor) + 36g (motor holder) + 27g + 13g (Gripper) + 10g + 200g (misc.)

= 426g * 9.8 = 4.1748N

Length (L) = 90mm

Using equation 1 above:

$$\partial = \frac{4.1748N \ \times 90^3 mm}{3 \times 68.9GPa \times 828mm^4}$$

$$\partial = 0.000017782m$$

$$\partial = 0.017782mm$$

Repeating this calculation with ABS plastic, which has a modulus of 1.4 - 3.1 GPa:

$$\partial = \frac{4.1748N \ \times 90^3 mm}{3 \times 1.4GPa \times 828mm^4} \text{ MAX}$$

$$\partial = \frac{4.1748N \ \times 90^3 mm}{3 \times 3.1GPa \times 828mm^4} \text{ MIN}$$

$$8.752mm\ MAX\ and\ 3.953mm\ MIN$$

This does not meet the accuracy of 0.5mm accuracy.

Appendix D: Handler gripper analysis

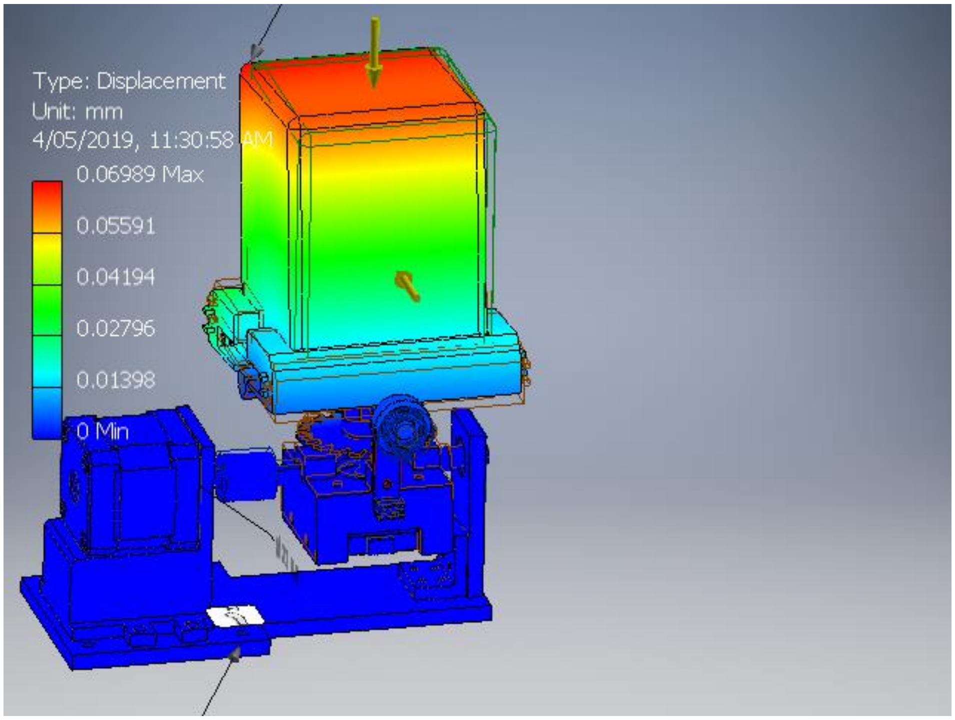


Figure 12: Handler displacement simulation

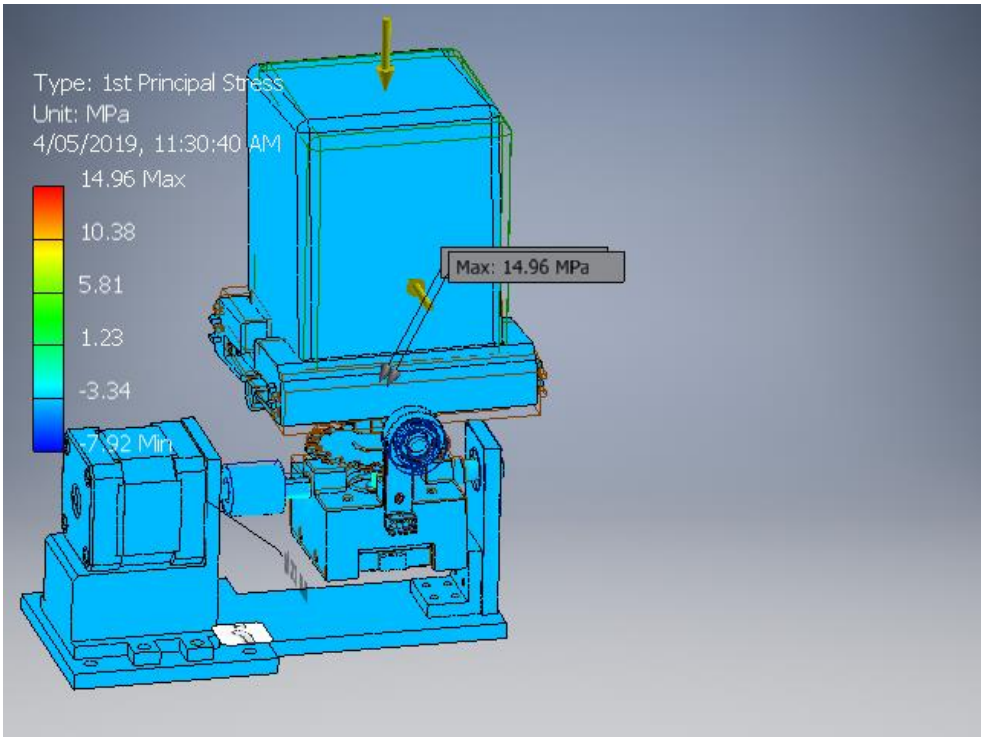


Figure 13: Handler stress simulation

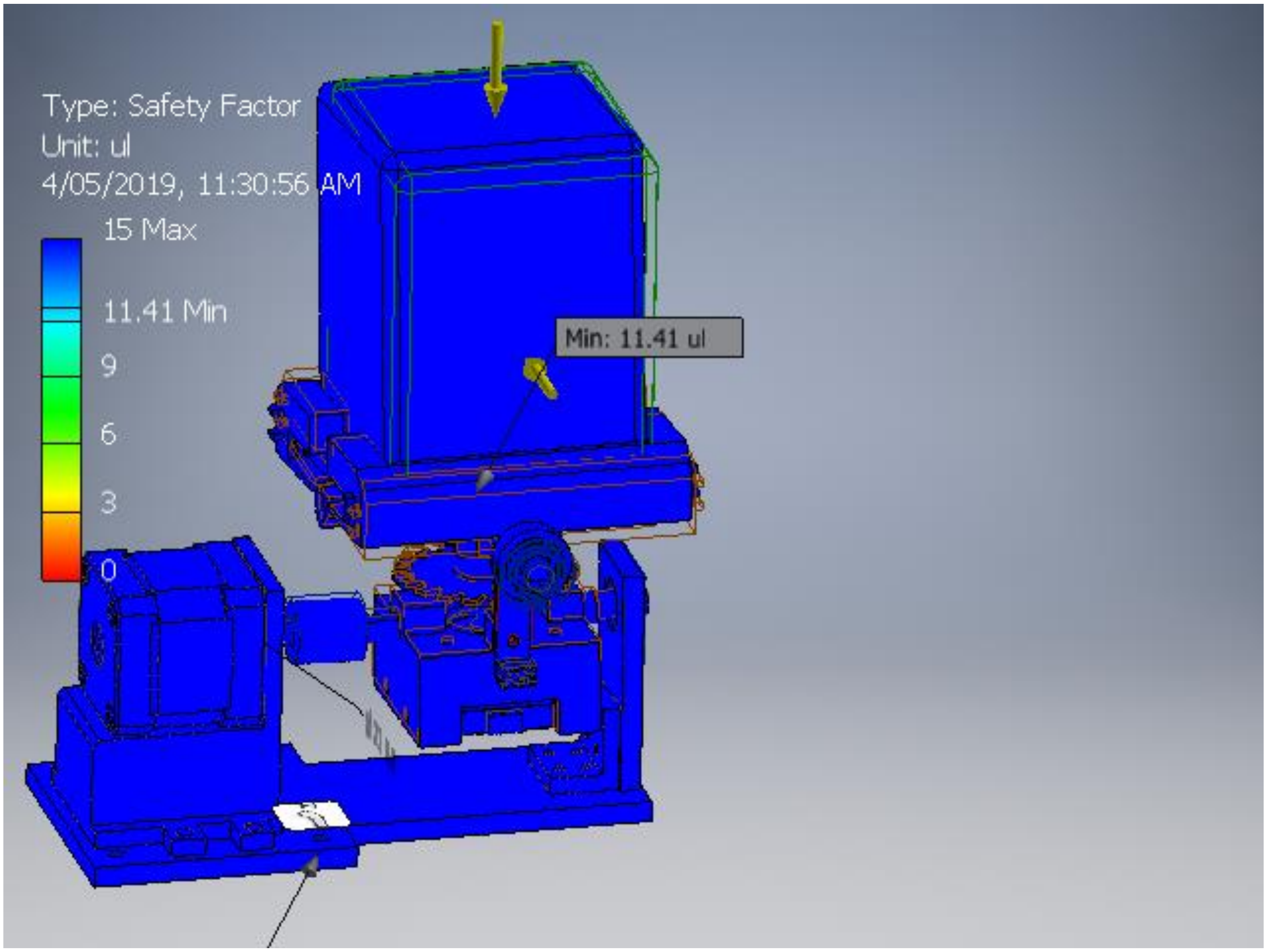


Figure 14: Handler safety factor simulation

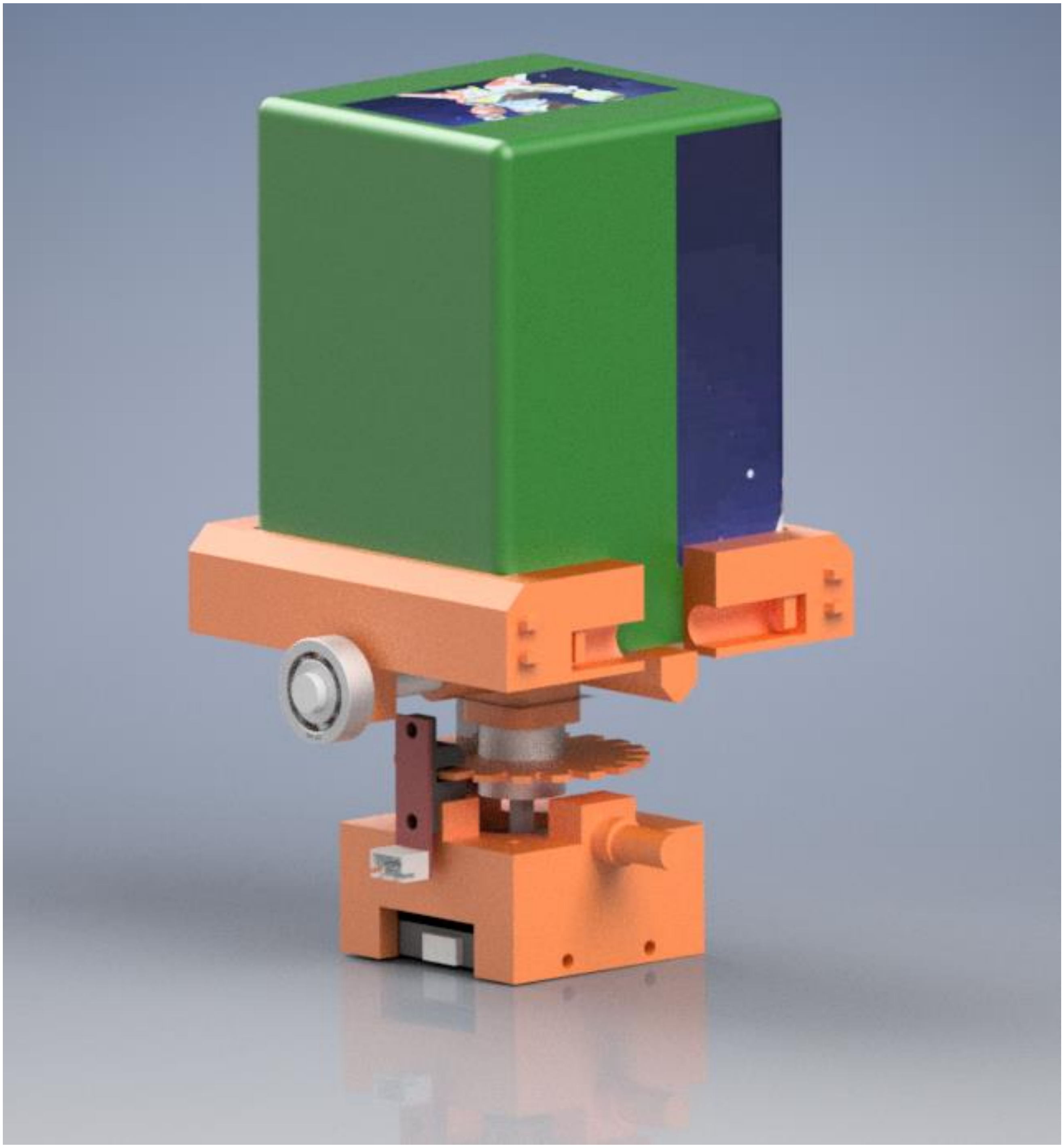


Figure 15.1: Gripper assembly

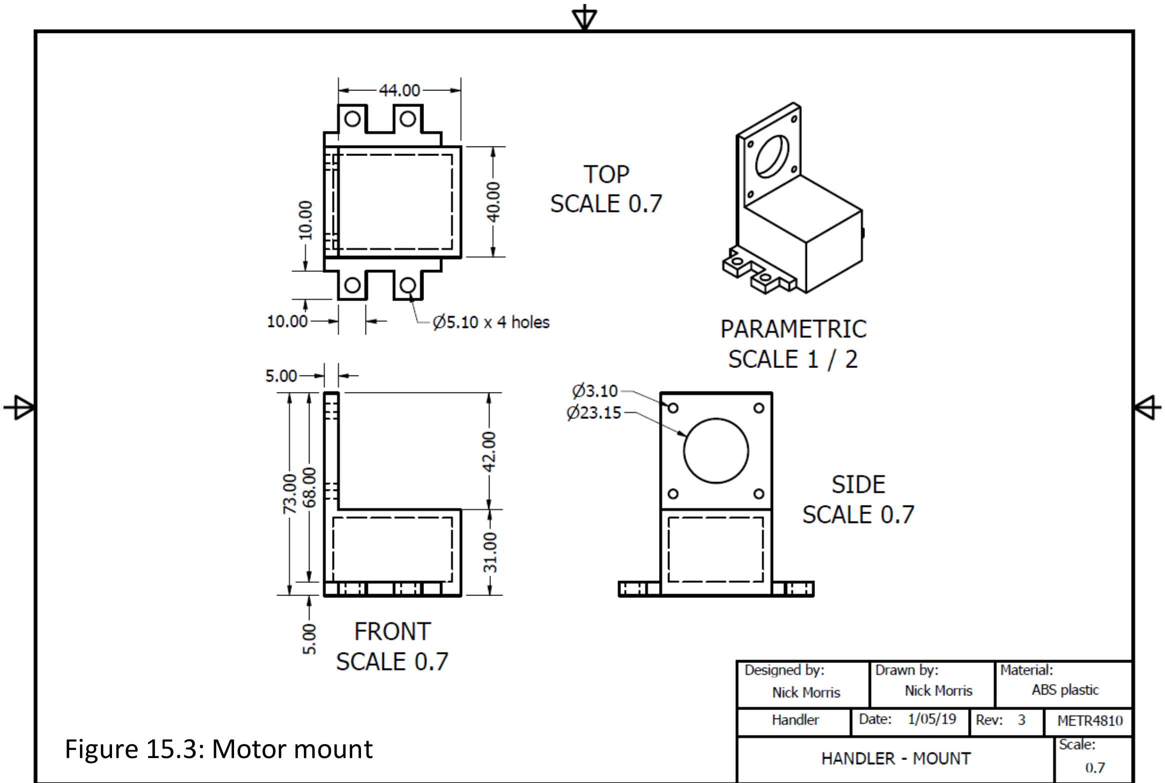


Figure 15.3: Motor mount

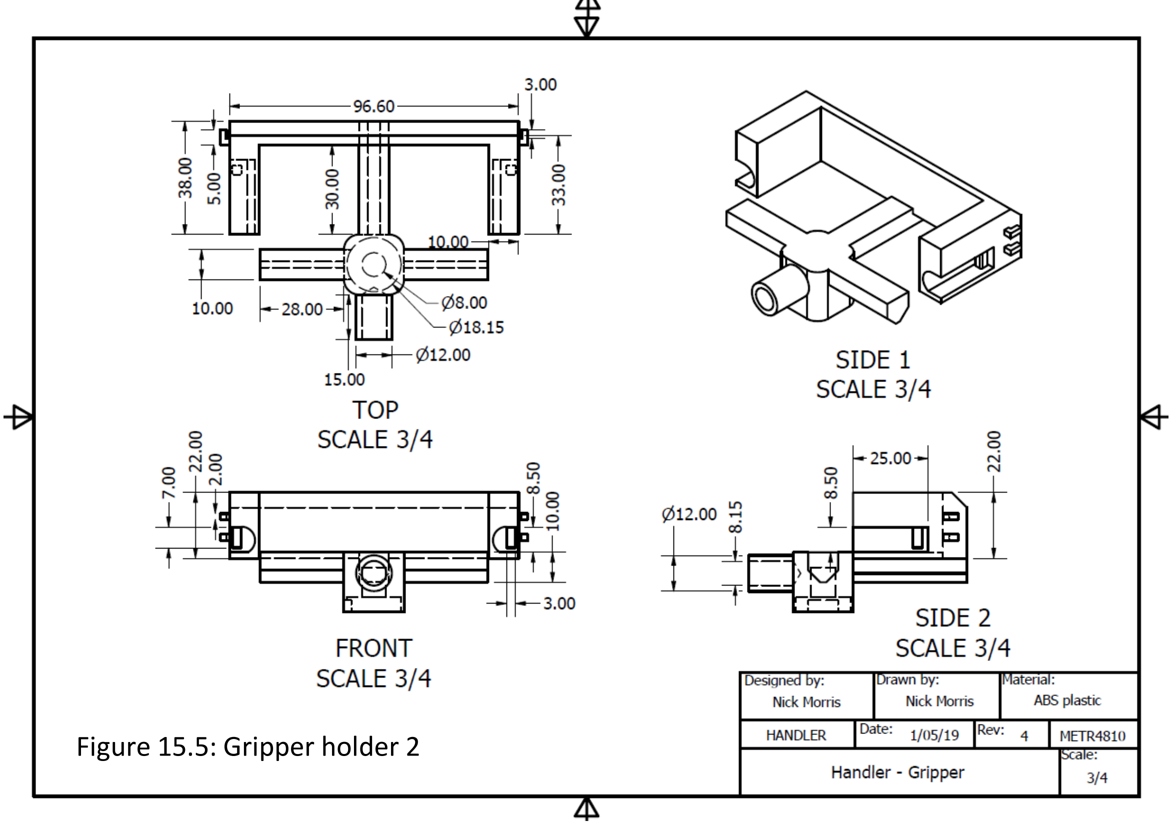


Figure 15.5: Gripper holder 2

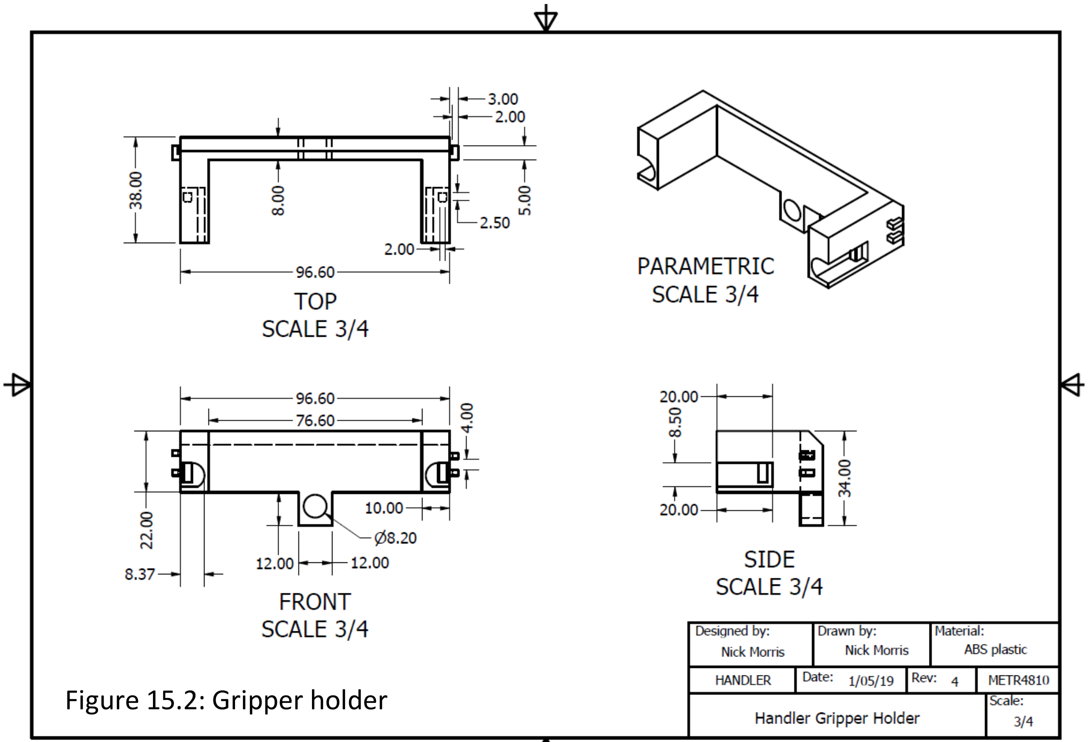


Figure 15.2: Gripper holder

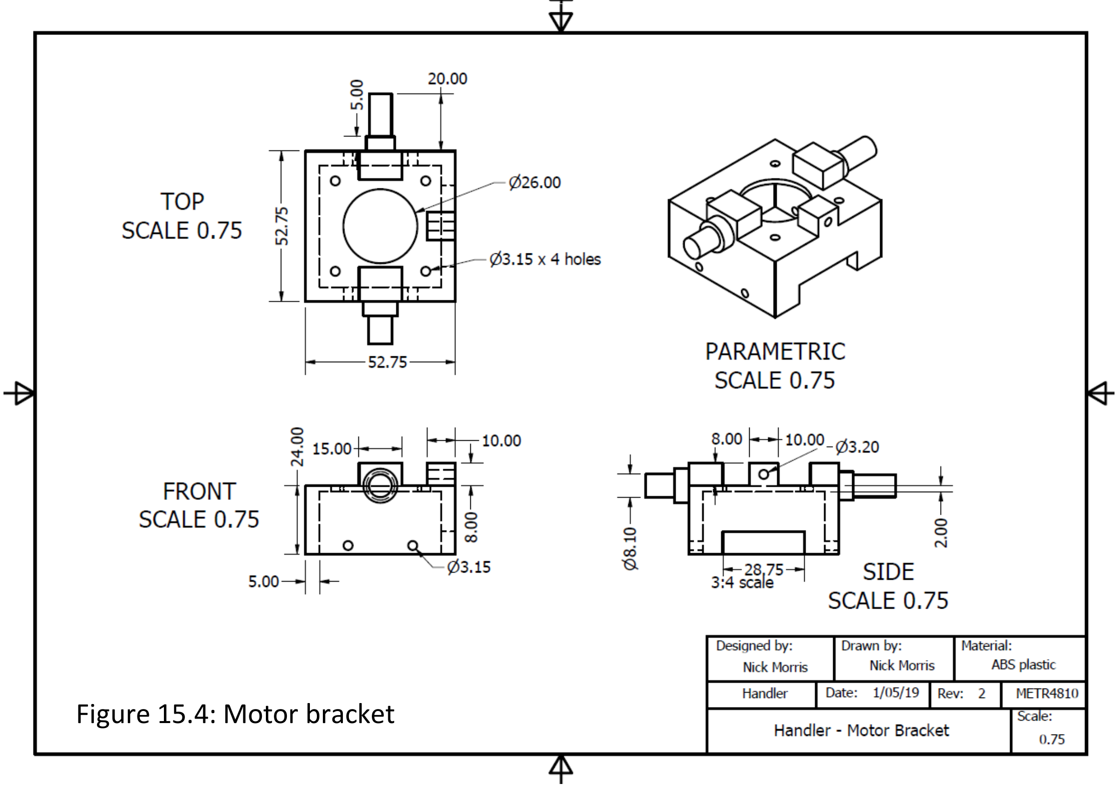


Figure 15.4: Motor bracket

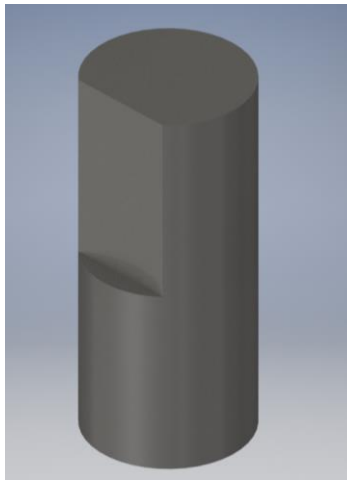


Figure 16: Stainless steel D shafts

Appendix E:

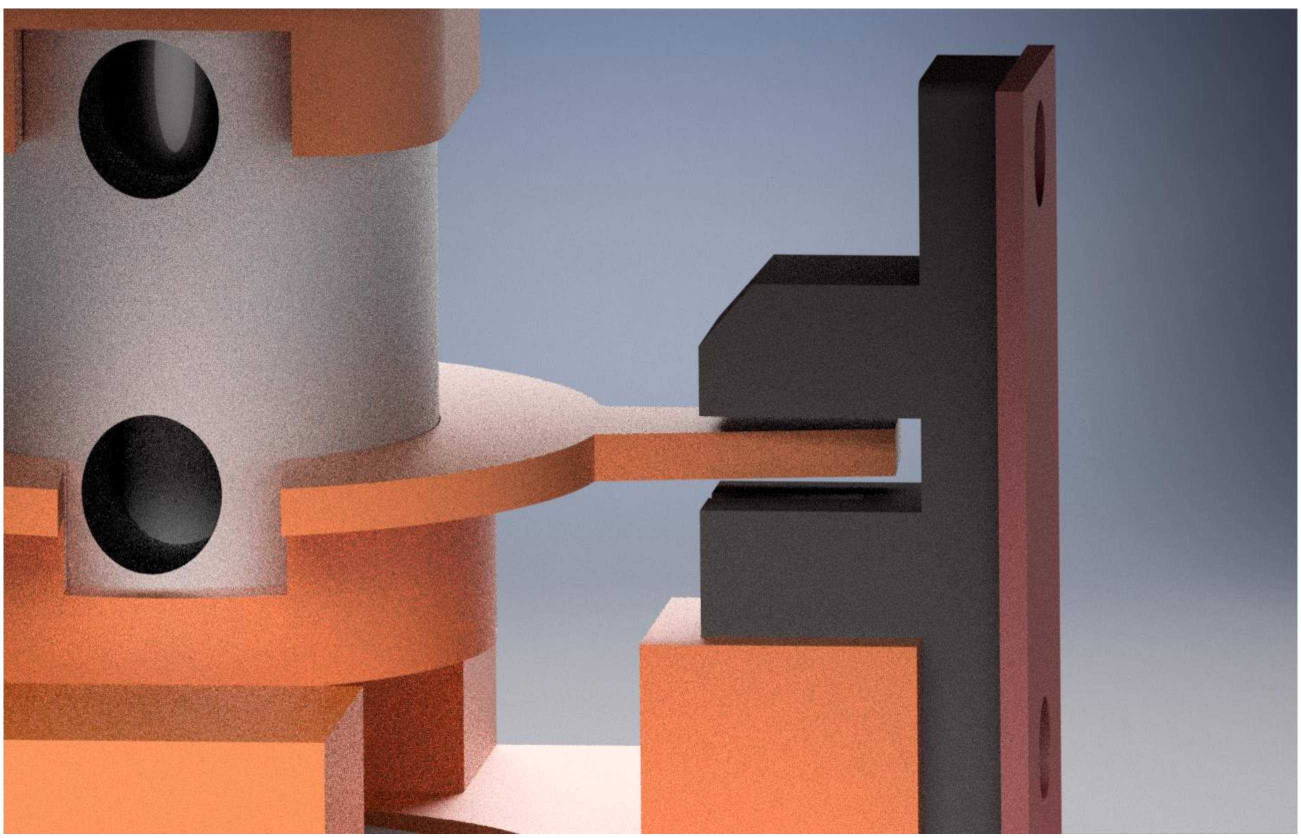


Figure 3: Optical end stop and Opto disk

	PCB assembly method				
Method	Cost	PCB protection	Time to assemble/Ease of use	Size	Total
Velcro hook and fastener	**1**	**3**	**4**	**3**	**11**
3D printed hook and bracket	2	2	2	1	7

Table 1: Hooke and fastener [Get Packed 2019]

Appendix F: Actuator analysis

Appendix F1: Actuator analysis

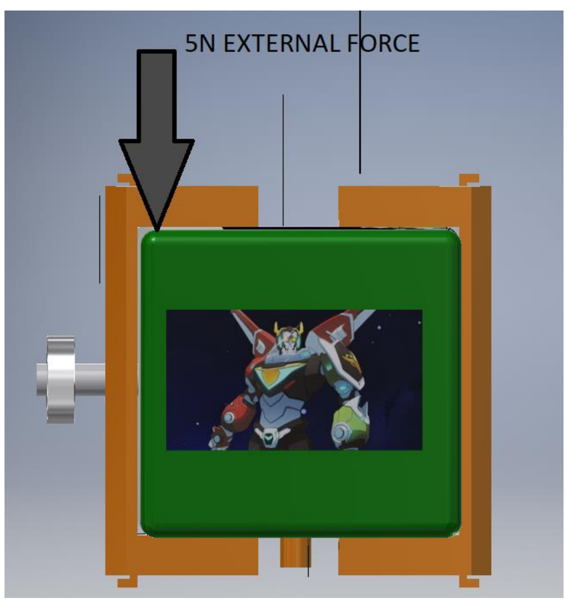


Figure 17: Force acting on foam block from submachine tool

The worst-case scenario is depicted in figure 17 above, where the moment arm is greatest. From the mill and drill, the provided force external force was estimated to be 5N from experimental testing.

$$\tau = Fsin\theta\, r$$

$$\tau = 5N \cdot 1 \cdot\ 0.04m$$

$$\tau = 0.2Nm$$

Furthermore, from experimentation it was estimated the motor needs to spin ~150RPM for the lathe to cut it.

$$\tau = I\alpha$$

$$I = \frac{1}{2}mr^2$$

Mass of the that must be rotated is approximately 100g (Foam block + gripper 3D printed parts and shaft coupler):

The coupler is what is being rotated, which has a radius of 8mm:

$$I = \frac{1}{2} \cdot 0.1kg \cdot 0.008mm^2$$

$$I = 3.2\ \cdot 10^{-6}Kgm^2$$

To find the angular velocity

$$\alpha = \frac{\Delta w}{\Delta t}$$

To go from 0 to 150 RPM, in 0.5 seconds:

$$\alpha = \frac{20\pi}{0.5}$$

$$\alpha = 40\pi$$

$$\tau = 3.2\ \cdot 10^{-6} \cdot 40\pi$$

$$\tau = 0.0004021\text{Nm}$$

$$\tau = 0.04021Ncm$$

Appendix F2: Linear actuator torque calculation

The load on the lead screw is given as approximately 850g (See appendix H), which is 0.85 * 9.81 = 8.3385N. The lead screw moves 20mm/s, which is 0.02m/s.

$$Power = 8.3385N * 0.020 = 0.16677\ watts$$

Metric Leads	Metric Screw Diameter Efficiency Percentages			
	6mm	10mm	12mm	16mm
1 mm	37	35	35	35
2 mm	53	41	41	41
4 mm	62	59	54	47
5 mm	64	64	59	52
6 mm	66	67	63	63
8 mm	68	73	70	70
10 mm	72	76	73	73
12 mm	76	78	75	75
16 mm		79	78	78
25 mm	81	83	82	82

Figure 18: lead screw efficiency [PBC Linear]

From figure 18, using an 8mm diameter lead screw, with 2mm pitch falls between 53% and 41% efficiency. Thus, taking the worst-case scenario, the efficiency can be assumed to be approximately 41%.

$$Power = 0.40675\ watts$$

For a rotation of 150 RPM, there must be 2.5 rev/second or 2.5 x 2pi

$$Torque = \frac{0.407}{7.854} = 0.0518Nm$$

$$Torque = 5.18Ncm$$

$$RPM = 150$$

$$60\left[\frac{PPS}{200}\right] = 150$$

$$PPS = 500$$

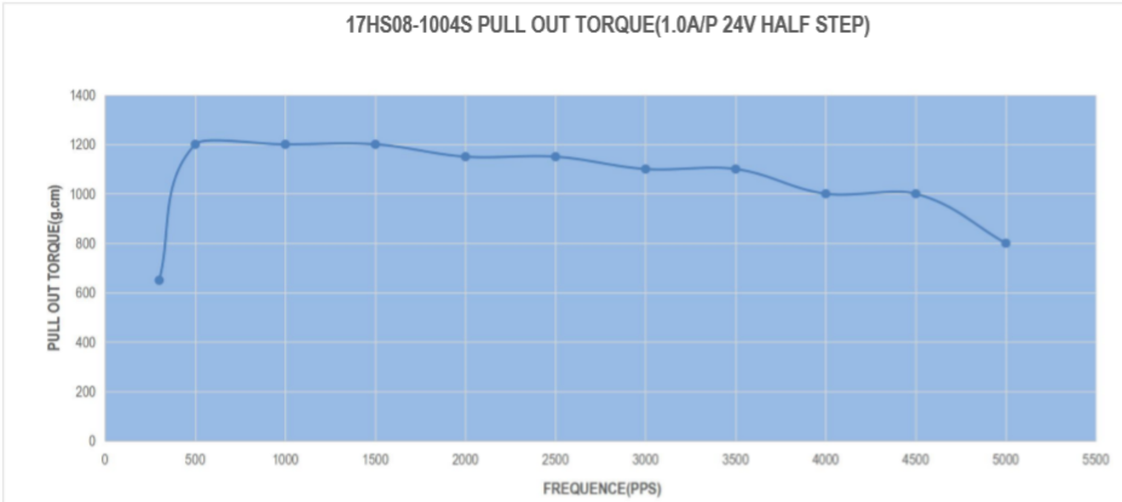


Figure 19: Torque curve for 17HS08-1004S NEMA 17 20mm body stepper motor [12]

From the torque speed curve in figure 19 for a NEMA 17 x 20mm body at 24V and half stepping, the torque for 500 PPS is approximately 1200gcm. During operation, the stepper motors will be run at full step and 12V, thus the steppers will be able to produce 1200gcm at 500 PPS. This is the equivalent of 11.76798Ncm.

Safety factor:

$$SF = \frac{11.768}{5.18}$$

$$SF > 2\ (2.27)$$

Appendix F3 - Y axis rotation torque requirement:

This motor will be responsible for holding and rotating 90 degrees of the foam block.

Mass = mass of motor 1 + mass part

$$Mass = 0.125Kg\ (motor) + 0.100Kg\ (Gripper)\ +\ 0.036Kg\ (Motor\ holder) \\ +\ 0.011Kg\ (Coupler)$$

$$Mass = 0.272Kg$$

Treating the problem as a fixed beam where the torque will be Ma on the left:

Figure 20: Force from handler's gripper

The centre of gravity acts in the middle of the foam. Adding a point mass with the following force:

$$F = ma$$

$$F = 0.272 \cdot 9.81$$

$$F = 2.66832\ N$$

To find the torque:

$$\tau = Fr$$

$$\tau = 2.66832 \cdot 0.06$$

$$\tau = 0.16\ Nm\ (holding\ torque)$$

From steppers online, a NEMA 17 stepper motor with a 45Ncm torque rating has been selected. The chosen stepper motor meets specifications and has a safety factor of 2.8125.

Safety factor:

$$SF = \frac{0.45}{0.16}$$

$$SF = 2.8125$$

The motor must also be able to rotate the foam block 90 degrees:

$$\tau = I\alpha$$

$$I = \frac{1}{2}mr^2$$

$$I = \frac{1}{2}r^2$$

$$I = \frac{1}{2} \cdot 0.272 \cdot 0.060^2$$

$$I = 4.896 \cdot 10^{-4}$$

Angular velocity

$$\alpha = \frac{\Delta w}{\Delta t}$$

Let it take 0.5s to speed up and stop from 60 rpm

$$\alpha = \frac{2\pi \cdot 60}{0.5 \cdot 60}$$

$$\alpha = \pi \, rad/s^2$$

To find the torque required:

$$\tau = I\alpha$$

$$\tau = \pi \cdot 4.896 \cdot 10^{-4}$$

$$\tau = 0.001538Nm$$

$$\tau = 0.1538Ncm$$

Appendix G: Power calculations

Z axis stepper motor power requirements

- Current rating: 0.4A per coil
- Resistance: 30 Ω per coil

$$P = I^2R$$
$$P = 0.4^2 \cdot 30$$
$$P = 4.8w$$

Linear actuator stepper motor requirements

- Current rating: 1A per coil
- Resistance: 3.5 Ω per coil

$$P = I^2R$$

$$P = 1^2 \cdot 3.5$$

$$P = 3.5w$$

Y axis stepper motor power requirements:

Current rating: 1.5A per coil

Resistance: 2.3 Ω per coil

$$P = I^2 R$$

$$P = 1.5^2 \cdot 2.3$$

$$P = 5.175w$$

APPENDIX I: Toolpath and g-code generation

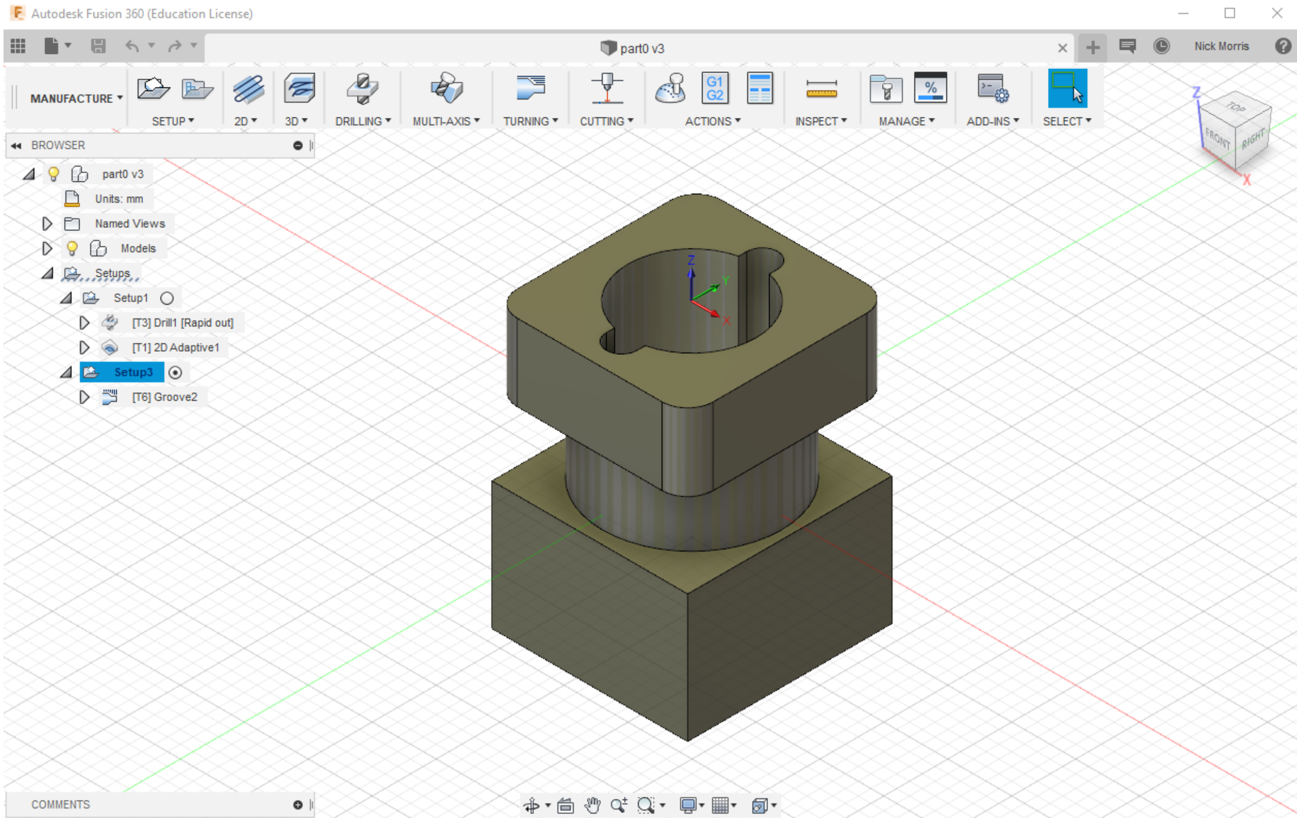


Figure 21: Fusion 360 part tool path set up

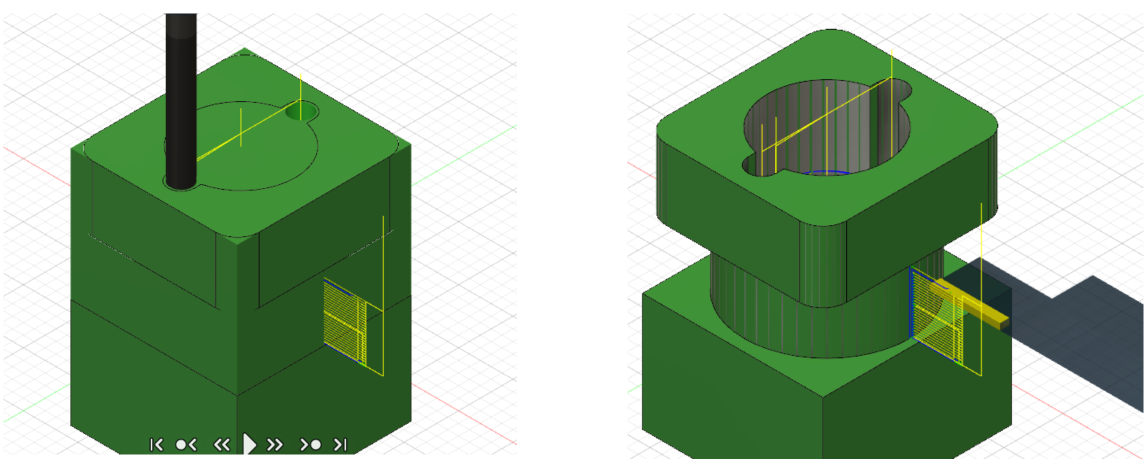
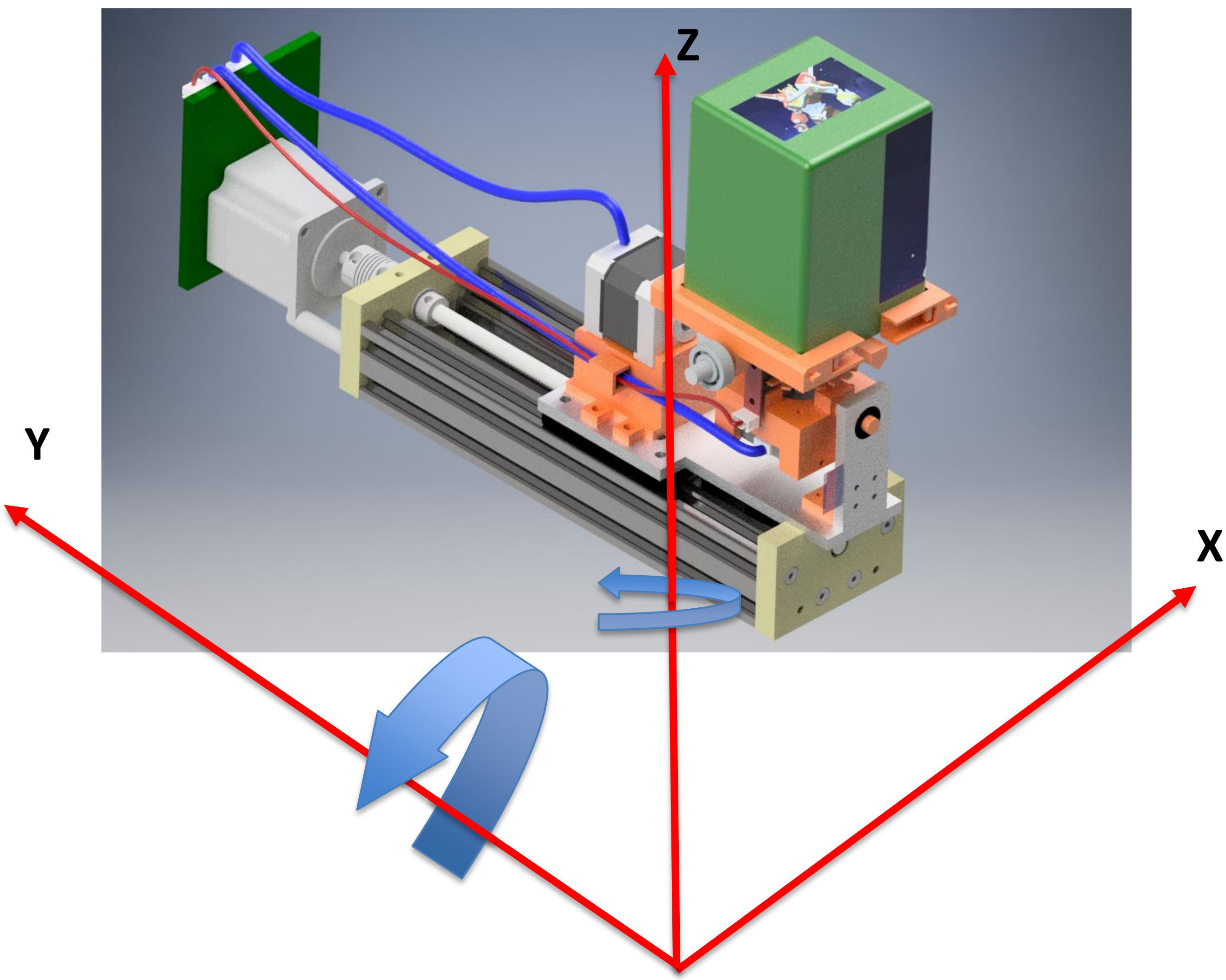


Figure 22: Fusion 360 tool path simulation. Drilling (left) and lathing. (right)

APPENDIX K: Handler coordinate system



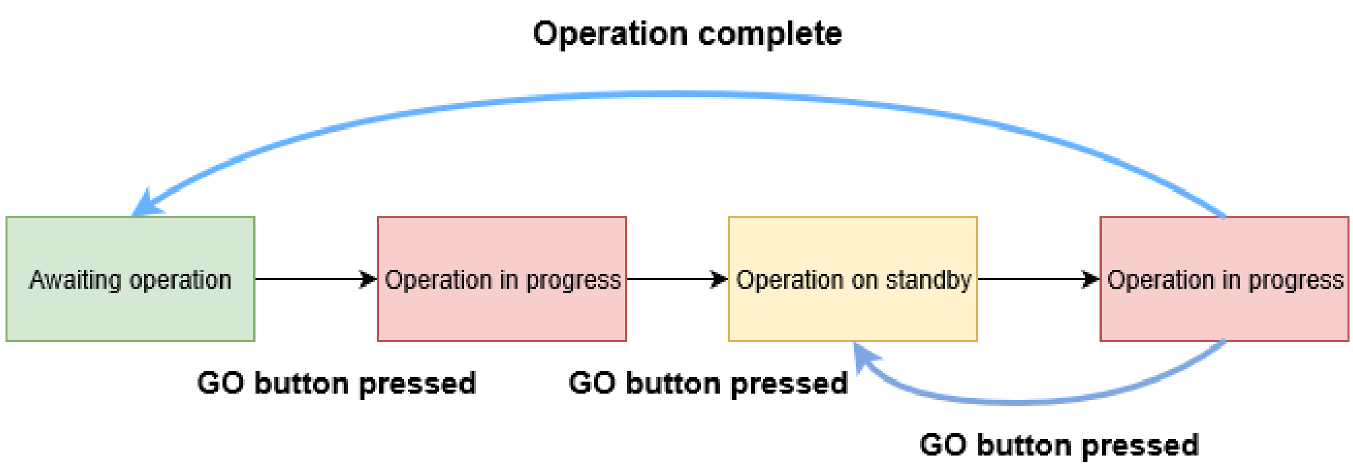
Appendix K1: Handler coordinate system

APPENDIX L: Shaft material decision matrix

	Shaft material					
Method	Cost	Strength	Availability	Mass	Reliability /Durability	Total
Stainless steel	**1**	**10**	**10**	**3**	**10**	**34**
Mild steel	2	8	10	3	8	31
Aluminium	3	5	5	5	10	28
ABS plastic	8	3	10	8	3	32

Appendix L1: Shaft material decision matrix

APPENDIX M: MACHINETRON Status state diagram



Appendix M1: MACHINETRON status state diagram

APPENDIX N: Communication protocol

	Communication protocol			
Method	Complexity	Speed	Distance (for application short distance)	Total
UART	**6**	**3**	**10**	**19**
SPI	4	10	1 (Not applicable)	15
I2C	3	7	7	17

Appendix M1: Communication protocol decision matrix

Appendix J: Software/code

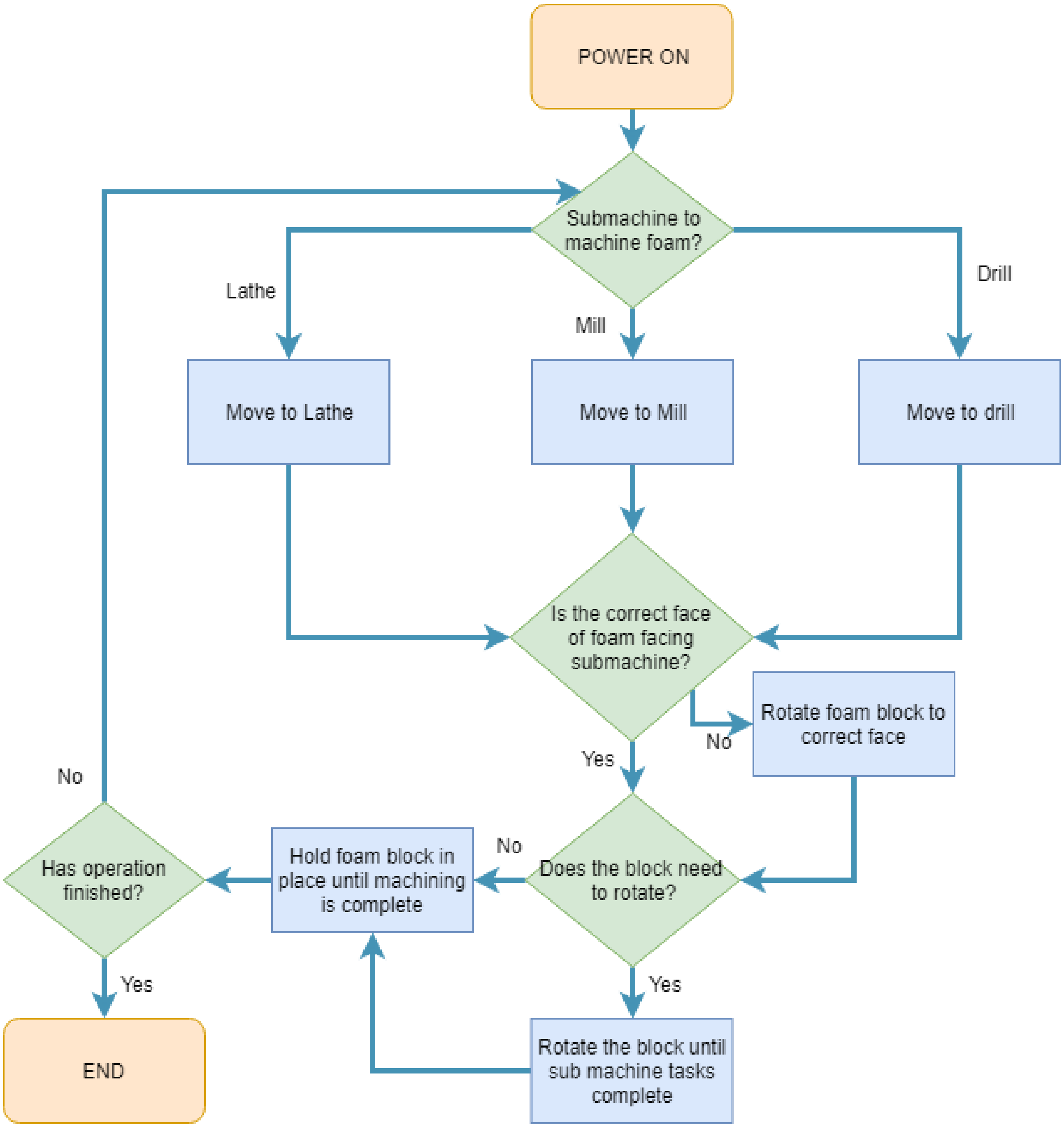


Figure 24: Handler software flowchart

Linear actuation function

```
/**********************************************************************
 * LinearActuation.c                                                  *
 * Created: 4/27/2019 3:56:25 PM                                      *
 * Author : Nick Morris                                               *
 * s44373388 - Team 8 - METR4810                                      *
 * Linear Actuation given a desired and current position              *
 **********************************************************************/
#include "main.h"
#include "math.h"
#include "stdlib.h"
#include "linearactuation.h"
/*
#define MILL 0
#define DRILL 10
#define LATHE 20
#define LENGTH 20
*/
int current_position = 0;
/**************************************************
* void test_function(void)                        *
* Test function to test actuator functionality    *
* Rotate the steppers back and forth 4 rotations *
**************************************************/
void test_function(void){
 HAL_GPIO_WritePin(GPIOA, DIR_Pin, GPIO_PIN_SET);
 for(int i = 0; i < 800; i++){
  HAL_GPIO_WritePin(GPIOA, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOA, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
 }
 HAL_Delay(1000);
 HAL_GPIO_WritePin(GPIOA, DIR_Pin, GPIO_PIN_RESET);

 for(int i = 0; i < 800; i++){
  HAL_GPIO_WritePin(GPIOA, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOA, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
 }
 HAL_Delay(1000);
}
/************************************************************************
*************************************************************************
** move_to(float, float)                                               **
```

```
** Inputs: two floats - current_position and desired_position in mm    **
** Outputs: Linear actuation to desired_position                             **
** Returns desired position - new position after operation              **
** E.g current_position = move_to(current_position, LATHE);            **
** NOTE: WILL MOVE TO CLOSEST POSITION (STEP) WITHIN 0.04mm             **
************************************************************************
************************************************************************/
int move_to(float current_position, float desired_position){
 //Each step does 0.04mm
 int steps_taken = 0;
 //Multiply by 25 to convert steps to mm
 int steps_total = (desired_position - current_position)*25;
 //If the handler must move backwards, set rotation to anticlockwise

 if(steps_total < 0){
  HAL_GPIO_WritePin(GPIOA, DIR_Pin, GPIO_PIN_SET);
 }
 //Move forward, set direction to clockwise
 else{
  HAL_GPIO_WritePin(GPIOA, DIR_Pin, GPIO_PIN_RESET);
 }


 //Move forward until destination reached
 //printf("Moving along the horizontal axis captain");
 for(steps_taken = 0; steps_taken<abs(steps_total); steps_taken++){
  HAL_GPIO_WritePin(GPIOA, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOA, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
 }
 HAL_Delay(1000);
 //printf("Horizontal actuation complete captain!");
 return desired_position;
}
```

```
/*****************************************************************
 * rotate.c                                                       *
 * Created: 4/27/2019 3:56:25 PM                                  *
 * Author : Nick Morris                                           *
 * s44373388 - Team 8 - METR4810                                  *
 * rotate to desired face and spin the z axis                     *
 *****************************************************************/
```

Face rotation example

```
//#include <avr/io.h> For prototyping in Atmelstudio
#include "main.h"
#include "rotate.h"
#define FACE_ONE 1
#define FACE_TWO 2
#define FACE_THREE 3
#define FACE_FOUR 4
#define FACE_FIVE 5

/*
int current_face = FACE_FOUR;
int current_z_face = FACE_FOUR;
*/
/************************************************************************
*************************************************************************
** rotate_face_one(int*, int*)                                         **
** Inputs: two int pointers - current face, current_z_face             **
** Outputs: rotate to face one                                         **
** Updates value of current_face and current_z_face                    **
** E.g rotate_face_one(&current_face, &current_z_face);                **
*************************************************************************
************************************************************************/
void rotate_face_one(int *current_face, int *current_z_face){
 switch(*current_face){
  //Face 1
  case FACE_ONE:
  //printf("Face 1 to Face 1...");
  *current_face = FACE_ONE;
  *current_z_face = FACE_ONE;
  //printf("Operation complete!");

  break;

  //Face 2
  case FACE_TWO:
   //printf("Face 2 to Face 1");
   //Set rotation to clockwise
   HAL_GPIO_WritePin(DIR_GPIO_Port, DIR_Pin, GPIO_PIN_SET);
   //Rotation in Z direction clockwise 90 degrees (50 steps)
   for(int i = 0; i < 50; i++){
    HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
    HAL_Delay(5);
    HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
    HAL_Delay(5);
   }
```

```
 *current_face = FACE_ONE;
 *current_z_face = FACE_ONE;
 //printf("Operation complete!");
 break;

//Face 3
case FACE_THREE:
 //printf("Face 3 to Face 1");
 //Set rotation to anti-clockwise (Does not matter here)
 HAL_GPIO_WritePin(DIR_GPIO_Port, DIR_Pin, GPIO_PIN_RESET);
 //Rotation in Z direction anti-clockwise 180 degrees (100 steps)
 for(int i = 0; i < 100; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(5);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(5);
 }
 *current_face = FACE_ONE;
 *current_z_face = FACE_ONE;
 //printf("Operation complete!");
 break;

//Face 4
case FACE_FOUR:
//printf("Face 4 to Face 1");
 //Set rotation to anti-clockwise
 HAL_GPIO_WritePin(DIR_GPIO_Port, DIR_Pin, GPIO_PIN_RESET);
 //Rotation in Z direction anti-clockwise 90 degrees (50 steps)
 for(int i = 0; i < 50; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(5);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(5);
 }
 *current_face = FACE_ONE;
 *current_z_face = FACE_ONE;
 //printf("Operation complete!");
 break;

//Face 5
case FACE_FIVE:
 //printf("Face 5 to Face 1");
 //Set rotation to anti-clockwise
 HAL_GPIO_WritePin(DIR2_GPIO_Port, DIR2_Pin, GPIO_PIN_RESET);
 //Rotation towards submachines of 90 degrees
 for(int i = 0; i < 50; i++){
  HAL_GPIO_WritePin(STEP2_GPIO_Port, STEP2_Pin, GPIO_PIN_SET);
```

```
    HAL_Delay(5);
    HAL_GPIO_WritePin(STEP2_GPIO_Port, STEP2_Pin, GPIO_PIN_RESET);
    HAL_Delay(5);
   }
 //recall function for z axis alignment
   rotate_face_one(current_z_face, current_face);

   //printf("Operation complete!");
   break;
  //Something has gone wrong here
  default:
   //printf("Error 11: You tried to rotate to face one but something went wrong.");
  break;
 }

}
```

Lathe Function

```
/***********************************************************************
************************************************************************
** void lathe_time(void)                                               **
** Inputs: void                                                        **
** Outputs: rotates z axis 150rpm                                      **
** E.g lathe_time()                                                    **
************************************************************************
***********************************************************************/
void lathe_time(void){
 ////printf("It's lathe time!");

 //Set rotation to clockwise
 HAL_GPIO_WritePin(DIR_GPIO_Port, DIR_Pin, GPIO_PIN_SET);
 //Rotate 150 rpm for 10 rotations
 //Speed up function
 for(int i = 0; i < 50; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(5);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(5);
 }
 for(int i = 0; i < 50; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(4);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(4);
 }
 for(int i = 0; i < 50; i++){
```

```
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(3);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(3);
 }
 for(int i = 0; i < 50; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(2);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(2);
 }
 //Rotate at 150rpm now.
 for(int i = 0; i < 1600; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
 }
 for(int i = 0; i < 200; i++){
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_SET);
  HAL_Delay(3);
  HAL_GPIO_WritePin(STEP_GPIO_Port, STEP_Pin, GPIO_PIN_RESET);
  HAL_Delay(3);
 }
 ////printf("Back to work... lathe time over...");
}
```

Timer handling code

```
//48MHZ timer. LD2 is a debugging LED, LEDG is the green LED, LEDA is the amber LED
and LED2 is the red LED.
void TIM14_IRQHandler(void)
{

  HAL_TIM_IRQHandler(&htim14);

 if(HAL_GPIO_ReadPin(ENDSTOP_GPIO_Port, ENDSTOP_Pin) == 0)
 {
  HAL_GPIO_WritePin(LEDG_GPIO_Port, LEDG_Pin, GPIO_PIN_RESET);
 }
 else
 {
  HAL_GPIO_WritePin(LEDG_GPIO_Port, LEDG_Pin, GPIO_PIN_SET);
 }
 //If button is pressed
  if(HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin) == 0)// ||
```

```
HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin) == 0)
 {

  if(buttonPressed == 0)
  {
   //Once confidence has passed the confidence level
   if(buttonPressedConfidenceLevel > confidenceThreshold)
   {
    //Toggle LEDS
    if(LEDState == 0)
    {
     LEDState = 1;

     HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
     HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
     //Pause entire program by entering while loop until button pressed again
     while(1)
     {
      HAL_GPIO_WritePin(LEDA_GPIO_Port, LEDA_Pin, GPIO_PIN_SET);
      //Wait for long press to avoid accidental unpause
      if(buttonPressedConfidenceLevelTwo > confidenceThreshold+50000)//50000
      {
       buttonPressedConfidenceLevelTwo = 0;
       HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
       HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
       HAL_GPIO_WritePin(LEDA_GPIO_Port, LEDA_Pin, GPIO_PIN_RESET);

       LEDState = 0;
       break;
      }
      //Read for button press
      else if(HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin) == 0)// ||
HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin) == 0)
      {
       buttonPressedConfidenceLevelTwo++;

      }
      else
      {
       buttonPressedConfidenceLevelTwo = 0;
      }

     }
    }
    else
    {
     LEDState = 0;
```

```
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
      HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
     }
     //Update button pressed to ON
     buttonPressed = 1;
    }
    else
    {
     buttonPressedConfidenceLevel++;
     buttonReleasedConfidenceLevel = 0;
    }
   }
  }
  else
  {
   if(buttonPressed == 1)
   {

    //Once release condience has been reached
    if(buttonReleasedConfidenceLevel > confidenceThreshold)
     {
      //Set button pressed to OFF
     buttonPressed = 0;
    }
     else
    {
     //Increase buttoon released confidence level
    buttonReleasedConfidenceLevel++;
    buttonPressedConfidenceLevel = 0;
    }
   }
  }




 }
```

Appendix O: Budget

Handler total: $85.62

Mechanical Parts					
Stepper motor NEMA 17 42x42x34mm			6.02	1	6.02
Stepper motor NEMA 17 42x42x22mm			5.88	1	5.88
Stepper motor NEMA 17 42x42x39mm			9.11	1	9.11
Handler machined platform			1.75	1	1.75
Handler machined support			1.75	1	1.75
C-Beam			6	1	6
C-Beam Gantry Plate			4.16	1	4.16
Compression springs			0.03635	2	0.0727
400mm T8 lead screw			4.02	1	4.02
Xtreme Mini V Wheel Kit			1.63	4	6.52
				Frame Total:	**39.2627**

PCB/Electrical

Item	Part Number	manufacturer	Unit Price au	Units Used	Total
Switch Operation:Momentary	26-725	MCM	1.44	1	1.44
Wire-To-Board Connector, Right Angle, 2.5 mm, 6 C	S6B-XH-A	JST	0.172	3	0.516
Connector Housing, XH Series, Receptacle, 6 Ways	XHP-6	JST	0.073	3	0.219
Wire-To-Board Connector, Right Angle, 2.5 mm, 2 C	S2B-XH-A	JST	0.094	6	0.564
Connector Housing, XH Series, Receptacle, 2 Ways	XHP-2	JST	0.05	6	0.3
SMD Chip Resistor, 0603 [1608 Metric], 1 kohm	WR06X1001FTL	Walsin	0.004	4	0.016
MD Chip Resistor, 0603 [1608 Metric], 100 kohm,	WR06X1003FTL	Walsin	0.001	3	0.003
Wire-To-Board Connector, Right Angle, 2.5 mm, 4 C	S4B-XH-A	JST	0.174	3	0.522
Connector Housing, XH Series, Receptacle, 4 Ways	XHP-4	JST	0.05	3	0.15
ALUMINUM ELECTROLYTIC CAPACITOR, 100UF,	EEE-FP1H101AP	Panasonic	0.286	3	0.858
Fixed LDO Voltage Regulator, 3.5V to 20V, 1.07V D	NCP1117ST33T3G	ON Semiconduct	0.251	2	0.502
ARM MCU, Value Line, STM32 Family STM32F0 Se	STM32F070RBT6	STMicroelectroni	3.76	1	3.76
USB Connector, Micro USB Type B, USB 2.0, Rece	10118193-0001LF	Amphenol	0.29	1	0.29
SMD Multilayer Ceramic Capacitor, 10 µF, 25 V, 08(	GRM21BR61E106MA73L	Murata	0.176	4	0.704
Red LED side mount	L-710A8EW/1LID	Kingbright	0.115	1	0.115
Yellow LED	L-710A8EW/1YD	kingbright	0.115	1	0.115
Green LED	L-710A8EW/1GD	Kingbright	0.115	1	0.115
Stepper Driver	a4988		0.65	3	1.95
Optical endstop			0.73	1	0.73
				Total	12.869

Handler

Mechanical Fixings

Item	Part Number	manufacturer	Unit Price	Units Used	Total
Custom 5 to 8mm rigid coupler			7.5	1	7.5
M3 20mm bolt			0.111	6	0.666
M3 30mm bolt			0.0758	4	0.3032
M3 10mm bolt			0.0966	12	1.1592
M5 nuts			0.0808	4	0.3232
M5 10mm bolt			0.0637	4	0.2548
M8x16mm bearing			3.51	2	7.02
Lock Collar 8mm			0.14	1	0.14
Motor Flex Coupler - 5mm x 8mm			0.76	2	1.52
Eccentric Spacers – 8mm Hex – 6mm Height			1.63	4	6.52
M5 Screw 15mm			1.35	2	2.7
M5 Screws 20mm			0.18	8	1.44
M5 Screws 25mm			0.085	2	0.17
M5 Screws 55mm			0.085	4	0.34
Aluminium Spacing 3mm			0.26	2	0.52
Aluminium Spacing 40mm			0.4	2	0.8
Aluminium Spacing 6mm			0.4	2	0.8
Rubber bands			0.01	4	0.04
M4 grub screw x 5mm			0.04	4	0.16
Hard Nut Lock 5mm			0.92	2	1.84
				Fixing Total:	**34.2164**

Appendix P: Motor comparison

	size	price	torque	inductance mh	max current	rated volt	steps	phase resistance	voltage usec	Power W
nema14	Φ36x12mm	$15.95	7Ncm	6.5	0.5	8.5	400	17	6	3
nema 14	35x35x26mm	$10.69	14Ncm	30	0.4	12	200	30	6	2.4
nema14	35x35x28mm	$9.73	12.5Ncm	3.5	1	3.5	200	3.5	6	6
nema14	35x35x34mm	$11.12	18Ncm	10	0.8	5.4	200	6.8	6	
Nema 17	42x42x20mm	**$9.31**	13Ncm	4.5	1	3.5	200		6	6
Nema17	42x42x25mm	$9.60	13Ncm	5.5	0.7	2.9	200	4.2	6	4.2
Nema17	42x42x33mm	$8.84	15.8	21	0.31	12	200	38.5	6	1.86
nema17	42x42x34mm	$8.57	26Ncm	37	0.4	12	200	30	6	2.4
nema17	42x42x34mm	$9.81	26Ncm	37	0.4	12	200	30	6	2.4
nema17	42x42x33mm	$8.84	16		0.95	4	200	4.2	6	5.7
nema17	42x42x39mm	$10.55	40Ncm	58	0.4	12	200	30	6	2.4
nema17	42x42x39mm	$9.36	45Ncm	4.4	1.5		200	2.3	6	9
nema23	57x57x41mm	$13.98	0.6Nm	21.7	0.88	6.6	200	7.5	6	5.28
nema23 dual	57x57x41	$14.29	0.55Nm	1.4	2.8	2	200		6	16.8
nema23	57x57x44mm	$13.98	0.6Nm	5.5	1.4	4.1	200	2.9	6	8.4

Appendix P: List of stepper motors compared from Stepper Online