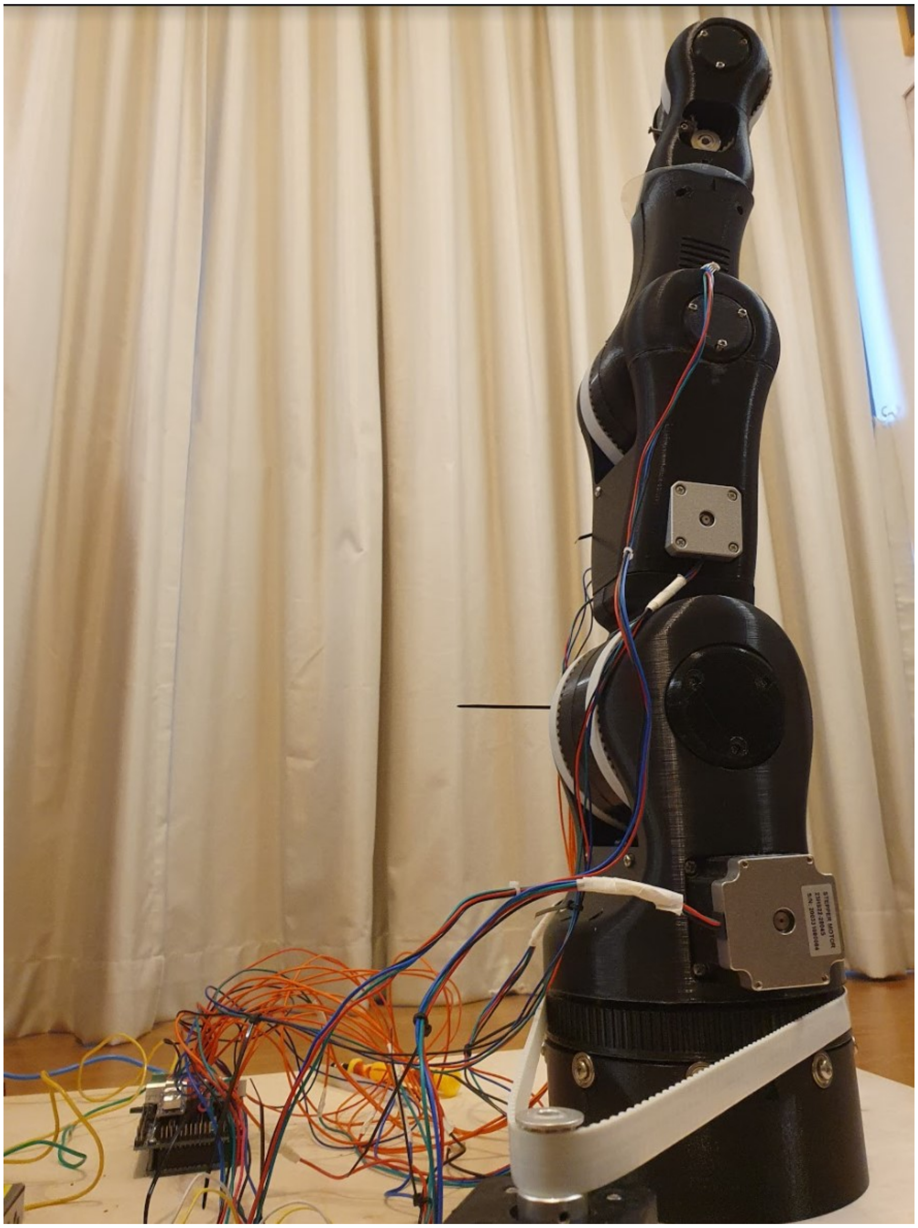
Engineering an articulated robotic arm to aid automated additive manufacturing microfactories



Abstract

Microfactories are small to medium scale manufacturing units with the ability to produce small dimension products. Their output can be scaled up by replicating such setups in large numbers (Descourvières et al., 2007). Specifically, additive manufacturing microfactories leverage 3D printing technology to develop an agile manufacturing environment: efficiently and economically producing mass-customized products (FutureBridge, 2020). The focus of this project is the design, manufacturing, and testing of a six-degree-of-freedom articulated robot to aid the automation of such microfactories. The implementation of automation retrofits, such as the robotic arm, will improve the productive capacity, return on capital, and safety of microfactories.

Introduction

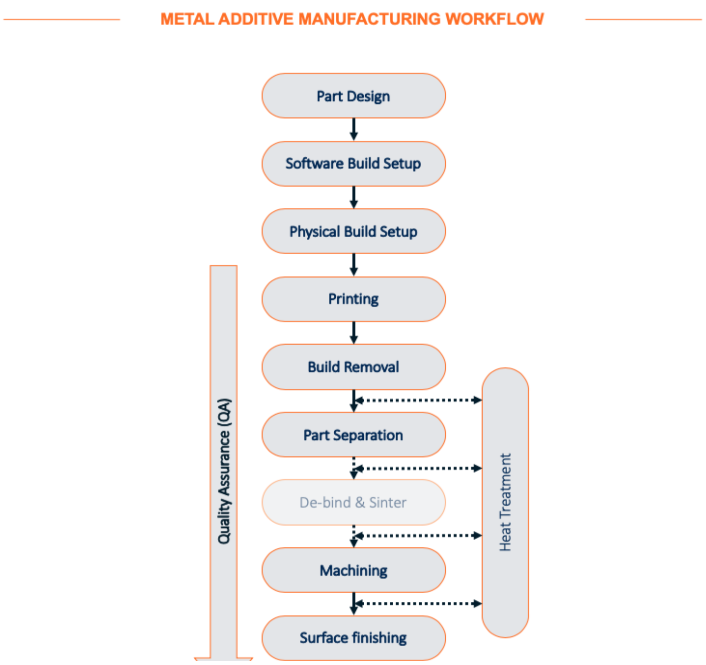
Skillset and interest alignment

The skills required to design, build, and test a 3D printed robotic arm will include Computer Aided Design (CAD) software, basic workshop machinery (traditional subtractive manufacturing and additive manufacturing like 3D printing), and electrical & electronic experience including microcontrollers such as Arduino. Over the past few years, I have become familiar with the aforementioned skills through the numerous projects and prototypes I have built (which can be seen on appendix link I). These include, but are not limited to, previously using a range of microcontrollers, sensors, and motors (including steppers, servos, and DCs). The skills and nuances I have learned through this would prove invaluable in the manufacturing and testing stages of my EPQ.

During the documentation and write-up stages, I found my patent writing experience has been of great help, especially since both, my patented device and my EPQ, are fairly technical and share the use-case driven development approach.

Another reason I chose a project such as this, is because of the close overlaps it has with the university course I would like to study: Design and Mechanical Engineering. The technical and research skills I have learnt, as discussed in detail below, will be a great foundation for the projects I will be involved in at university.

Use-case identification



(Digital Alloys, 2017)
Figure 1

The concept for this project stemmed from a documentary I came across, which discussed the plethora of industries 3D printing technology could disrupt (VICE News, 2020). A concept that is commonly associated with industrial 3D printing is the adoption of microfactory based manufacturing. Despite the multi-fold increase in efficiency and return on capital, there is still unused production capacity since machines need to be manned during the print cycle. Currently, the only step which does not require human intervention is the Printing stage. The aim of automation retrofits, such as the robotic arm, is to minimise human intervention from the Physical Build Setup stage, to the Build Removal stage. This enables an increase in efficiency as the next print can be started before the Part Separation stage has been carried out on the initial print. Voodoo Manufacturing, a global 3D printing microfactory firm, stated that they found a three times increase in productivity when using Project Skywalker - an automation retrofit (Engineering.com, 2018), a project that played a key role in my own research stage.

> *This “harvesting” process accounts for 10 percent of the total work taking place in the factory, resulting in a 3x productivity boost for all of the machines that are being run with robotic arms. When Voodoo employees arrive to work each day, they simply collect the batch of parts made overnight, remove them from their beds, and perform the post-processing required to ship the parts off to customers.*
> (Engineering.com, 2018)

The above statement was written by Michael Hou after an interview with Jonathan Schwartz, Co-Founder and CPO at Voodoo Manufacturing, in an article published on Engineering.com, an established site for developments in technology. Schwartz’s statement is fairly authoritative and reliable, as he is the Co-Founder of an internationally renowned manufacturing company. Although this may lead to the statement being slightly biased, as this would be in his benefit as a businessman. It is also fairly current as it was only written two year ago.
Michael, the journalist, is the Editor in Chief at 3Dprint.com, an additive manufacturing news site which has articles written by well-established industry professionals and university professors.

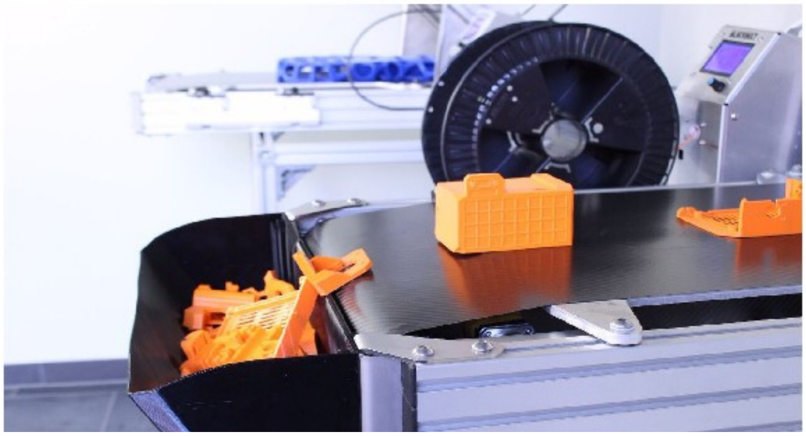
Research stage

Current industrial solutions

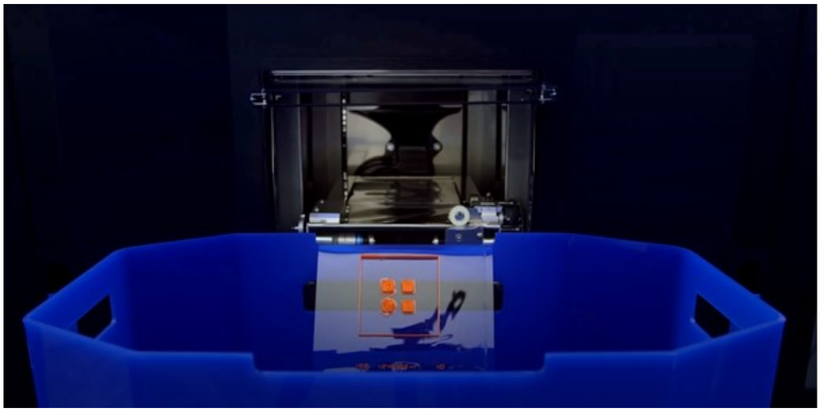
Over the course of my research, I found that most current industrial solutions fell into either one of two categories: ejecting the part through a conveyor belt setup, or a pick-and-place robotic arm to replace the printer bed.

Blackbelt's solution (*Figure 2*) is made out of Carbon Fibre (All3dp, 2017), this leads to an extremely high cost of prototype development and would also require extremely specialized workshop tools. I explored the idea to use alternative materials such as bendable acrylic sheets, although during the testing stage I found that the print would more often than not get stuck to the bed and would hence require me to use a metal blade to remove it. From this process I found that using materials other than carbon fibre would not work in the same way.

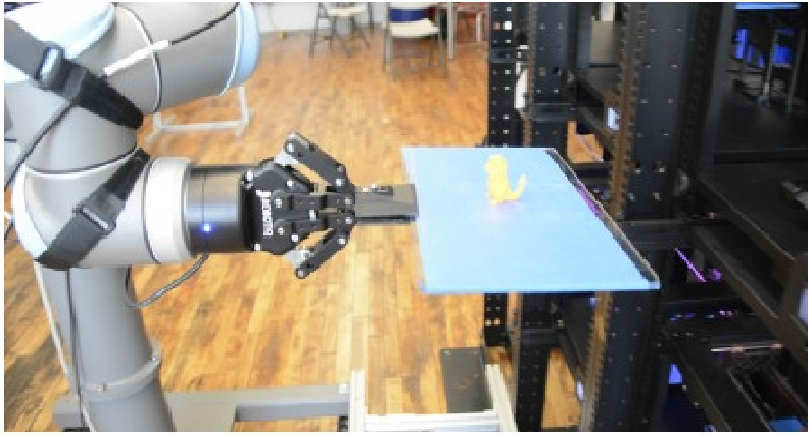
The Stratasys design (*Figure 3*) makes use of a lateral blade to cut the flexible bed, and much like the problem I faced with the first design, I could not land upon a material that would:
(i) be able to easily cut using a blade
(ii) not melt due to the hot bed



(Blackbelt, 2020)
Figure 2



(Stratasys, 2017)
Figure 3



(RBR, 2017)
Figure 4

(iii) have enough adhesion with the PLA filament that the print does not warp during printing, due to the hot bed. This led to exploring the third potential solution, a robotic arm.

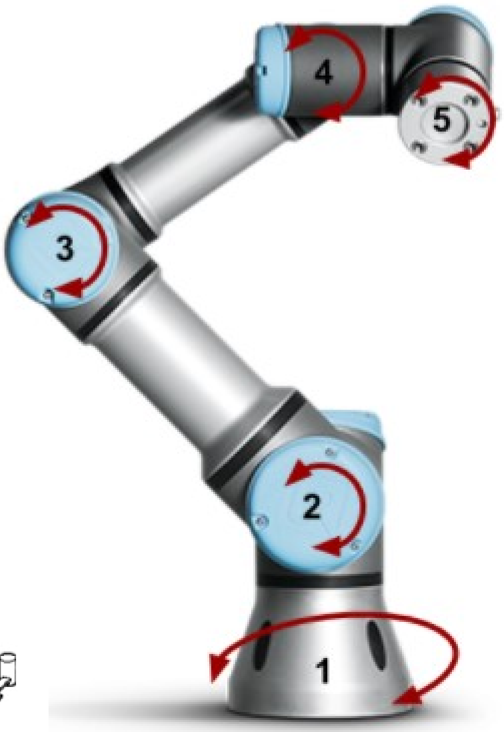
Project Skywalker, a robotic arm developed and implemented by Voodoo (Voodoo, 2017), is based on the pick and place design. The arm (*Figure 4*) has a gripper mechanism to replace a used print bed with a new one. Following this, the signal to start the next print cycle is sent.

Existing robotic arms

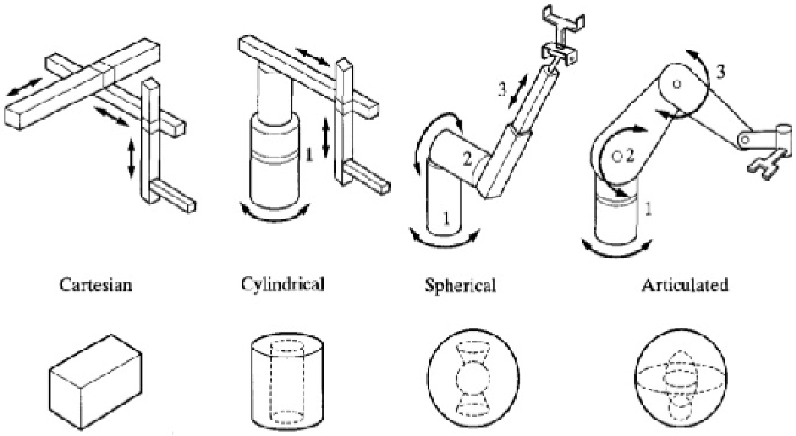
Once I decided a robotic arm was the most viable prototype to build, exploring and analysing current open source designs was the next step. Most open source arms I researched had a few commonalities, (i) 3D printed (ii) 5 or 6 Degrees of Freedom (iii) Used an Arduino as the microcontroller.

The term ‘Degrees of Freedom’ (DOF) refers to the number of independent movements a rigid body has (Zhang, n.d). For example, the articulated arm, also referred to as anthropomorphic robotic arm (*Figure 5*) has 5 Degrees of Freedom. This allows for the end effector, labelled as 5, to reach any point in the operating space as shown in the figure below (*Figure 6)*.

The end effector refers to the part of the robotic arm that would be considered the ‘hand’, interacting with the environment, in this case - the printer bed.



(Smlease, 2020)
Figure 5



(Massmind, 2020)
Figure 6

When deciding which 3D models, I should use as the base to build my robotic arm, I came across a range of open source projects, each greatly varied in the depth of documentation. Skyentific's 6 DoF 3D printed arm is a well-documented project; the build process, and ultimately testing, is shown through a three-part YouTube video (Skyentific, 2017). The arm was designed and built in 2018, meaning that the technologies he uses are fairly current, with only minor updates of software like Fusion 360 and the Arduino IDE happening since. The Skyentific YouTube channel, which is run by an engineer with over 10 years of practical experience, focuses on building 3D printed robotic arms and has 58000+ subscribers (YouTube, 2020). From seeing multiple videos, it is evident that he actively replies to comments and questions. This makes the source current, reliable, and authoritative.

Another source I heavily used is Toglefritz's Instructable guide. 'Build a Giant 3D Printed Robot Arm' is a detailed documentation on how Scott Hatfield 3D prints and builds a 6 DOF robotic arm (Toglefritz, 2017). His project is based on BCN3D MOVEO's 'Fully Open Source 3D printed robot arm' (BCN3D, 2019). It is constructed using ABS (Acrylonitrile butadiene styrene) filament, a common filament used in 3D printing. Although, the filament I had access to from school was mostly PLA (Polylactic acid).

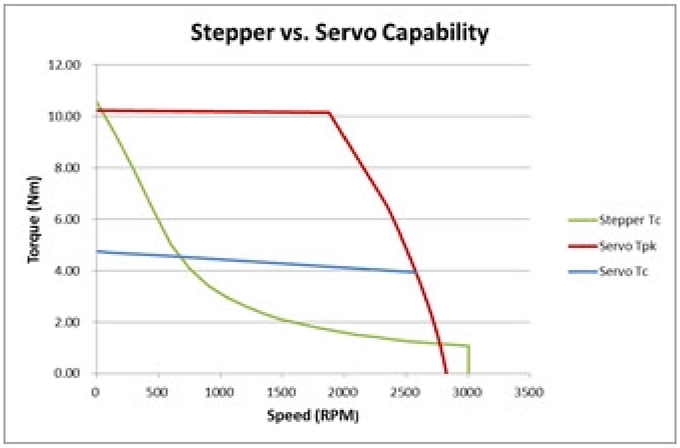
When weighing up if I should order, and hence wait for, ABS filament, I learnt that ABS is stronger, more flexible, and more durable (All3dp, 2019). Considering that the load on the robotic arm would be relatively low, at least for testing stages, I decided that I would go ahead and print using PLA filament, which I already had access to at school. Another factor that I considered was the safety of using these filaments in closed printing environments; in later stages of my project I found myself printing in my living room at home. PLA filament is safe to print without an extractor as it does not release toxic fumes, unlike ABS filament (All3dp, 2019). This information can also be found on numerous other websites; hence it can be taken as credible and reliable. All3dp's articles are written by professional engineers, and are updated frequently to match any innovations in the technology, making this source current and accurate.

With two fairly well documented open source robotic arms which I can use as reference, I went on to the next stage of the build: part selection and ordering.

Part selection and sourcing

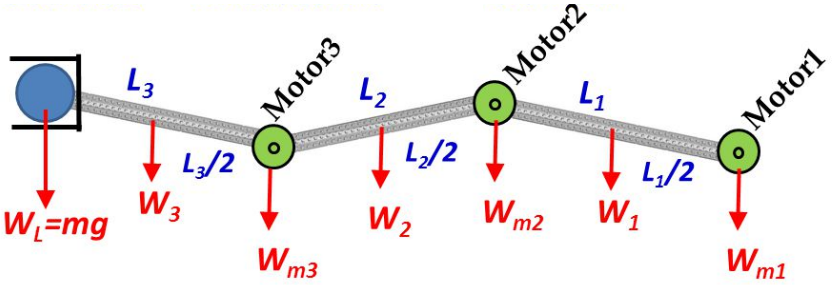
Referring to the parts used to build Toglefritz's 'Giant Robot Arm' (Toglefritz, 2017), I came up with a Bill of Materials (BOM). A decision I had to make was whether to use stepper or servo motors to actuate the arm. Both open source arms, referred to in the previous section, make use of stepper motors. Although, other arms which I came across in my research stage made use of servo motors. What I deduced from this research was that it mainly depended on the intended payload. Servo motors were easier to implement as they, themselves, are smaller and lighter as they are designed to carry a lower payload. On the other hand, stepper motors are much heavier, get hotter, and require much more current, although are able to carry a much larger payload.

The graph (*Figure 7*) compares the Torque-Speed relationship between stepper and servo motors (Gill, 2018). The red line represents the peak running of a servo motor, where excessively high currents are used hence not sustainable; the blue line represents the continuous running of a servo motor; the green line represents the continuous running of a stepper motor. When analysing the continuous running behaviour of the two motors, it is obvious that a stepper motor exponentially increases torque when speed (RPM) is reduced, as compared to servo motors which have a fairly constant torque despite the speed. Considering the intended speed of movement of the robotic arm (RPM required), and payload of the arm, I decided to use stepper motors.



(Gill, 2018)
Figure 7

When choosing which stepper motors to use for each joint of the arm, I had to take into account the principle of moments: for transitional and rotational equilibrium there must be a resultant moment of 0 *Nm* (Thorogood, 2015). The diagram (*Figure 8*) represents the increasing moment acting upon each motor (joint), as it goes further away from the base. In essence, the motors closest to the base will need to have the highest payload capacity as they will counter the moments induced by the sections' weight acting further away from the pivot.



$$\tau_3 = (mg \times L_3) + (W_3 \times L_3 / 2)$$

$$\tau_2 = \left[mg \times (L_3 + L_2)\right] + \left[W_3 \times (L_2 + {}^{L_3}\!/\!_2)\right] + (W_{m3} \times L_2) + (W_2 \times {}^{L_2}\!/\!_2)$$

$$\tau_1 = \left[mg \times (L_3 + L_2 + L_1)\right] + \left[W_3 \times (L_1 + L_2 + {}^{L_3}\!/\!_2)\right] + \left[W_{m3} \times (L_1 + L_2)\right] + \left[W_2 \times (L_1 + {}^{L_2}\!/\!_2)\right] + (W_{m2} \times L_1) + (W_1 \times {}^{L_1}\!/\!_2)$$

(Thorogood, 2015)
Figure 8

Upon finalising the motors, I curated the final BOM (*Figure 9*). Most of the mechanical hardware components are fairly standard and were easily available from websites like Aliexpress.com. The stepper motors were ordered from omc-stepperonline.com since they were fairly specialized and hence not as commonly available. One major problem I faced when ordering the parts was ensuring they would couple (join when assembling). This was not made easier by the fact that most of the Aliexpress descriptions were in Chinese. To overcome this, I asked a few of my friends for translation help. I also took a few risks with the parts as some of them lacked descriptions, and I had to decode the specifications from the photos provided.

		Part	Quantity	Approx. Cost (SGD)
Components	Bearings	8mm x 22mm x 7mm Bearing	10	$12.43
		5mm x 16mm x 5mm Bearing	8	
		4mm x 13mm x 5mm Bearing	9	
		3mm x 10mm x 4mm Bearing	3	
	Rods	M8 x 40mm Threaded Rod	1	$4.84
		8mm x 140mm Smooth Rod	1	$13.11
		8mm x 115mm Smooth Rod	1	
		8mm x 80mm Smooth Rod	1	
	Belts	65cm T5 Belt	1	$22.00
		50cm T5 Belt	1	
		35cm T5 Belt	1	
	Pulleys	T5 Pulley 8mm Bore	2	$29.37
		T5 Pulley 1/4" Bore	1	
		T5 Pulley 5mm Bore	1	
	Couplers	5mm to 8mm Shaft Coupler	2	$4.21
Fasteners	Heat-Set Inserts	M3 Heat-Set Inserts	41	$16.98
		M4 Heat-Set Inserts	22	
		M5 Heat-Set Inserts	16	
	Screws	M5 x 14mm Screw	16	$55.00
		M8 x 65mm Screw	1	
		M3 x 25mm Screw	5	
		M4 x 20mm Screw	7	
		M4 x 40mm Screws	11	
		M3 x 40mm Screw	7	
		M3 x 10mm Screw	20	
		M4 x 10mm Screw	4	
		M3 x 16mm Screw	10	
		M3 x 30mm Screw	1	
		M4 x 55mm Screw	4	
		M4 x 45mm Screw	4	
		M3 Thread-Forming Screw	4	Already have (DT)
	Nuts	M3 Nut	4	$0.77
		M4 Nut	14	$1.40
		M8 Locknut	1	$4.10
		M3 Washer	7	$0.77
	Crimps	Crimps	25	$8.73
		Wire	100	$18.00
Electronics	Motors	Nema 23 Motor	2	$71.14
		Nema 17 Motor	1	$14.36
		Nema 17 Long Motor	1	$30.98
		Nema 14 Motor	1	$17.79
		Nema 17 5:1 Geared Motor	1	$37.93
	Controller	Arduino Mega 2560		Already have
		Electromagnet module	1	$10.00
		RAMPS Shield	1	$9.11
		DRV8825 Stepper Drivers	5	$12.08
		24V 240W Power Supply	1	$19.80
		Power Supply Interfaces	1	$10.00
			Approx.	$416.40

Figure 9

Another major decision I had to make was which power supply I would use. I was contemplating between a 12V and a 24V PSU. The RAMPS shield, to run the stepper motors, was made for 24V, although the Arduino board could only handle 12V. After a lot of testing and researching, I figured that the Vin pin on the Arduino is what led to the voltage limit, and if I managed to desolder that, I could connect the 24V 240W power supply to the RAMPS shield without burning the fuse on the Arduino. Since the position of the Vin pin was very awkward, I had to bend the tip of my soldering iron to be able to reach it (*Figure 10*).

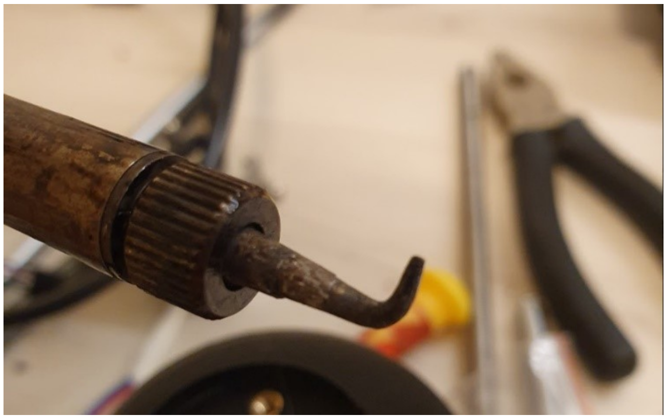


Figure 10

As I started receiving the orders, I started the build process. Below is a photo of the parts as they came during the peak of the lockdown. To document this further, I have created a video montage, referred to later. Along with the stepper motors, I received the respective data sheets, which allowed me to understand better the technicalities behind each stepper motor and how individual parameters were calculated.

Figure 11

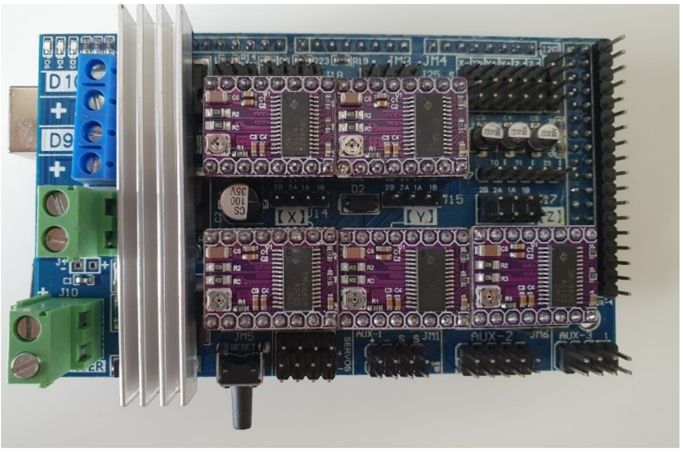
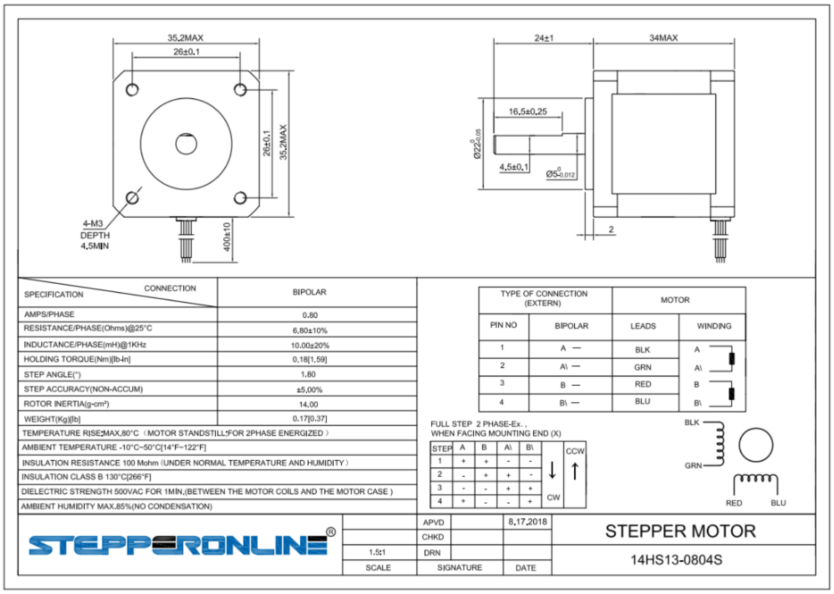


Figure 12



SPECIFICATION \ CONNECTION	BIPOLAR
AMPS/PHASE	0.80
RESISTANCE/PHASE(Ohms)@25°C	6,80±10%
INDUCTANCE/PHASE(mH)@1KHz	10.00±20%
HOLDING TORQUE(Nm)[lb-In]	0,18[1,59]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5,00%
ROTOR INERTIA(g-cm²)	14,00
WEIGHT(Kg)[lb]	0.17[0.37]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL:FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

TYPE OF CONNECTION (EXTERN)		MOTOR	
PIN NO	BIPOLAR	LEADS	WINDING
1	A —	BLK	A
2	A\ —	GRN	A\
3	B —	RED	B
4	B\ —	BLU	B\

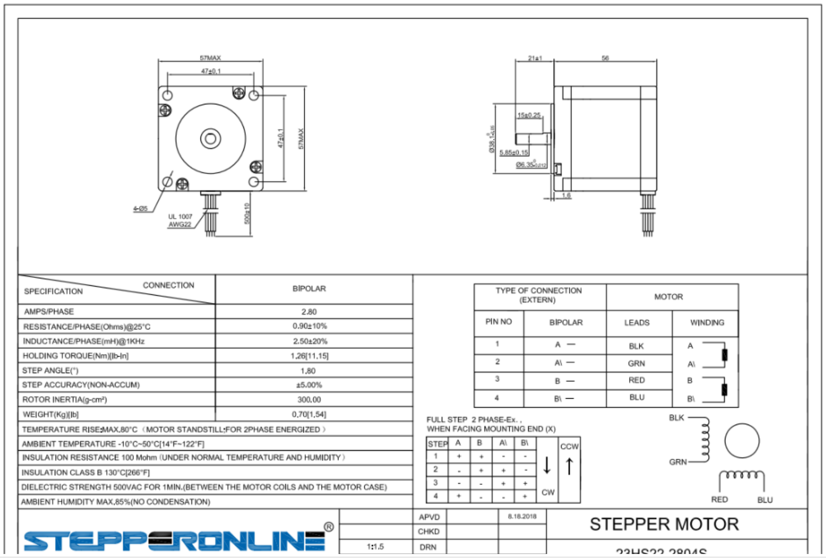
FULL STEP 2 PHASE-Ex. ,
WHEN FACING MOUNTING END (X)

STEP	A	B	A\	B\	↓ CW	CCW ↑
1	+	+	-	-		
2	-	+	+	-		
3	-	-	+	+		
4	+	-	-	+		

STEPPERONLINE®

	APVD		8.17.2018	STEPPER MOTOR
	CHKD			
1.5:1	DRN			14HS13-0804S
SCALE	SIGNATURE		DATE	

The datasheet for the NEMA 14 motor (Stepperonline, 2020)
Figure 13



SPECIFICATION \ CONNECTION	BIPOLAR
AMPS/PHASE	2.80
RESISTANCE/PHASE(Ohms)@25°C	0.90±10%
INDUCTANCE/PHASE(mH)@1KHz	2.50±20%
HOLDING TORQUE(Nm)[lb-In]	1,26[11,15]
STEP ANGLE(°)	1,80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	300,00
WEIGHT(Kg)[lb]	0,70[1,54]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL:FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

TYPE OF CONNECTION (EXTERN)		MOTOR	
PIN NO	BIPOLAR	LEADS	WINDING
1	A —	BLK	A
2	A\ —	GRN	A\
3	B —	RED	B
4	B\ —	BLU	B\

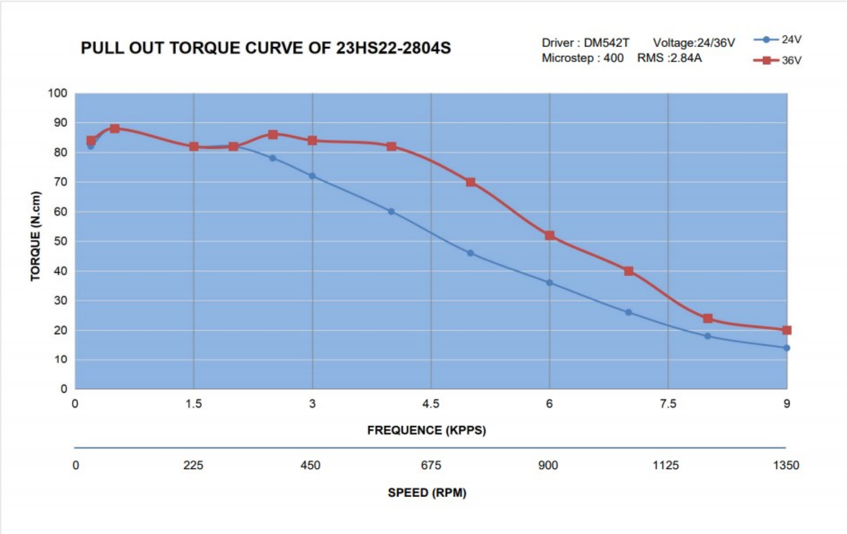
FULL STEP 2 PHASE-Ex. ,
WHEN FACING MOUNTING END (X)

STEP	A	B	A\	B\	↓ CW	CCW ↑
1	+	+	-	-		
2	-	+	+	-		
3	-	-	+	+		
4	+	-	-	+		

STEPPERONLINE®

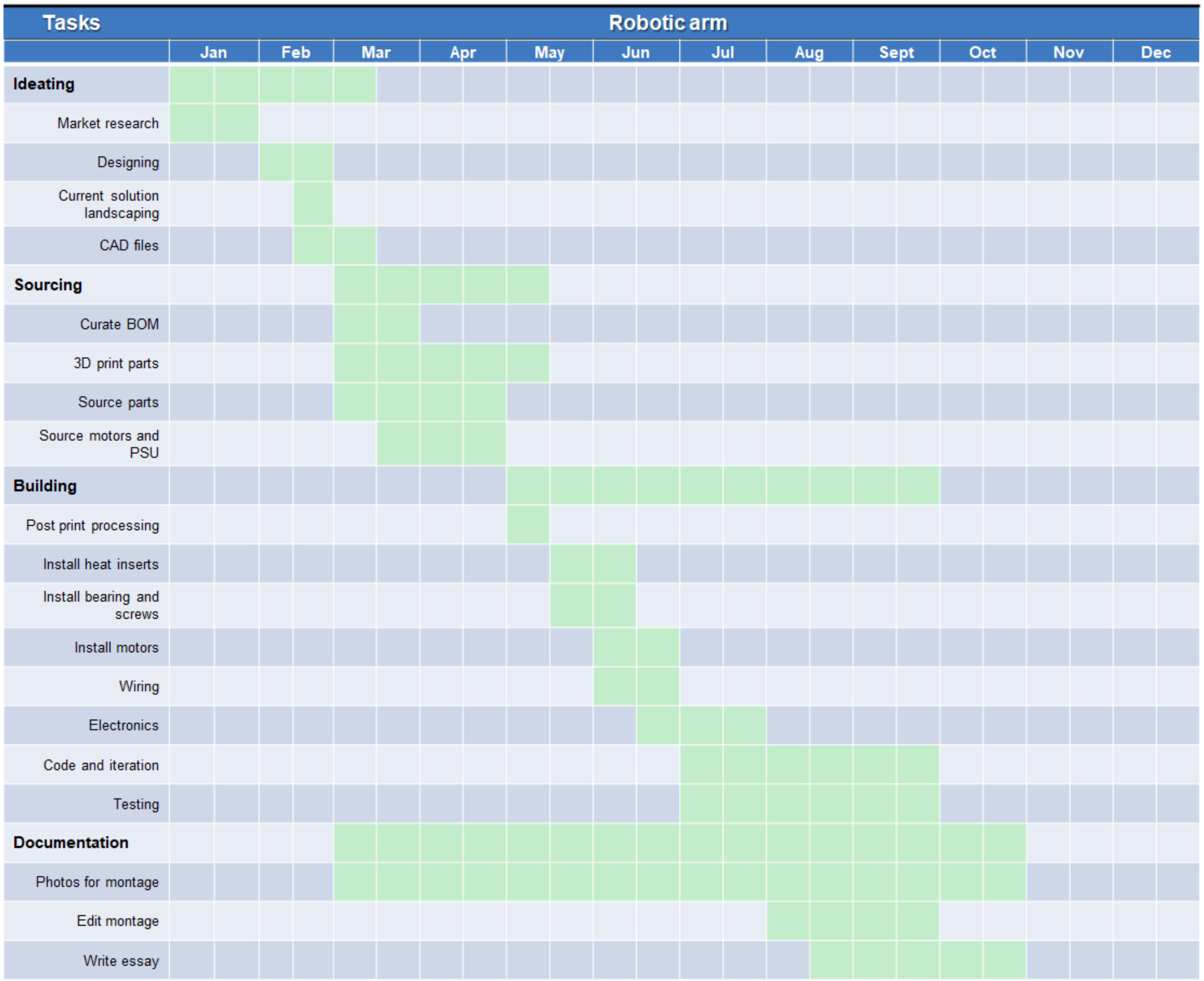
	APVD		8.18.2018	STEPPER MOTOR
	CHKD			
1:1.5	DRN			23HS22-2804S

The datasheet for the NEMA 23 motor (Stepperonline, 2020)
Figure 14



The Torque-Speed curve for the NEMA 23 motor (Stepperonline, 2020)
Figure 15

Manufacturing stage



Tasks	Robotic arm																						
	Jan		Feb		Mar		Apr		May		Jun		Jul		Aug		Sept		Oct		Nov		Dec	
Ideating	🟩	🟩	🟩	🟩	🟩																			
Market research	🟩	🟩																						
Designing			🟩	🟩																				
Current solution landscaping				🟩																				
CAD files				🟩	🟩																			
Sourcing					🟩	🟩	🟩	🟩	🟩															
Curate BOM					🟩	🟩																		
3D print parts					🟩	🟩	🟩	🟩	🟩															
Source parts					🟩	🟩	🟩	🟩																
Source motors and PSU						🟩	🟩	🟩																
Building									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩						
Post print processing									🟩															
Install heat inserts										🟩	🟩													
Install bearing and screws										🟩	🟩													
Install motors											🟩	🟩												
Wiring											🟩	🟩												
Electronics												🟩	🟩	🟩										
Code and iteration													🟩	🟩	🟩	🟩	🟩	🟩						
Testing													🟩	🟩	🟩	🟩	🟩	🟩						
Documentation					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Photos for montage					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Edit montage															🟩	🟩	🟩	🟩						
Write essay																🟩	🟩	🟩	🟩	🟩				

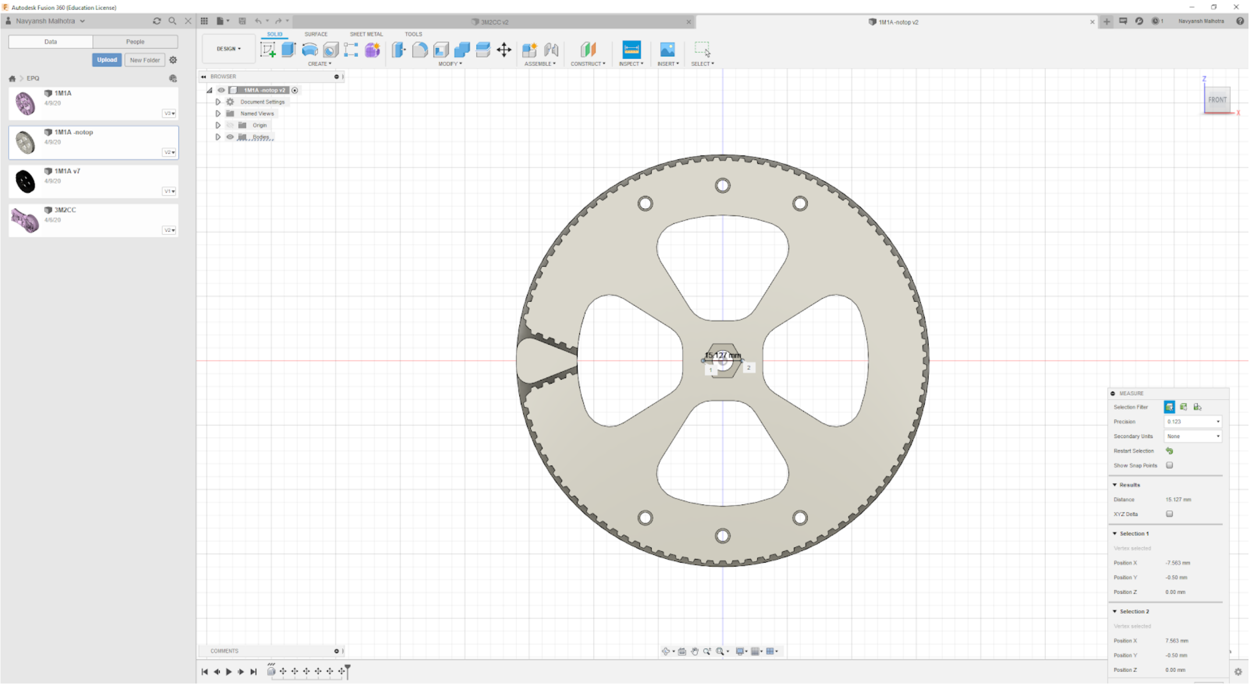
Figure 16

The Gantt chart above splits up the different stages of this project, right from ideation, to documentation. The below section highlights the building stage: 3D printing the parts, the mechanical & electronic assembly, and the coding.

3D printing

In order to 3D print a model, you need to design a Computer Aided Design (CAD) file. For this, I used Autodesk Fusion 360, a software I am familiar with for designing, editing, and testing CAD components. I was able to use a lot of the BCN3D files, after editing them slightly. This is an open source repository of STL files for 3D printed robotic arms. These files have been tested numerous times by 3D printing hobbyists and educators (BCN3D, 2019). I went through every part individually to ensure that they were designed for the parts I had ordered. One part for example, the base rotating plate's inner bolt's spacing was 12mm, although I had to expand this to 15mm (*Figure 17*) so that the bolt would fit.

This process was more complicated than I expected since the file from the open source repository for this part was a mesh file, which cannot be directly edited on Fusion 360. I had to therefore convert the file to STL using a process known as vectorising. I had never done this before, although it certainly proves a useful skill, especially for reverse engineering parts.



MEASURE
Selection Filter
Precision: 0.123
Secondary Units: None
Restart Selection
Show Snap Points ☐

Results
Distance: 15.127 mm
XYZ Delta ☐

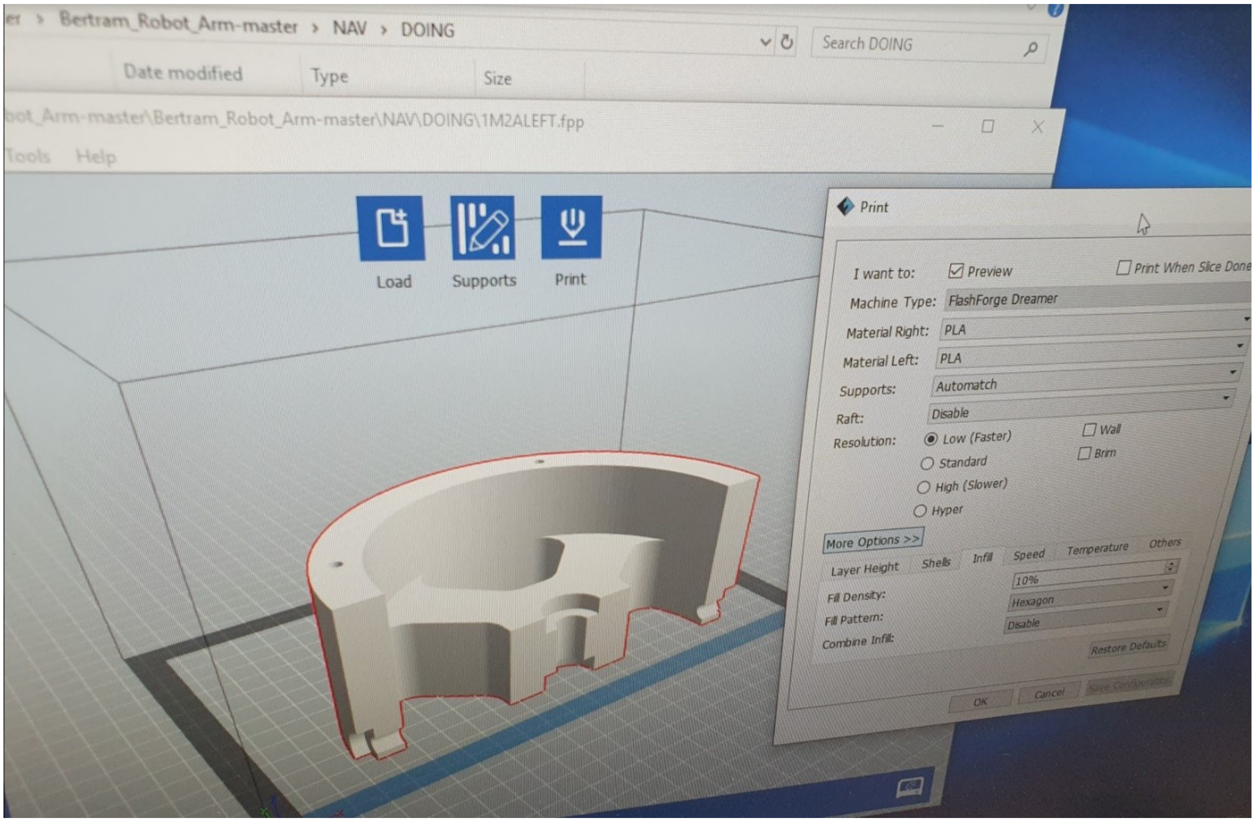
Selection 1
Vertex selected
Position X: -7.563 mm
Position Y: -0.50 mm
Position Z: 0.00 mm

Selection 2
Vertex selected
Position X: 7.563 mm
Position Y: -0.50 mm
Position Z: 0.00 mm

Figure 17

After ensuring all the parts' designs were correct and would mate during assembly, I went on to actually print these CAD files. I was fairly familiar with the printers at school, and hence decided that I would print the parts there.

When setting up the prints there were a few parameters that I could configure: Fill Density, Raft or Brim, and Quality of Finish. The fill density plays a pivotal role in the strength of the print, although there is an exponential relationship between time taken and fill density. After referring to numerous websites, and the DT teacher, Mr Bleach, I decided on printing at a 15% in-fill. Below (*Figure 18*) is an example of the interface of FlashForge, the software which interacts with the 3D printer.



Print

I want to: ☑ Preview ☐ Print When Slice Done
Machine Type: FlashForge Dreamer
Material Right: PLA
Material Left: PLA
Supports: Automatch
Raft: Disable
Resolution: ● Low (Faster) ○ Standard ○ High (Slower) ○ Hyper
☐ Wall
☐ Brim
Fill Density: 10%
Fill Pattern: Hexagon
Combine Infill: Disable

Figure 18

I had been fairly lucky that the COVID situation did not impact my project significantly, at least until this stage. In March, school closed, and hence I did not have access to the 3D printers anymore. I was done printing with around half the parts. Since I did not have a printer at home, I reached out to a Maker Community Facebook Group. That is where I got my hands on to an Ender 3, a large volume 3D printer I was kindly given to use free of cost. Although, the printer needed some fixing before it could be used.

Figure 19

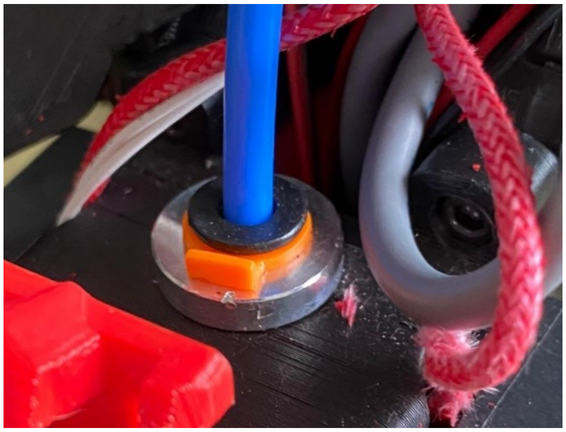
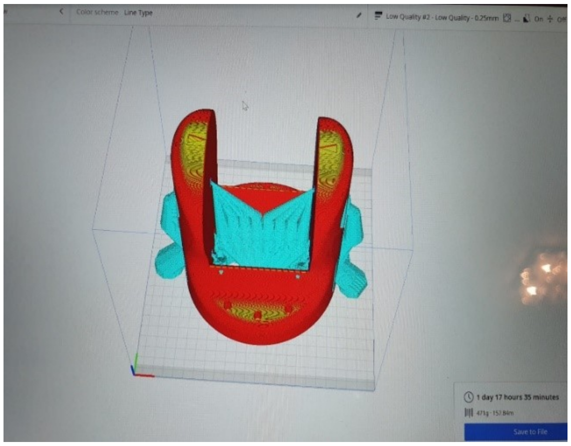


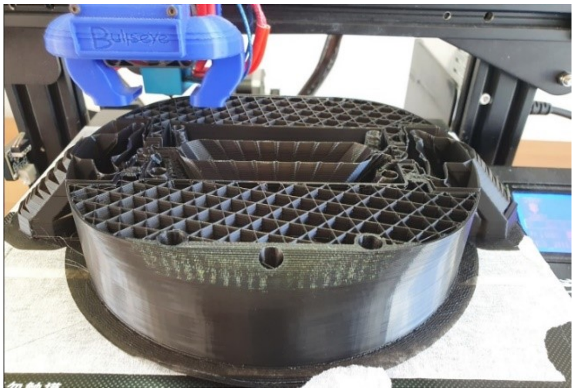
Figure 20

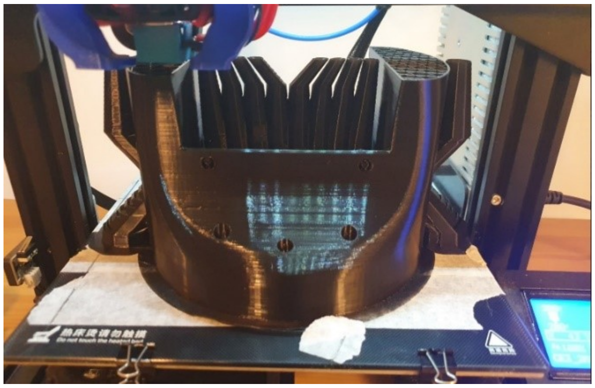
I had never built a 3D printer before and hence opening up and replacing parts of a 3D printer was a new experience for me. After running a few tests, I realised the printer had a jammed nozzle. The stem of the problem was the deposited black filament

(*Figure* 19), this led the bowden tubing to melt and get blocked. The easiest solution would be to replace the part, and that is what I ended up doing. Although frustrating at the time, I definitely got a much better understanding of the inside working of a 3D printer.

Below is an example of a part being fabricated, from the slicing stage to the post-print stage. The slicing stage refers to a CAD file being broken down into layers, converting designs into points on a cartesian plane, so that the nozzle of the 3D printer knows where to eject filament (All3dp, 2020).
Appendix link III is a video I made showing how a part is broken down into slices, and how each layer is built up, making the final 3D printed part.







Once all the parts were printed, a process known as support removal had to be carried out on all the parts. Whilst I knew this was a tedious task, I did not expect it to take nearly as much time as it did. Supports are printed as part of the design as they allow for overhanging sections.

Figure 21

After I was done printing all the parts I learnt about a filament (the plastic for the 3D printer) which dissolves in water (3DInsider, 2018). Using a dual nozzle primer, you are able to simply dip the part in water to remove all the support structures. Although I had to do this using a knife, pliers, and a screwdriver for this project, it is something I will keep in mind for the future.

Figure 22

Below (*Figure* 23) is a photo of all the 3D printed components, after which I started the assembly process.

Figure 23

Assembly

I started the mechanical assembly by inserting heat inserts into all the parts. Heat inserts add threads to thermoplastic parts, allowing nuts and bolts to be used (Hackday, 2019). I had around 50 inserts of various sizes to attach; I used a plain-vanilla soldering iron as a workaround to using the suggested tool (*Figure 24*), since shipping for this tool would delay my project.

(Hackday, 2019)
Figure 24

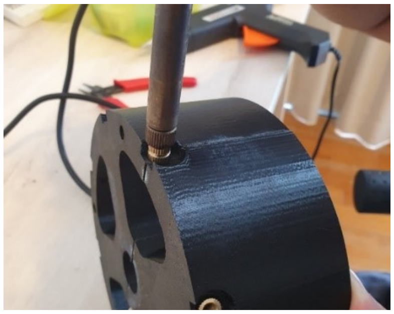


Figure 25

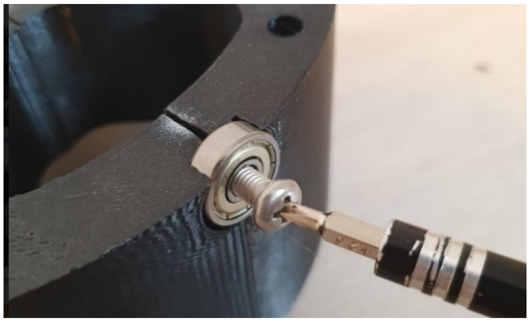


Figure 26

The design of the arm can be broken down into 3 main sections (bottom to top of the arm): Shoulder, Elbow, and Wrist. Each one is mechanically bound and controlled in the same way, the only difference being sizes. The sizes are designed in a way which allows for a 'telescopic joint' (*Figure 28*). Each joint has a 5mm stainless steel rod with 2 ball bearings on either side (*Figure 29*). The stainless-steel rod had to be cut to size using a power tool (*Figure* 27), since it was hardened 304 stainless steel (Makerslegacy, n.d).

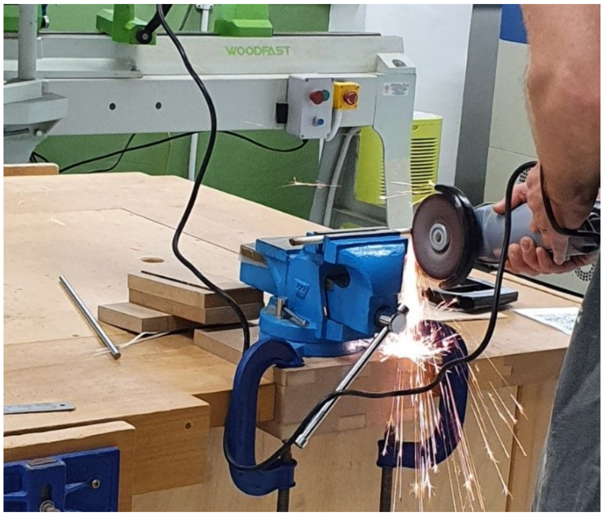


Figure 27

Figure 28

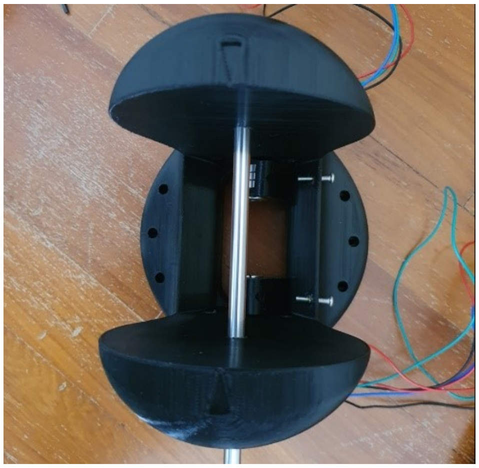


Figure 29

Figure 30

For every joint, the axle had a timing belt pulley (*Figure 31*). The other end of the timing belt would be connected to the belt pulley on the motors.

The holes printed for the axle were exactly 5mm in diameter, although since the rod was exactly 5mm too, the rod would not pass through. To overcome this problem, I had to use a blowtorch to heat the plastic slightly and push it inwards by increments of 0.1mm (*Figure 31*). I have learnt to keep a 0.5mm tolerance on either side when designing parts in the future.

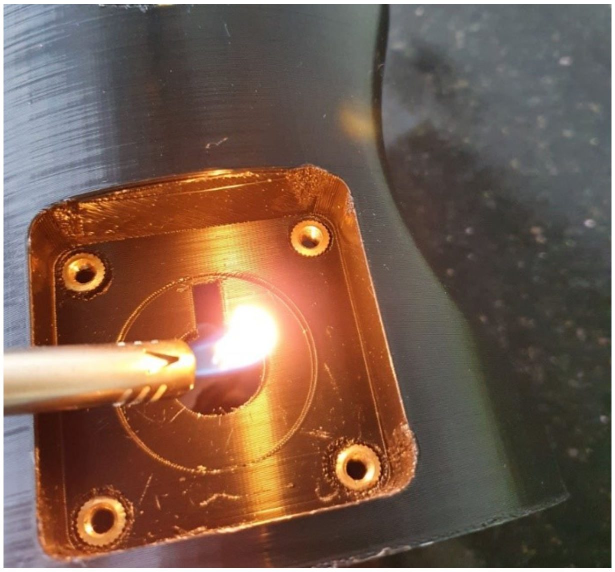


Figure 31

The figure on the right (*Figure* 32) shows the arm when the three joints were connected and attached. This was the first time the overall shape of an arm was starting to form, instead of just individual components.

The final step of the mechanical build was attaching the 3 sections to the rotating base mounted on a piece of plywood (*Figure 33*). This not only holds the main body and base motor in place, but also makes transporting the arm easier (for the testing stage).
The images below (*Figure 34 and Figure* 35) show the cutting of the plywood followed by the mounting of the arm and the supplementary electronic components.

Figure 32

Figure 33

Figure 34

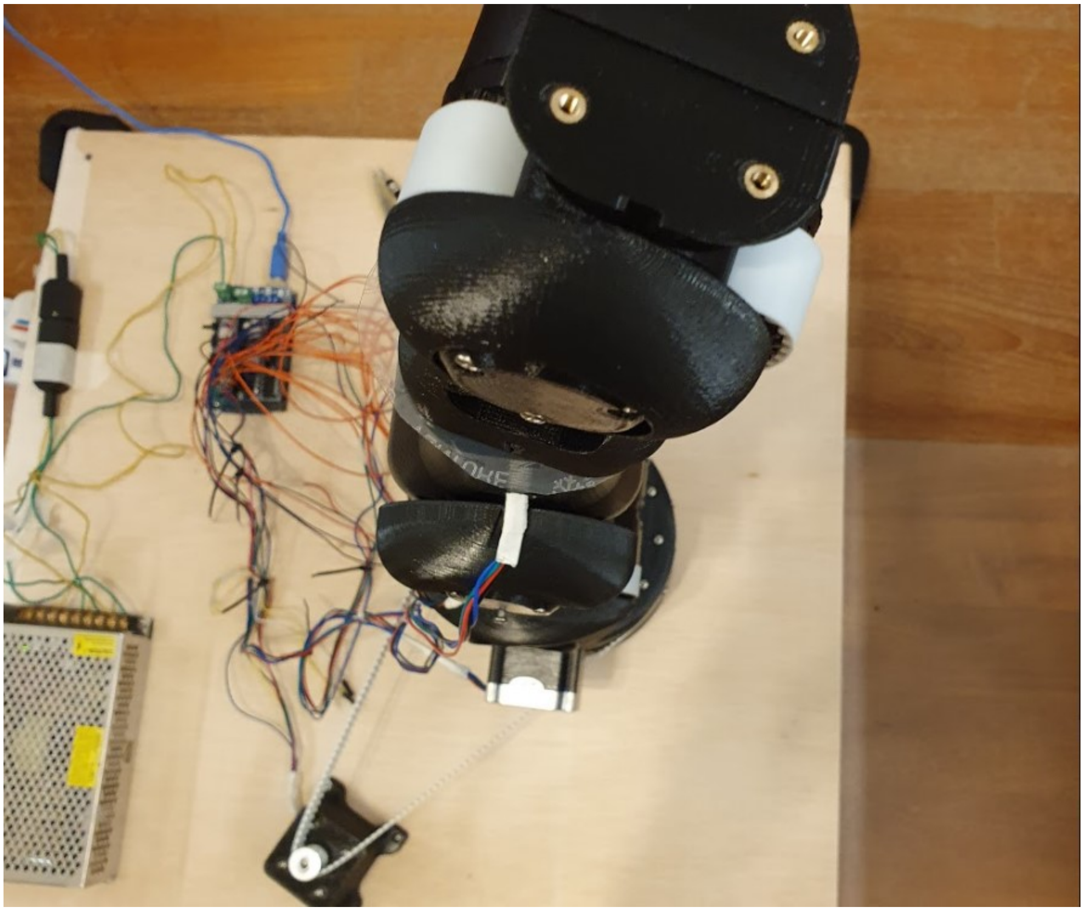


Figure 35

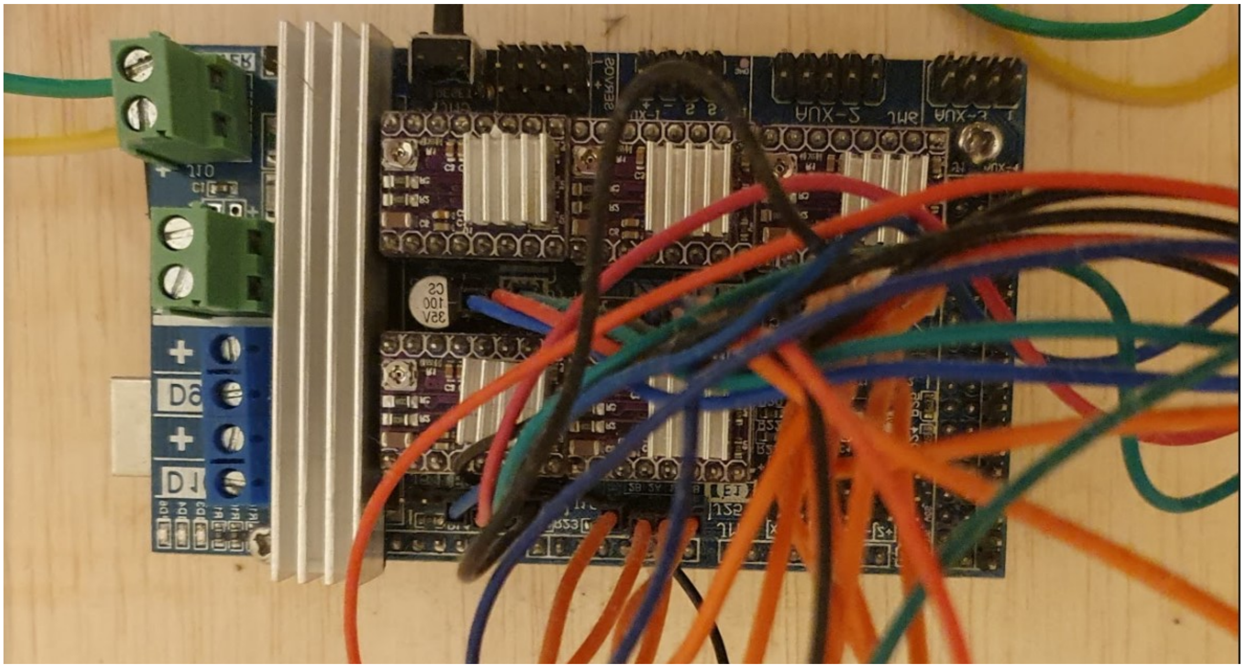


Figure 36

The final step to complete the entire build was the electronic system. This process happened after the Isolated component testing stage, detailed below.

At the core of the electronics there are 5 main components: Arduino Mega, RAMPS 1.6 Stepper Controller, 24V 240W PSU, 6 DRV 8826 Stepper drivers, and the 6 Stepper motors.
The RAMPS shield is placed on the Arduino control board, and for each axis, there is a DRV 8836 stepper driver (*Figure 36*). The entire RAMPS shield is connected to the 24V power supply, and the Arduino is connected to a computer for control commands, as detailed below. There are only 5 controllers for 6 motors, as on the X-axis, corresponding to the shoulder section of the arm, there are two motors which run in tandem.

Since I did not find a push button switch which was rated for 10 amps (the currents supplied by the power supply), there was no way of switching off the power supply to the robotic arm without switching off the mains. Therefore, I wired a plug which would control the power supply (*Figure 37*).

Appendix link IV is a link to the time lapse which shows the full build process.

Figure 37

Coding

In order to run the mechanical side of the project, I had to upload code which could take commands and output the relevant signals to each of the motors as required. Most 3D printers are very similar in electronic architecture: A stepper controller with multiple stepper motors. This is what led me to repurpose open source 3D printer firmware. A common open source firmware for 3D printers is Marlin. Toglefritz's robotic arm also made use of Marlin as the firmware (Toglefritz, 2017).

Below is a screenshot of the code which runs in the backend (*Figure 38*). The only difference between the 3D printer firmware and this code is that the hotplate does not run in a robotic arm, and hence those pins are not enabled.

Apart from the firmware of the robotic arm, I set up a system known as OctoPi. OctoPi is a remote monitoring system for 3D printers (Häußge, n.d). I was hoping to use OctoPi in tandem with the robotic arm to automate the microfactories. The OctoPi interface allows me to see the print in real time along with various other parameters of the 3D printer (*Figure 39 and Figure 40*).

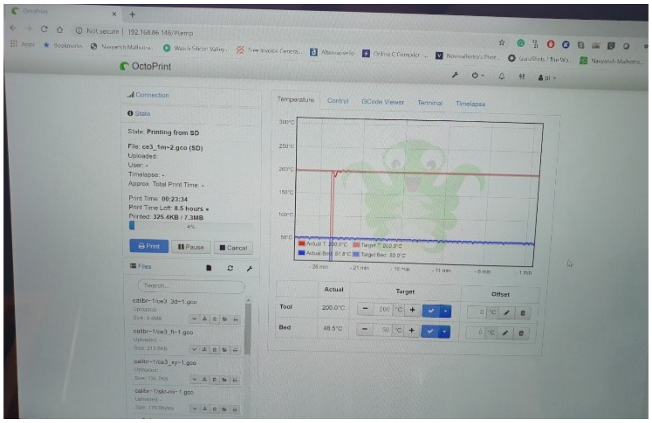


Figure 39

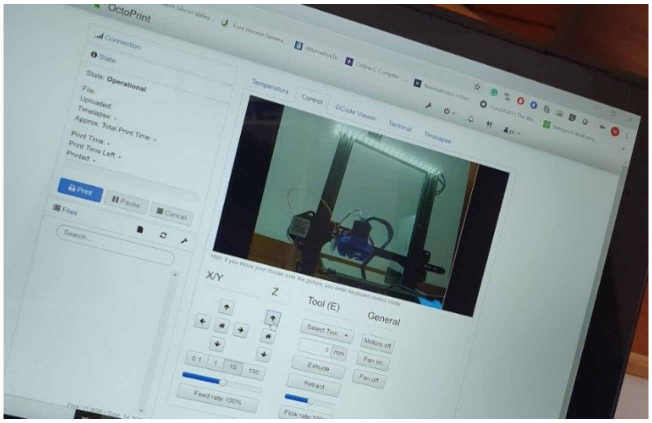
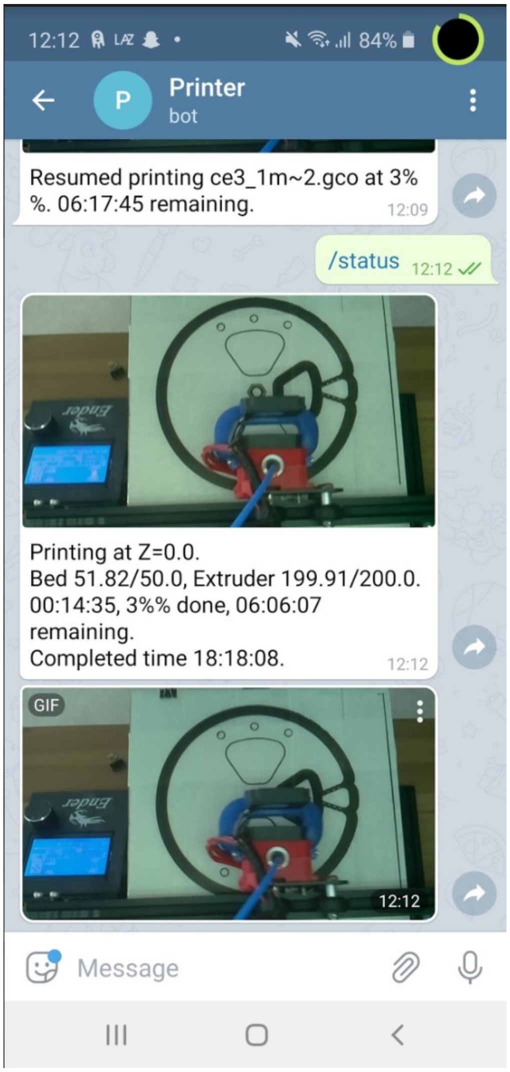


Figure 40

```
#include "Configuration.h"
#include "pins.h"


#ifdef ULTRA_LCD
  #if defined(LCD_I2C_TYPE_PCF8575)
    #include <Wire.h>
    #include <LiquidCrystal_I2C.h>
  #elif defined(LCD_I2C_TYPE_MCP23017) || defined(LCD_I2C_TYPE_MCP23008)

void get_command();
void process_commands();

void manage_inactivity(bool ignore_stepper_queue=false);

#if defined(DUAL_X_CARRIAGE) && defined(X_ENABLE_PIN) && X_ENABLE_PIN > -1 \
    && defined(X2_ENABLE_PIN) && X2_ENABLE_PIN > -1
  #define  enable_x() do { WRITE(X_ENABLE_PIN, X_ENABLE_ON); WRITE(X2_ENABLE_PIN, X_ENABLE_ON); } while (0)
  #define disable_x() do { WRITE(X_ENABLE_PIN,!X_ENABLE_ON); WRITE(X2_ENABLE_PIN,!X_ENABLE_ON); axis_known_position[X_AXIS] = false; } while (0)
#elif defined(X_ENABLE_PIN) && X_ENABLE_PIN > -1
  #define  enable_x() WRITE(X_ENABLE_PIN, X_ENABLE_ON)
  #define disable_x() { WRITE(X_ENABLE_PIN,!X_ENABLE_ON); axis_known_position[X_AXIS] = false; }
#else
  #define enable_x() ;
  #define disable_x() ;
#endif

#if defined(Y_ENABLE_PIN) && Y_ENABLE_PIN > -1
  #ifdef Y_DUAL_STEPPER_DRIVERS
    #define  enable_y() { WRITE(Y_ENABLE_PIN, Y_ENABLE_ON); WRITE(Y2_ENABLE_PIN,  Y_ENABLE_ON); }
    #define disable_y() { WRITE(Y_ENABLE_PIN,!Y_ENABLE_ON); WRITE(Y2_ENABLE_PIN, !Y_ENABLE_ON); axis_known_position[Y_AXIS] = false; }
  #else
    #define  enable_y() WRITE(Y_ENABLE_PIN, Y_ENABLE_ON)
    #define disable_y() { WRITE(Y_ENABLE_PIN,!Y_ENABLE_ON); axis_known_position[Y_AXIS] = false; }
  #endif
#else
  #define enable_y() ;
  #define disable_y() ;
#endif

#if defined(Z_ENABLE_PIN) && Z_ENABLE_PIN > -1
  #ifdef Z_DUAL_STEPPER_DRIVERS
    #define  enable_z() { WRITE(Z_ENABLE_PIN, Z_ENABLE_ON); WRITE(Z2_ENABLE_PIN, Z_ENABLE_ON); }
    #define disable_z() { WRITE(Z_ENABLE_PIN,!Z_ENABLE_ON); WRITE(Z2_ENABLE_PIN,!Z_ENABLE_ON); axis_known_position[Z_AXIS] = false; }
  #else
    #define  enable_z() WRITE(Z_ENABLE_PIN, Z_ENABLE_ON)
    #define disable_z() { WRITE(Z_ENABLE_PIN,!Z_ENABLE_ON); axis_known_position[Z_AXIS] = false; }
  #endif
#else
  #define enable_z() ;
  #define disable_z() ;
#endif

#if defined(E0_ENABLE_PIN) && (E0_ENABLE_PIN > -1)
  #define enable_e0() WRITE(E0_ENABLE_PIN, E_ENABLE_ON)
  #define disable_e0() WRITE(E0_ENABLE_PIN,!E_ENABLE_ON)
#else
  #define enable_e0()  /* nothing */
  #define disable_e0() /* nothing */
#endif

#if (EXTRUDERS > 1) && defined(E1_ENABLE_PIN) && (E1_ENABLE_PIN > -1)
  #define enable_e1() WRITE(E1_ENABLE_PIN, E_ENABLE_ON)
  #define disable_e1() WRITE(E1_ENABLE_PIN,!E_ENABLE_ON)
#else
  #define enable_e1()  /* nothing */
  #define disable_e1() /* nothing */
#endif

#if (EXTRUDERS > 2) && defined(E2_ENABLE_PIN) && (E2_ENABLE_PIN > -1)
  #define enable_e2() WRITE(E2_ENABLE_PIN, E_ENABLE_ON)
  #define disable_e2() WRITE(E2_ENABLE_PIN,!E_ENABLE_ON)
```

Figure 38

The setup for OctoPi was based on documentation from the OctoPrint website; I went ahead and set up a bot for OctoPi which would send me a text when a print would finish or if there would be an error during the print (*Figure 41*). The aim of this would be to integrate it with the code of the robotic arm in order to automatically start the next print and alert the factory team when a print is finished, potentially allowing them to carry out remote quality control.



Resumed printing ce3_1m~2.gco at 3% %. 06:17:45 remaining. 12:09

/status 12:12

Printing at Z=0.0.
Bed 51.82/50.0, Extruder 199.91/200.0.
00:14:35, 3%% done, 06:06:07 remaining.
Completed time 18:18:08. 12:12

12:12

Figure 41

Testing stage

Isolated component testing

To ensure all motors were correctly working, I tested them before attaching them to the mechanical structure first. To do this I sent GCode commands related to the individual axis the motors were connected to (*Figure 42*).

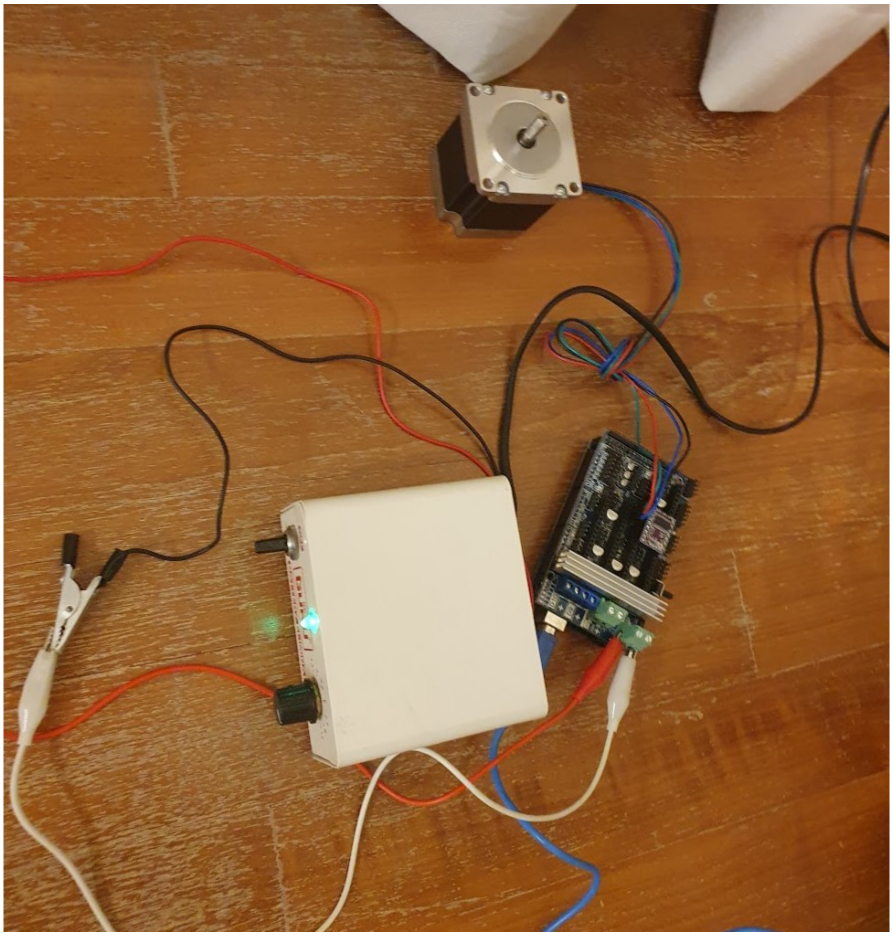
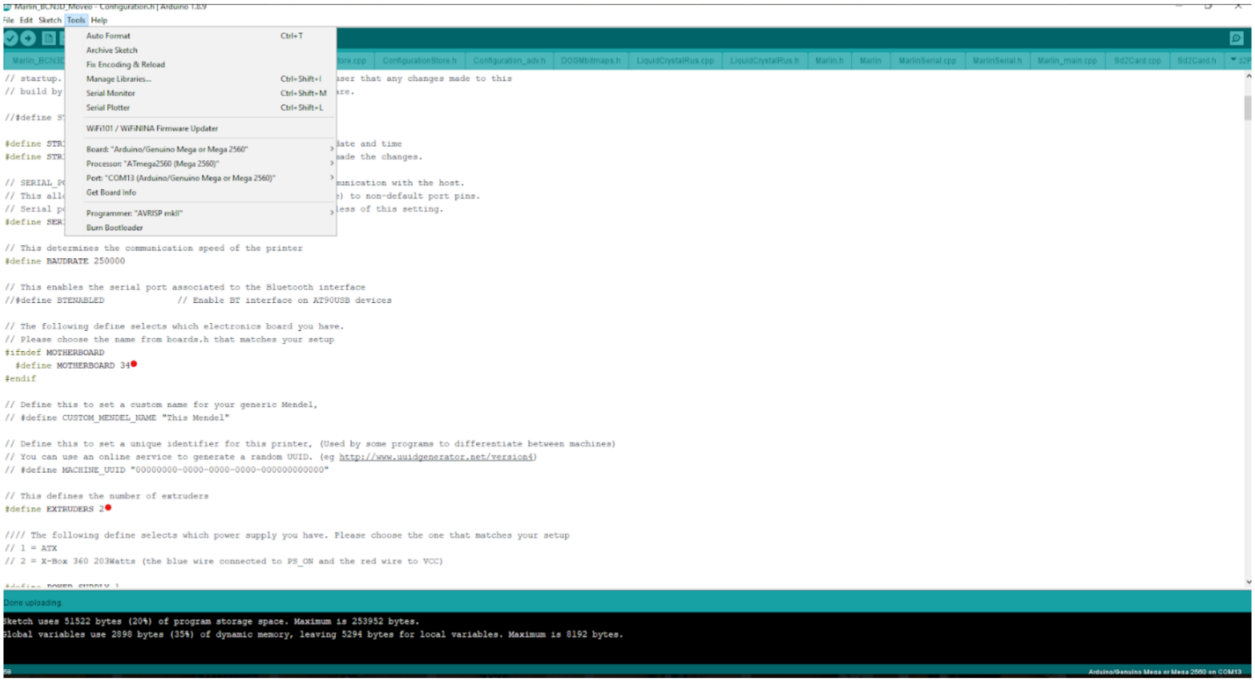


Figure 42

By doing this not only did I get familiar with the sensitivity of each motor, but I was also able to finetune the motor drivers. This means adjusting the current output for a given motor, if the current is too high or too low, the motor will not rotate, but just vibrate and eventually burn out. Only after this process did I realise that this is why a motor had burnt out initially.

When I was testing each axis' motor, I realised that no matter which motor I connected to E1, it would not rotate. After searching for hours, I figured it was due to a set up command in the firmware, as shown on the lines with the red dots (*Figure 43*) (Sarf2k4, 2020). After changing the number of configured extruders in the setup file, all the motors worked perfectly.



```
// startup.                                         ser that any changes made to this
// build by                                         re.

//#define S

#define STR                                         date and time
#define STR                                         made the changes.

// SERIAL_P                                         munication with the host.
// This all                                         ) to non-default port pins.
// Serial p                                         less of this setting.
#define SER

// This determines the communication speed of the printer
#define BAUDRATE 250000

// This enables the serial port associated to the Bluetooth interface
//#define BTENABLED              // Enable BT interface on AT90USB devices

// The following define selects which electronics board you have.
// Please choose the name from boards.h that matches your setup
#ifndef MOTHERBOARD
  #define MOTHERBOARD 34
#endif

// Define this to set a custom name for your generic Mendel,
// #define CUSTOM_MENDEL_NAME "This Mendel"

// Define this to set a unique identifier for this printer, (Used by some programs to differentiate between machines)
// You can use an online service to generate a random UUID. (eg http://www.uuidgenerator.net/version4)
// #define MACHINE_UUID "00000000-0000-0000-0000-000000000000"

// This defines the number of extruders
#define EXTRUDERS 2

//// The following define selects which power supply you have. Please choose the one that matches your setup
// 1 = ATX
// 2 = X-Box 360 203Watts (the blue wire connected to PS_ON and the red wire to VCC)
```

```
Done uploading.
Sketch uses 51522 bytes (20%) of program storage space. Maximum is 253952 bytes.
Global variables use 2898 bytes (35%) of dynamic memory, leaving 5294 bytes for local variables. Maximum is 8192 bytes.
```

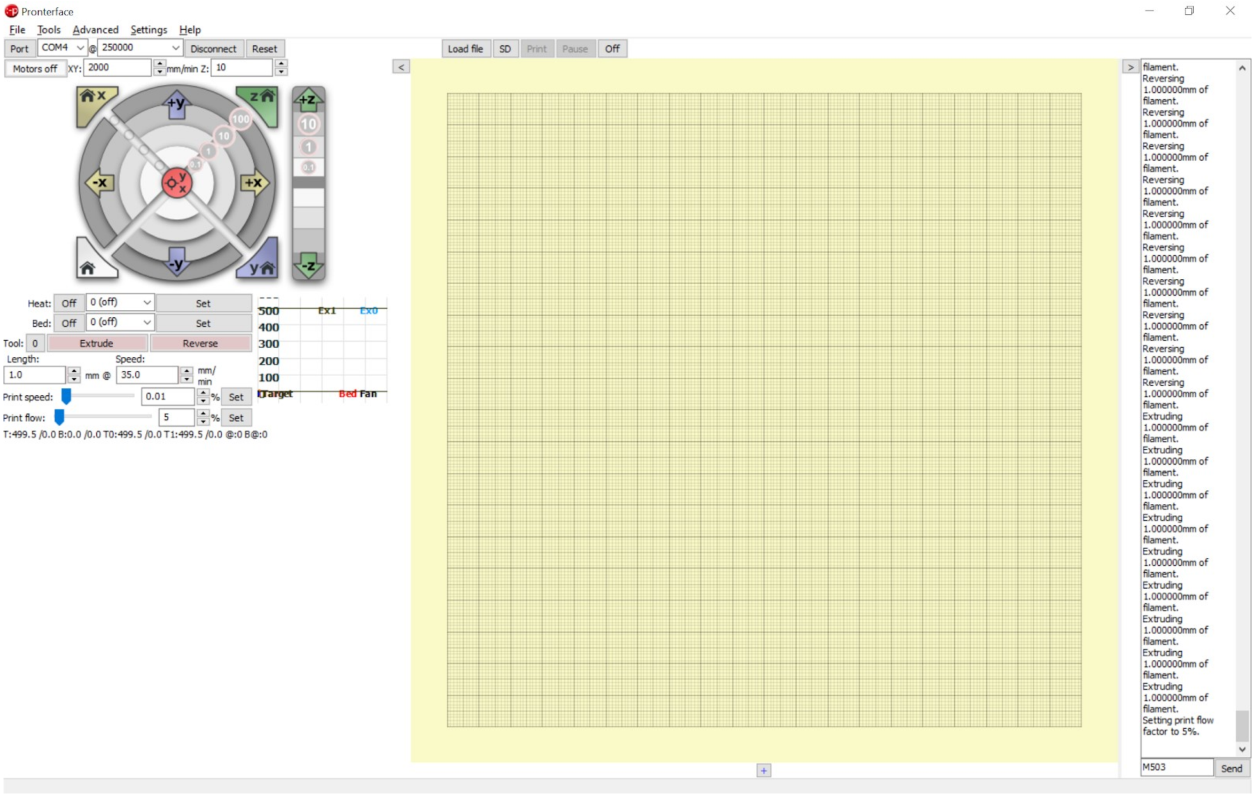
Figure 43

The final individual component I tested was the electromagnet for the picking mechanism. I ordered an electromagnet which had a 10kg holding weight rating, although when I connected it to the Arduino, it could barely hold a paper clip. I learnt that electromagnets can usually hold only 5-10% of their rating (Adafruit, 2019). This meant that if I were to use an electromagnet to lift the print bed, I would require one which can hold 100kg. This is not viable as the magnet is the end effector, and a 100kg holding weight electromagnet would weigh at least 15kg, not feasible for the robotic arm build. This is what led me to explore other alternatives which have still not been finalized. This is definitely something I did not expect and may hinder the ability for the robotic arm to do a full demo of lifting a printer bed, although I am still testing alternatives such as a forklift mechanism.

System testing

With all individual components working, I went on to test the electronic and electromechanical system when fully integrated. The first problem I faced was that when controlling the shoulder axis, it would never move despite the audible moving of the motors.
After tracing the wiring back, I realised this was because the motors were meant to rotate in opposite directions and hence had to be connected in mirror image. The other problem I faced was that the pulley belts were not tight enough, causing teeth to skip when rotating. To overcome this, I made use of the screw tighteners.

In order to control the axis, I used a software called Pronterface. This is a graphical user interface which converts buttons to GCode commands (*Figure 44*).



Port: COM4 @ 250000
XY: 2000 mm/min Z: 10
Heat: Off 0 (off)
Bed: Off 0 (off)
Tool: 0
Length: 1.0 mm @ Speed: 35.0 mm/min
Print speed: 0.01 %
Print flow: 5 %

```
T:499.5 /0.0 B:0.0 /0.0 T0:499.5 /0.0 T1:499.5 /0.0 @:0 B@:0
```

```
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Reversing
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Extruding
1.000000mm of
filament.
Setting print flow
factor to 5%.
```

```
M503
```

Figure 44

The commands in Pronterface are relative commands, meaning the arm would move 5 mm from the current position rather than to a position defined as 5. This difference between absolute and relative positioning is a concept I learnt during the testing of the robotic arm.

With all the mechanical and electronic systems working, I was able to control the robotic arm to go from position A to position B. Appendix link V shows the aforedescribed. The movement of the arm was much more jagged than I expected, and the controlling of the arm is much harder than I thought, although I think those arose due to both software and hardware shortfalls. For example, the gaps between the couplings in the prints means that there is backlash in certain directions, reducing the accuracy and precision of the arm.

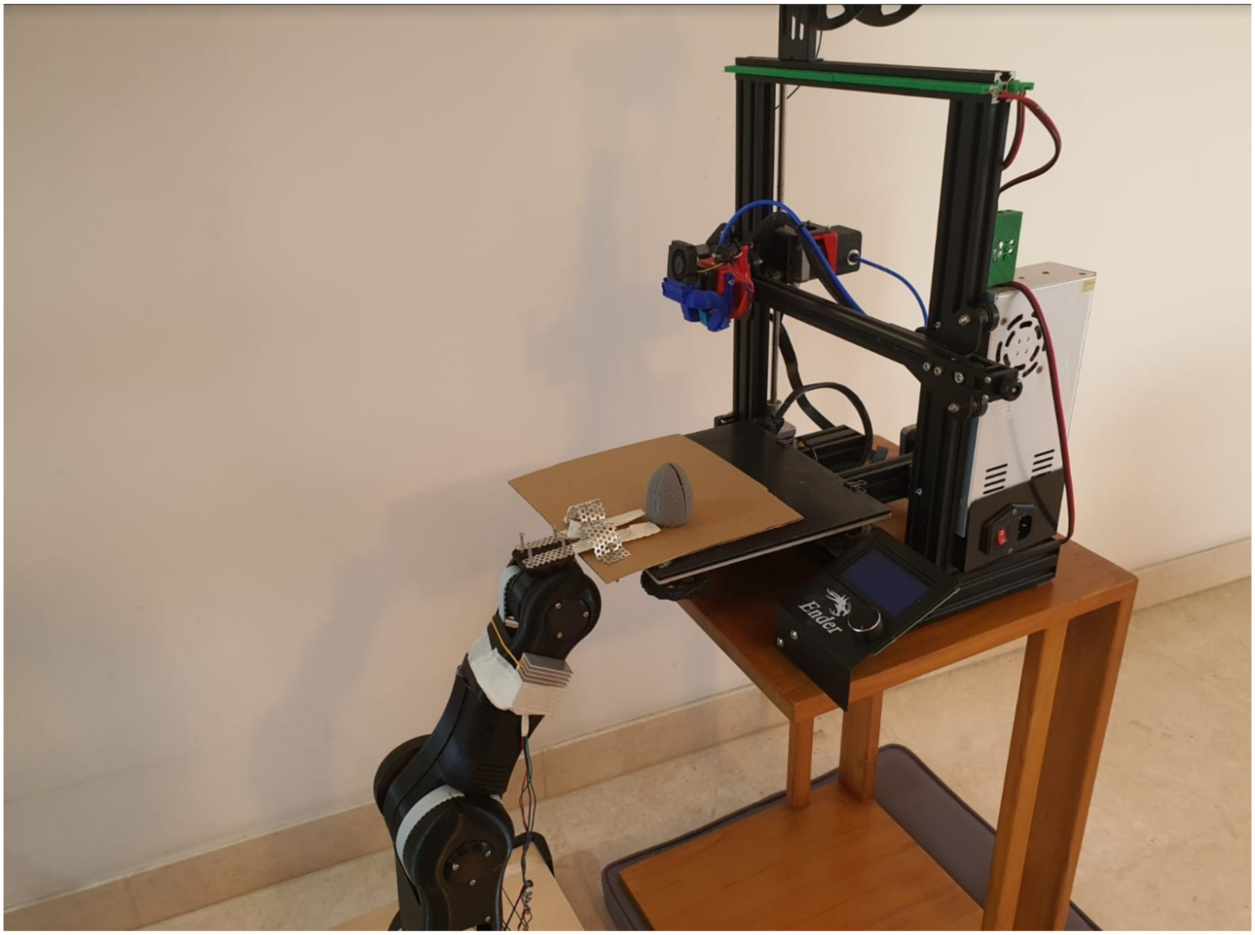


Figure 45

Project summary

Key learnings and conclusion

With the robotic arm working, and able to go between defined positions, I have definitely learnt numerous technical skills, but also invaluable project skills including research and documentation.

I had never done a project of this scale before, and despite the numerous challenges and setbacks faced, I was able to work through each one of them logically to bring this project one step closer to completion. Whilst the arm does perform the minimum expected function, the accuracy and smoothness of operation is something that I can definitely improve on. I would also like to test the arm in a real microfactory setting when I finalise the design for the end effector, since that would prove and validate my hypothesis for this project: using automation retrofits to improve the productive capacity, return on capital, and safety of microfactories.

> *“This automation retrofit will reduce costs and increase efficiency and safety of the 3D printing microfactories: a pivotal part of the supply chains of tomorrow”*

Hakon Ellekjear summarised the project in the above statement when I demonstrated it to him. Ellekjear is an engineer from NTNU and currently is head of a 3D printing venture in the maritime space. He has spoken at multiple global supply chain and innovation events and is well regarded in the industry. Since his quote was directly related to my EPQ project it is extremely purposeful and relevant.

The appendix contains links to various documentation of the build process and testing of the robotic arm.

Bibliography

All3dp, 2017. Updated: Blackbelt 3D Printer “Changing Paradigms” of FDM Technology. [online] All3DP. Available at: <https://all3dp.com/blackbelt-3d/> [Accessed 2 Oct. 2020].

Anon 2019. What is Degree of Freedom (DOF) in Mechanics. [online] SMLease Design. Available at: <https://www.smlease.com/entries/mechanism/what-is-degree-of-freedom-dof-in-mechanics/> [Accessed 4 Oct. 2020].

Anon 2020. 10 Most Efficient Ways to Cut Stainless Steel. [online] Available at: <https://makerslegacy.com/cutting-metal/how-to-cut-stainless-steel/> [Accessed 5 Sep. 2020].

Barnes, J., Forsmark, J. and Mutchler, E., 2018. Process Steps in the Metal Additive Manufacturing Workflow. [online] Digital Alloys. Available at: <https://www.digitalalloys.com/blog/process-steps-metal-additive-manufacturing-workflow/> [Accessed 20 Sep. 2020].

BCN3D MOVEO, 2019. BCN3D MOVEO - A fully Open Source 3D printed robot arm. [online] BCN3D Technologies. Available at: <https://www.bcn3d.com/bcn3d-moveo-the-future-of-learning/> [Accessed 4 Jan. 2020].

Blackbelt 3D, 2020. The Printer - BlackBelt The Printer. [online] BlackBelt. Available at: <https://blackbelt-3d.com/the-blackbelt-3d-printer/> [Accessed 2 Oct. 2020].

Descourvières, E. and Debricon, S., 2007. Towards automatic control for microfactories. [online] Research Gate. Available at: <https://www.researchgate.net/publication/29651035_Towards_automatic_control_for_microfactories> [Accessed 25 Aug. 2020].

Fawcett, J. and Burdess, J., 1988. Basic mechanics with engineering applications. Arnold.

Flynt, J., 2018. Water Soluble Filament: Properties, How to Use, and Best Brands. [online] 3D Insider. Available at: <https://3dinsider.com/water-soluble-filament/> [Accessed 20 Sep. 2020].

FutureBridge, 2020. Microfactories – The Next Big Thing in Manufacturing - FutureBridge. [online] FutureBridge. Available at: <https://www.futurebridge.com/uncategorized/microfactories-the-next-big-thing-in-manufacturing/> [Accessed 6 Oct. 2020].

Gill, H., 2018. stepper motor or servo motor which should it be? [online] Kollmorgen.com. Available at: <https://www.kollmorgen.com/en-us/service-and-support/knowledge-center/white-papers/stepper-motor-or-servo-motor-which-should-it-be/> [Accessed 16 Sep. 2020].

Häußge, G., 2020. Download & Setup OctoPrint. [online] OctoPrint.org. Available at: <https://octoprint.org/download/> [Accessed 25 Jul. 2020].

Instructables, 2017. Build a Giant 3D Printed Robot Arm. [online] Instructables. Available at: <https://www.instructables.com/Build-a-Giant-3D-Printed-Robot-Arm/> [Accessed 4 Jan. 2020].

Malhotra, N., 2020. Project Portfolio. [online] Navyansh Malhotra. Available at: <https://www.navyansh.com/portfolio> [Accessed 5 Oct. 2020].

Molitch-Hou, M., 2018. Automated 3D Printing: How Industrial Additive Manufacturing Is Evolving. [online] Engineering.com. Available at: <https://www.engineering.com/3DPrinting/3DPrintingArticles/ArticleID/16626/Automated-3D-Printing-How-Industrial-Additive-Manufacturing-Is-Evolving.aspx> [Accessed 20 Sep. 2020].

PBR Staff, 2017. Case Study: Voodoo Manufacturing Triples 3D Printing Production With Cobots. [online] Robotics Business Review. Available at: <https://www.roboticsbusinessreview.com/manufacturing/case-study-voodoo-manufacturing-triples-3d-printing-production-cobots/> [Accessed 20 Aug. 2020].

Sarf 2k4, 2020. Configuring Marlin. [online] Marlin Firmware. Available at: <https://marlinfw.org/docs/configuration/configuration.html> [Accessed 25 Sep. 2020].

Stratasys, 2017. Stratasys Continuous Build 3D Demonstrator - YouTube. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=uIy6rgwuIT0&feature=emb_logo&ab_channel=Stratasys> [Accessed 4 Feb. 2020].

Stratasys, 2019. 6DoF mostly 3D Printed Robot Arm (Part 3) - YouTube. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=12Be3Hoh-sY&ab_channel=Skyentific> [Accessed 3 Jan. 2020].

Thorogood, D., 2015. ROBOT DYNAMICS. MOTORS supply the FORCE that the robot needs to move Rotational Force is called TORQUE The motor needs to supply force to wheels arms. - ppt download. [online] slideplayer.com. Available at: <https://slideplayer.com/slide/4219624/> [Accessed 25 Aug. 2020].

VICE News, 2020. 3D Printing Is Changing the World - YouTube. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=GV8zPtqOyqg&ab_channel=VICENews> [Accessed 12 Feb. 2020].

Zhang, Y., 2020. Chapter 4. Basic Kinematics of Constrained Rigid Bodies. [online] Cmu.edu. Available at: <https://www.cs.cmu.edu/~rapidproto/mechanisms/chpt4.html> [Accessed 5 Jun. 2020].

Appendix

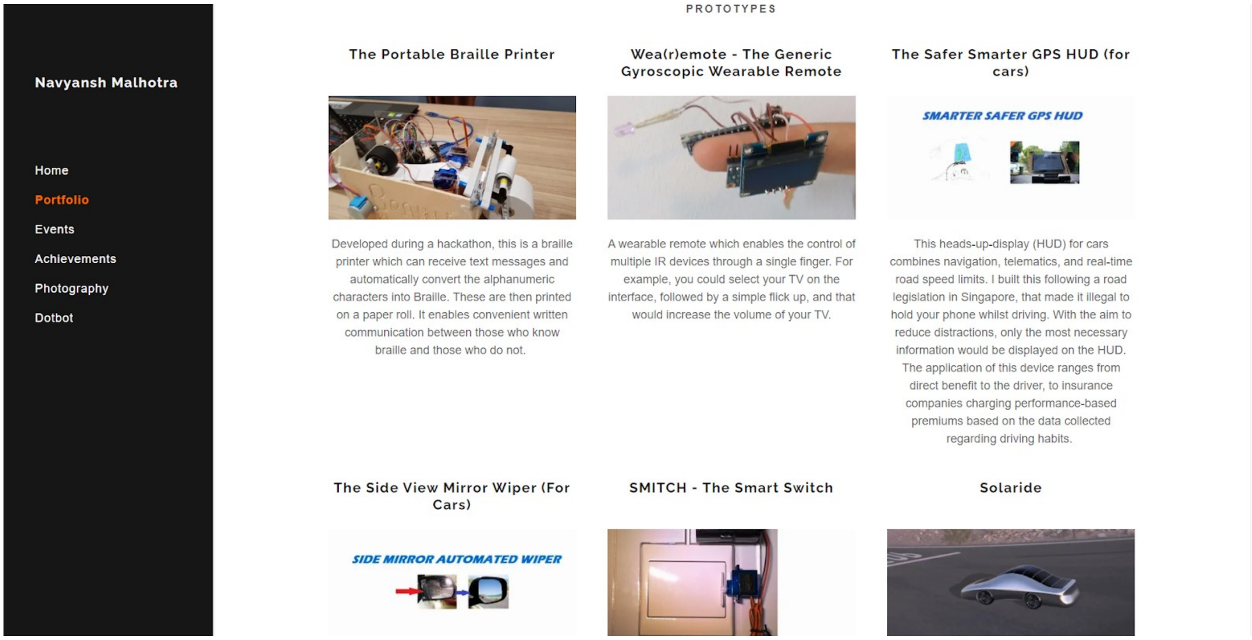
I. Project Portfolio - www.navyansh.com

Navyansh Malhotra

- Home
- Portfolio
- Events
- Achievements
- Photography
- Dotbot

Navyansh Malhotra

Maker, Inventor and Entrepreneur
Co-Founder of dotbot (US Patent Pending)



Navyansh Malhotra

- Home
- Portfolio
- Events
- Achievements
- Photography
- Dotbot

PROTOTYPES

The Portable Braille Printer

Developed during a hackathon, this is a braille printer which can receive text messages and automatically convert the alphanumeric characters into Braille. These are then printed on a paper roll. It enables convenient written communication between those who know braille and those who do not.

Wea(r)emote - The Generic Gyroscopic Wearable Remote

A wearable remote which enables the control of multiple IR devices through a single finger. For example, you could select your TV on the interface, followed by a simple flick up, and that would increase the volume of your TV.

The Safer Smarter GPS HUD (for cars)

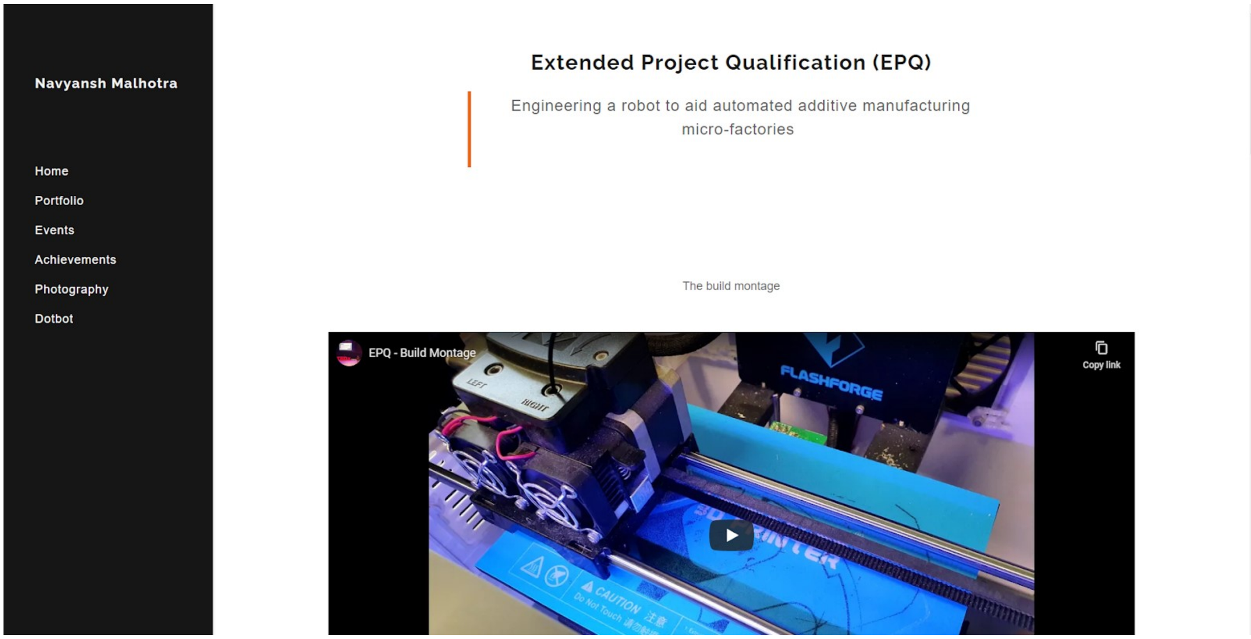
This heads-up-display (HUD) for cars combines navigation, telematics, and real-time road speed limits. I built this following a road legislation in Singapore, that made it illegal to hold your phone whilst driving. With the aim to reduce distractions, only the most necessary information would be displayed on the HUD. The application of this device ranges from direct benefit to the driver, to insurance companies charging performance-based premiums based on the data collected regarding driving habits.

The Side View Mirror Wiper (For Cars)

SMITCH - The Smart Switch

Solaride

II. EPQ website - www.navyansh.com/epq



Navyansh Malhotra

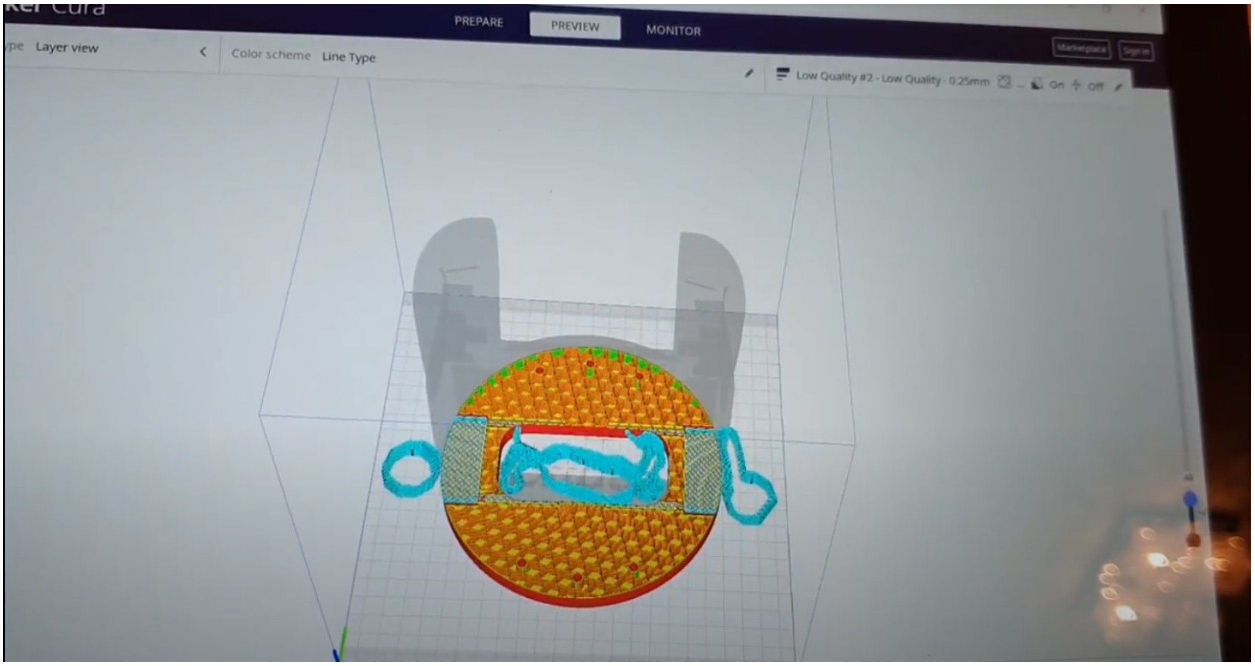
Extended Project Qualification (EPQ)

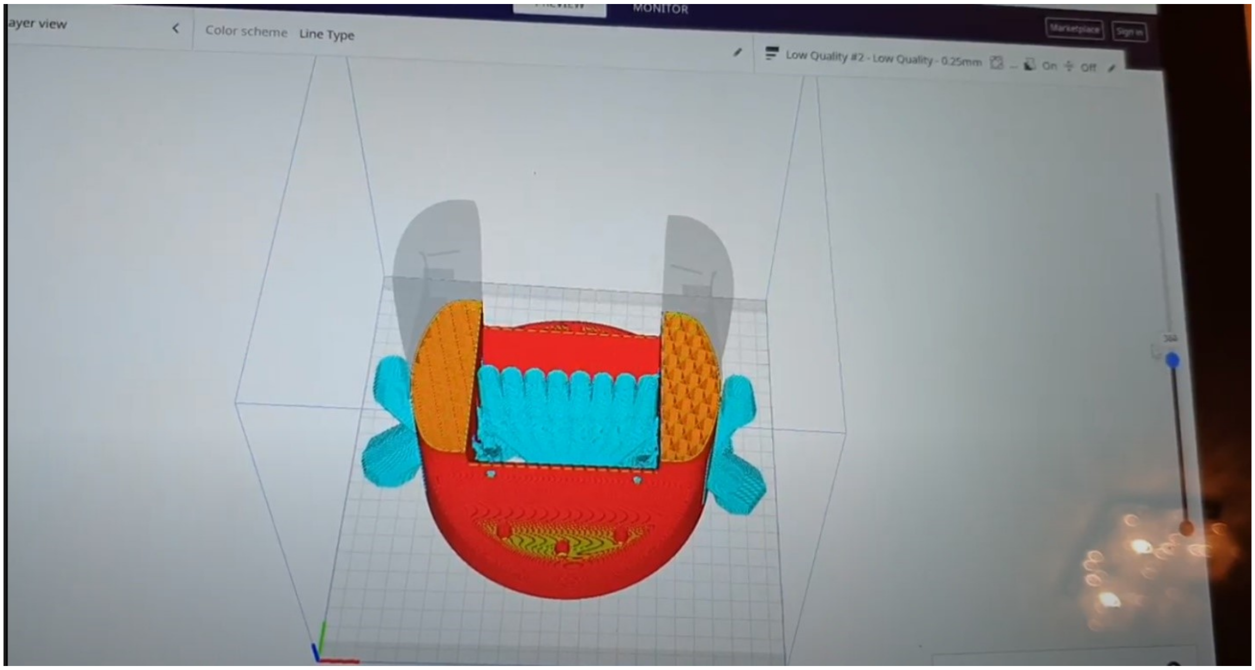
Engineering a robot to aid automated additive manufacturing micro-factories

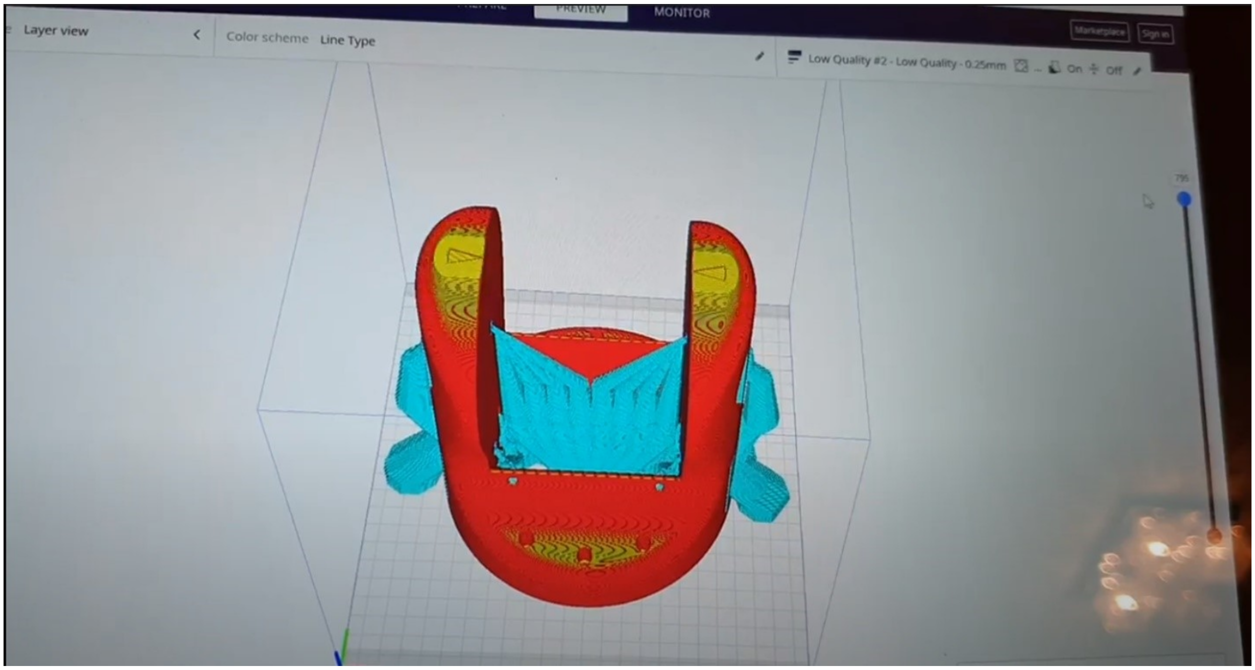
The build montage

EPQ - Build Montage

III. CAD model made up of layers - https://cutt.ly/XhWGjD6







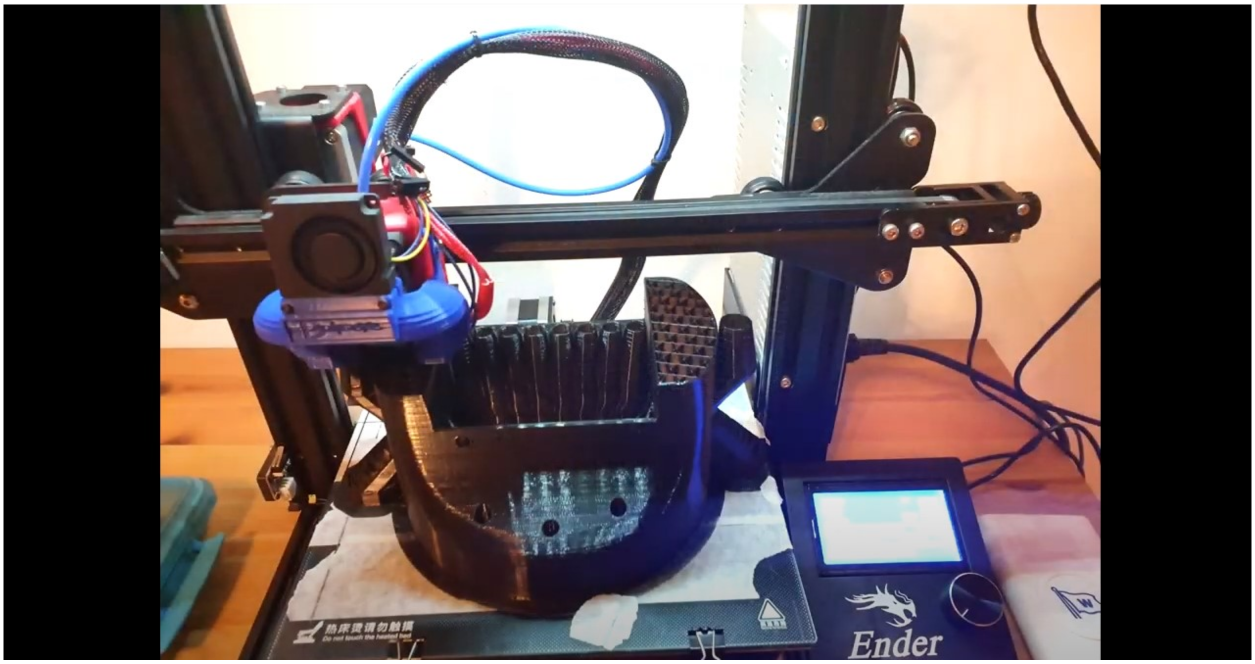
Color scheme Line Type

Low Quality #2 - Low Quality - 0.25mm

On

Off

IV. Build montage - https://youtu.be/BSCzHgXCamc



V. Robotic arm picking up a printer bed - https://youtu.be/DRXlUrvvbiI