

3D Printer Robotic Bed Removal

A Major Qualifying Project



WPI

Written By:

Wyatt Bahm (RBE)

Oliver Jay (RBE)

Tom Vagnini (RBE)

Max Westwater (RBE/CS)

Presented to:

Professor Nicholas Bertozzi (RBE)

Professor Brad Miller (RBE)

Professor Craig Putnam (RBE/CS)

Submitted in partial fulfillment of the
requirements for the Degree in Bachelor of
Science

August 2019-May 2020

Table of Contents

Table of Figures.....	3
Table of Tables.....	4
1.0 Introduction.....	5
2.0 Background	6
2.1 Part Removal	6
2.2 Queueing and Printer Management Software	7
2.3 3D Printers for Schools & Makerspaces	8
3.0 Methodology.....	9
3.1 Build Plate Removal.....	9
3.1.1 Considerations and Strategy.....	9
3.1.2 Selected Design.....	12
3.2 Part Removal	15
3.2.1 Plate Flexing.....	15
3.3 Electrical System	21
3.4 Queueing and Printer Control Software	24
3.4.1 Client Technologies	25
3.4.2 Server Technologies.....	25
3.4.3 Client and Server Communications	27
3.4.4 User Interface.....	27
3.4.5 User Workflow.....	28
3.4.6 Online Slicing.....	28
3.4.7 Part Visualization	29
3.4.8 Vision Processing	29
4.0 Results.....	29
4.1 Part Removal Results.....	29
4.2 Machine Performance	31
4.3 Website Features.....	33
4.3.1 User Authentication	33
4.3.2 Parts Management.....	34
4.3.2 Queueing and Printer Management	36
4.4 Printer Client Software	37
5.0 Discussion.....	38
5.1 The Implications of Global Pandemic.....	38
5.1.1 Utilization of the project due to COVID.....	39

5.2 Possible Future Work	39
5.2.1 Part Storage.....	39
5.2.1 Storage Constraints.....	39
5.2.2 Considered Storage Solutions	40
5.2.3 Possible Improvements to Brush Design.....	40
5.2.4 Software Improvements for Large Print Farms	41
6.0 Conclusion	41
7.0 Works Cited	42
8.0 Appendices	43
8.1 Bill of Materials	43
8.2 Project Assembly	46
8.2 Dual MC33926 Motor Driver Carrier Pin Descriptions	50

Table of Figures

Figure 1: MakerBot Thing-O-Matic (left) and Blackbelt (right)	6
Figure 2: The Multiprinter 2 During Assembly	7
Figure 3: The Multiprinter 2	7
Figure 4: Octoprint Web Interface	8
Figure 5: Build Plate Lifting Points Seen From Below	10
Figure 6: Blocks Permanently Attached to Build Plate	11
Figure 7: Anti-Catapult System	12
Figure 8: Pivot Rotation Utilizing Y axis Motion and Plate Tabs.....	13
Figure 9: Motor Speed Selection Calculations.....	13
Figure 10: Calculation of Usable Torque from Selected Motor.....	14
Figure 11: Plate Removal Prototype.....	14
Figure 12: Plate Removal Forces	15
Figure 13: Plate Bending Prototype.....	15
Figure 14: Early Plate Bending Mechanism and Testing	16
Figure 15: Finger Forcing Plate into Vertical Bend	17
Figure 16: Anti-Catapult.....	18
Figure 17: Anti-Catapult Wedge Separating the Build Plate from the Heated Bed	18
Figure 18: Part Removal Brush Assembly	19
Figure 19: Brush Hard Stop, Limit Switch, and Belt Tensioning	19
Figure 20: Target Speed Calculations	20
Figure 21: Brush Force Calculations	20
Figure 22: Brush Assembly Forces	21
Figure 23: System Schematic.....	22
Figure 24: Prototyped Wiring Setup.....	23
Figure 25: Final Electronics Board	23
Figure 26: Software System Overview.....	24
Figure 27: AWS Cloud Structure	26
Figure 28: User Software Workflow	28
Figure 29: Raw and Filtered Image of a Messy Build Plate	29
Figure 30: Face Shield Visor Test	30
Figure 31: 10mm Cube Test.....	30
Figure 32: Pivot Position and Power Over Time	32
Figure 33: User Signup UI	33
Figure 34: Example Email for Password Reset.....	34
Figure 35: Part Management UI	35
Figure 36: Example of Deleting a Part	35
Figure 37: Example File Preview	36
Figure 38: Printer Queueing Interface.....	36
Figure 39: Printer Management Interface.....	37

Figure 40: Printers Setup for Printing Face Shields	39
Figure 41: Storage System Design	40
Figure 42: Attachment to Prusa i3 MK3	46
Figure 43: Full Part Removal System Exploded by Subassembly	47
Figure 44: Attachments to Main Pivot Exploded.....	47
Figure 45: Brush Mount Exploded	48
Figure 46: Bed Locking Servo Mount Exploded	48
Figure 47: Bed Rotation Motor Mounting Exploded	49
Figure 48: Brush Control Gearing Exploded	49

Table of Tables

Table 1: Arduino Pin Mappings.....	24
Table 2: Flat Disk Removal Results	31

1.0 Introduction

As 3D printers have become more widely available and affordable, additive manufacturing has become very popular among makers, schools, and shared workshops like makerspaces. 3D printing allows for a totally new way of solving engineering problems and completing DIY projects, but there are still many limitations for users outside of a dedicated manufacturing environment. Hobbyist grade 3D printing is still very slow compared to techniques like machining and laser cutting, and once one part finishes, the printer cannot operate until a user manually clears the build plate and starts the next part. This makes it difficult to print a large number of parts consecutively because someone must be present each time a part finishes. Especially in schools and makerspaces that are closed at night, this leads to a lot of printer downtime and long queues. This project aims to automate this process and increase the throughput of a desktop 3D printer by adding an automatic build plate changer and part removal system. The system is composed of two main parts, the hardware and the software. The hardware includes mechanisms to lift the build plate, remove parts, and store them. The software includes the website and software stack required to manage users' print jobs and queue them onto printers.

The mechanical system utilizes a Prusa i3 MK3 3D printer as it is a reliable, popular, and inexpensive 3D printer. Our device autonomously detaches the build plate from the printer, manipulates the plate to remove the printed parts, and then readies itself for the next print job. This system is designed to operate in such a way that the print quality is not negatively impacted by the plate or part removal process. The mechanical system was designed to have a small footprint so that this project could be feasibly implemented in schools or makerspaces without having to change the way they have their printers setup. Because the system was designed to run unattended, simplicity and reliability were of high importance in system design and part selection.

The software system is made up of two main parts, the printer control software and the web-based queueing and management system. The printer control software runs on a raspberry pi and serves to connect a standard 3D printer to our online system. The web-based management system will allow users to control their printers remotely. Users can upload parts to a cloud storage system, slice them, and then add them to a queue of prints for the system to run.

These components integrate together to form a system that is easy to use and allows for increased throughput. This project increases the usefulness of a printer by its ability to run over nights and weekends, as well as coordinate jobs from many different users uploaded to one website. The system also simplifies the organization of parts since the state of different users' jobs is recorded and can be displayed to the operator.

2.0 Background

2.1 Part Removal

The main barrier to truly autonomous printing is removing printed parts from the build plate, which has been attempted by a number of companies without much success. The first attempt to create a print remover was made by Makerbot in 2010 with their Thing-O-Matic 3D printer with the Automated Build Platform, shown on the left in Figure 1 [1]. This is a cartesian 3D printer with a special belt that rotates the part off of the build plate at the end of the print. This system had several problems like warping from the belt, in addition to the lack of software to queue parts. Many hobbyists have turned to removing parts by ramming their print head into the part, although this doesn't work well with the majority of 3D printer parts and can damage the printer if the part does not dislodge easily. In more recent years, the Blackbelt 3D printer, seen on the right in Figure 1 [2], and the Printrbelt have been better automation solutions, although neither has been without drawbacks. The x-axis on those printers is set at an angle, so it is a very unconventional 3D printer and presents challenges in part design and slicing compared to the traditional workflow in a cartesian or delta machine. As a result of high price points, poor reliability, or decreased utility of the printer itself, none of these available options are a good choice for environments like schools or makerspaces.

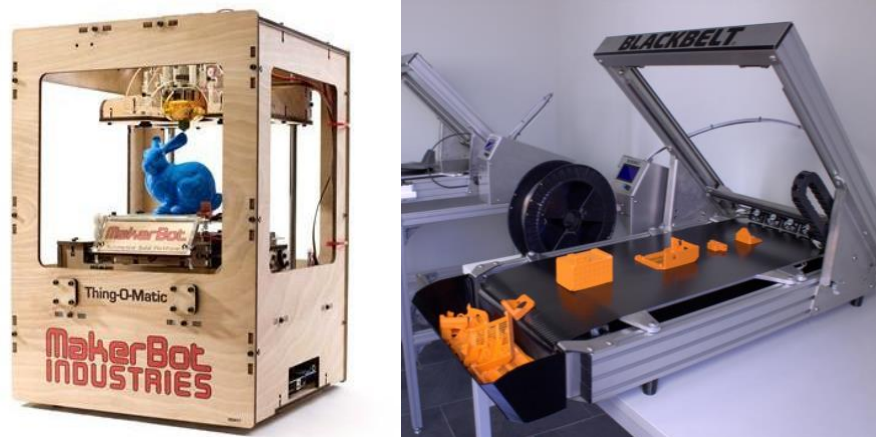


Figure 1: MakerBot Thing-O-Matic (left) and Blackbelt (right)

This project was conceived as an MQP as the result of one group member, Thomas Vagnini's previous work on this problem. He developed a system to load empty build plates into a 3D printer and remove the full build plates with parts still attached, before putting them onto a storage shelf. This was a good approach to the problem, and a somewhat innovative solution, but the storage of full plates with parts still attached created constraints in part size and the number of parts that could be stored. This system also lacked a queuing system and relied on the user manually setting up the sequence of parts.

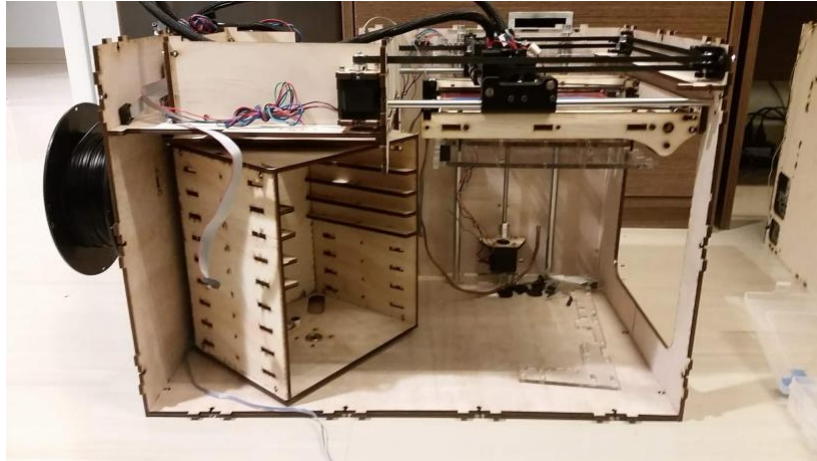


Figure 2: The Multiprinter 2 During Assembly

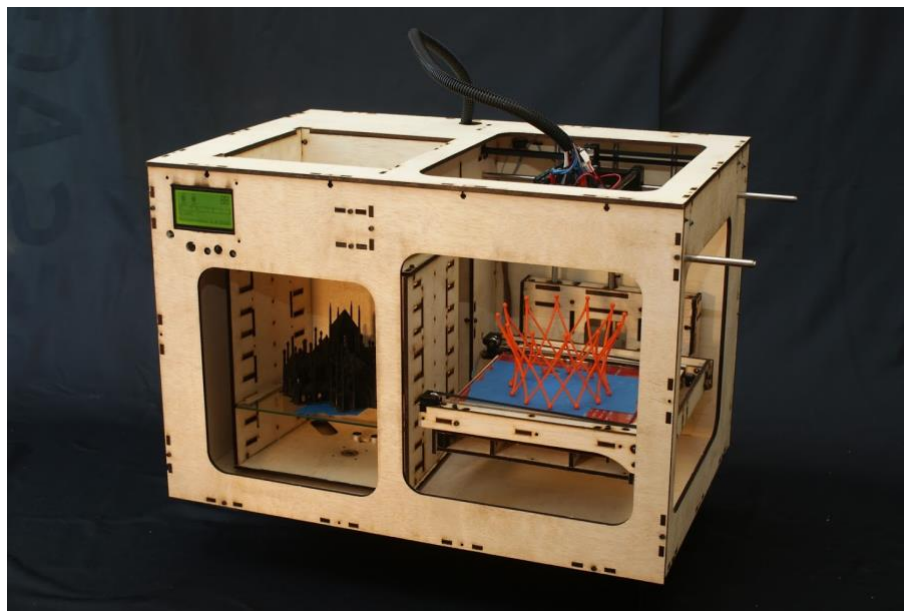


Figure 3: The Multiprinter 2

2.2 Queueing and Printer Management Software

Various companies and open source projects have created software tools to manage 3D printers. Each has their own specific application, but none of them target this project's specific use case. Some printers, such as the Ultimaker, now come packaged with their own software solution. This service, called Ultimaker Connect, allows for users to upload and slice parts online and manage their fleet of printers. Other solutions are more homebrew, and utilize a Raspberry Pi running open source software to control the printer.

One of these open source projects is called Octoprint. Octoprint runs on a Raspberry Pi and hosts a web server to allow a user on the local network to control their printer. There is a very active

community around support and plugin development, making it an easy option for new users to pick up [3] [4]. The web interface removes the need to manage writing gcode files to an SD card or connect the printer to a desktop computer. Having the printer run on the Raspberry Pi prevents errors associated with the host computer crashing or sleeping due to the user working on their own computer. The plugin repository also provides a wide array of advanced features which can be easily installed to add new functionality. One main drawback of this system is that due to the Raspberry Pi hosting the web server itself, there isn't an ability to connect multiple printers and access the printer from outside of the local network without port forwarding the connected router.

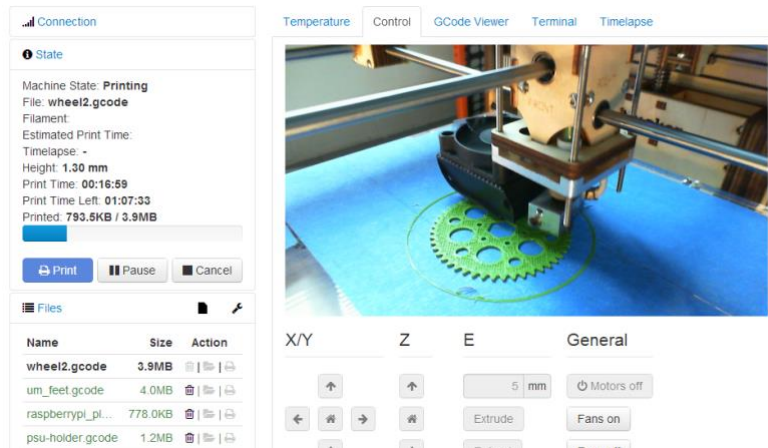


Figure 4: Octoprint Web Interface

Other solutions are cloud based interfaces which can manage multiple printers. Two examples of this are 3DPrinterOS and Astroprint. Both of these systems are a freemium business model with a free to use base tier and a paid subscription to access advanced features, like print queueing. Both of these systems utilize a Raspberry Pi for controlling the printer but, unlike Octoprint, host a majority of the processing in the cloud. This allows for the use of multiple printers grouped into one system. Each company sells a Raspberry Pi as a controller with their software preinstalled. It costs \$249 for 3DPrinterOS [4] and \$125 [5] for Astroprint, although Astroprint is open source so users can install it for themselves if they want. Both solutions provide some level of user authentication and print queue management, but both are paywalled behind the premium subscription.

2.3 3D Printers for Schools & Makerspaces

This project was developed to work with an existing popular 3D printer to make it easy to implement for users with limited technical ability. We selected the Prusa i3 MK3 for this purpose, as it is already one of the most popular and reliable printers on the market. Several aspects of the design make the printer ideal for the purposes of this project. Its removable magnetic build plate, automatic bed leveling, and robust error detection all lend themselves well to a part removal system. The removable magnetic build plate significantly simplifies the process of removing parts by allowing a plate changing mechanism to easily detach and then realign the build plate [6]. Additionally, its spring steel build plate can be flexed to release the parts much easier than with a traditional rigid plate. The printer's error

detection and inductive sensor bed leveling also make continuous operation more reliable, due to more consistent adhesion of the first layer. For each print, the Prusa MK3 homes off of 9 points on the heated bed to level the print for reliable first layer adhesion. This advanced bed leveling can also be increased to 49 points for better results. First layer issues are one of the most common causes of failed prints. While most printers are started with a human operator nearby to catch these issues, our system will not have an operator to easily stop it. Therefore, automatic bed leveling is essential to the success of this project. The Prusa i3 MK3 also includes a physical filament sensor to prevent the printer from continuing to run for an extended period of time if the filament spool runs out.

This project has a number of potential markets in which it could be useful, including schools, small businesses, makerspaces, and hobbyists that wish to run printers overnight or reduce labor costs. While it is common to start a long print overnight, most printer owners do not have a way to continuously run multiple prints without manually clearing the build plate and starting the next print. A system to remove and start prints could extend the operating time of a printer from a standard 40 work hours to 24-7 operation, theoretically improving the output of a printer by a factor of four.

3.0 Methodology

Our project consists of two mechanical systems built around a Prusa MK3 3D printer. The mechanical system consists of a mechanism to remove the build plate and a system to remove the part from the build plate. The software system provides an interface for users to upload and manage parts as well as monitor and manage the mechanical system. This allows the printer to print a continuous stream of parts printed without an operator present.

3.1 Build Plate Removal

3.1.1 Considerations and Strategy

The stock magnetic spring steel build plate on the Prusa allows users to easily detach and reattach the plate before and after part removal. However, there are a few issues with the stock build plate for our purposes. The plate is not designed to be lifted by a machine and only contains two small areas in which we can lift it. Figure 5 shows the heated bed viewed from the bottom, with the build plate on top of it. The top corners are the only accessible lifting points on the build plate.

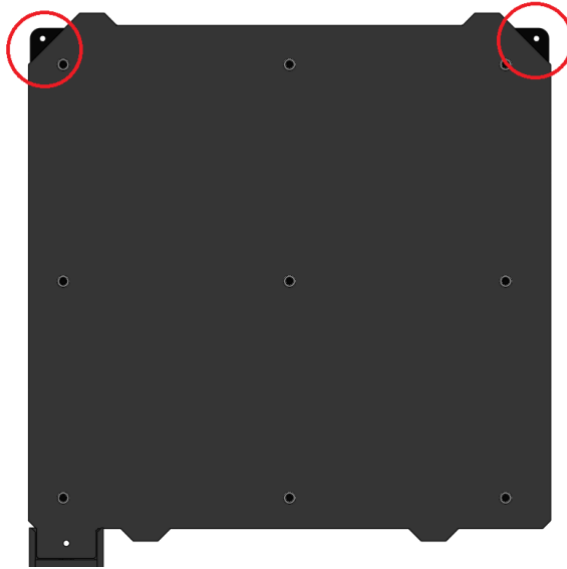


Figure 5: Build Plate Lifting Points Seen From Below

We considered a number of different methods to facilitate this manipulation with the limited contact area. One option was a strong gripper to grab the overhang tabs at the front of the printer. This is most similar to how a person removes the build plate, although it would require a significant gripping force. Slipping or alignment issues are also likely with such an approach.

Early prototypes of a part extending the build plate were promising. This part was designed to permanently attach to the build plate to assist with gripping the plate. These attached tabs are small enough to not impede printer functionality but give the plate removal system a much larger and more reliable connection point. We suggested different options for the tabs to engage with our lifting mechanism. One option used a dovetail joint to attach the plate to the plate removal system, though we decided to explore options due to the complexity of the proposed mechanism. We also considered an attachment that would slide into the receptacle using the y axis of the printer and use a key slot or finger to constrain the build plate, shown in Figure 6. After the shape of this part was refined and adjusted this was our most successful prototype and was carried through to the final design.



Figure 6: Blocks Permanently Attached to Build Plate

We next considered how the lifting mechanism would move once the plate was attached to it. We had a few requirements for this part of the design. The mechanism had to be strong enough to lift the plate off of the magnetic heated bed, and then move the plate into a position where it could be easily bent for part removal. The system had to be reliable, sturdy, and utilize few degrees of freedom.

When an operator manually removes the build plate from the printer, they lift from the front tabs, and the plate disengages from the magnets, starting at the front and going backwards, rotating around the back corner of the print bed. Ideally, we would rotate our lifting mechanism around the back edge of the print bed and easily disengage from the magnets. However, using a simple rotation around the back edge would leave the build plate in a poor position to perform part removal. A four-bar linkage achieves the desired motion but passes through the same space as the print head and increases complexity in the design and controls. It is more difficult to calculate and control the necessary torque and velocity at each position for the four-bar mechanism, especially given the complexity of the magnetic force and bending force from the plate.

We decided to start by prototyping the simplest configuration, which is a single pivot point in front of and coplanar with the build plate. In this configuration, the force component from the torque is directed upwards, and we theorized that it may be able to disengage the plate. If it has sufficient torque, then the pivot rotates the plate upwards to the front of the machine. This gives plenty of room for the part to be removed and then moved into the storage system. We began by prototyping the attachment between the plate and rotating mechanism. In initial testing we were able to rotate the pivot by hand to lift the plate off of the magnets. We were careful to closely approximate the center of rotation for the pivot, and test lifting with that rotation. Initial testing was promising, so we went to work to generate the pivot geometry in Solidworks. We performed motor calculations to choose a suitable motor to control the pivot and generated the mounting geometry in Solidworks as well. Our prototypes designed in Solidworks consisted of 40mm 8020 extrusion, laser cut wooden brackets, 3D printed parts, and purchased components.

3.1.2 Selected Design

After prototyping a few options for the build plate attachment and removal, we settled on a combination of systems that we felt were most likely to work well and integrate together. To easily pick up the build plate, we decided to use simple tabs that are permanently attached to the build plate. These blocks engage in a receptacle on the pivot by sliding along the y axis. This way, we could reuse the y axis of the printer for this mechanism and reduce the degrees of freedom of our design. We planned to use micro servos or solenoids to lock the plate into the rotating mechanism, but this proved unnecessary as the build plate is held in place by gravity and friction once it has been lifted.

We decided to use a single pivot to move the build plate, which is located in front of the printer and coplanar with the heated bed. This configuration allows for the build plate to be easily removed, however it results in a springing motion when the build plate disengages from the magnets. We also designed a thin wiper to slide underneath the build plate and disengage it from the magnets so it does not spring upwards. This system can be seen in Figure 7 and the mechanisms responsible for attaching and rotating the plate are shown in detail in Figure 8.

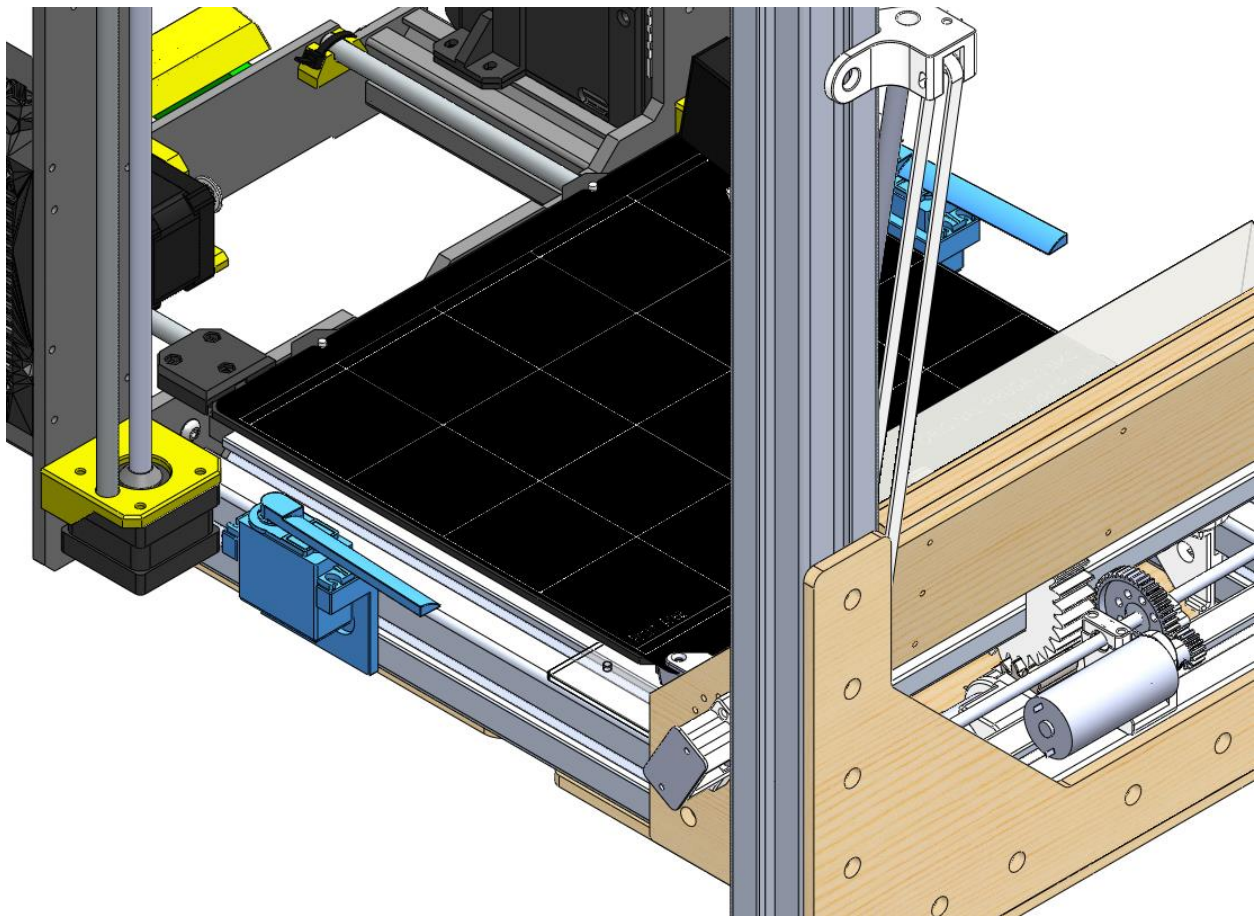


Figure 7: Anti-Catapult System

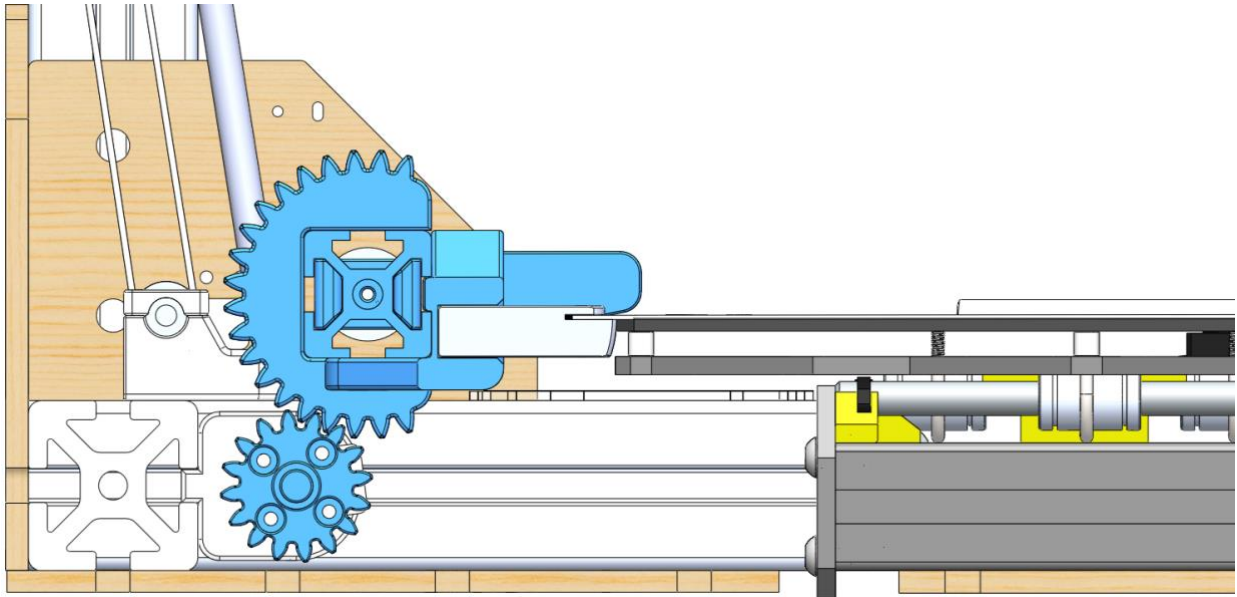


Figure 8: Pivot Rotation Utilizing Y axis Motion and Plate Tabs

To choose our motor for the main pivot, we first calculated the desired output speed based on our desired time to complete the bed lift and estimated the torque we would need with a safe margin for error. We wanted to rotate the build plate from horizontal to vertical (90 degrees) in no less than 2 seconds. Based on our calculations in Figure 9, this resulted in a target running speed target of around 7.5 RPM. However, this target speed is with the system operating with a load, so in order to have our pivot motor operating in the middle of its curve we looked for a no-load speed of around 15 RPM. We prototyped the build plate tabs and pivot to approximate the necessary torque before motorizing it and found that breaking the plate free from the magnetic bed would require at least 3 lbf*ft torque. To allow for sufficient usable torque and design in the middle of the motor curve, we required at least 6lbf*ft of stall torque from our selected motor.

$$\begin{aligned} \text{Time to travel } 90^\circ &= 2 \text{ seconds} \\ \text{Output speed} &= \frac{\frac{1}{4} \text{ rotations}}{2 \text{ seconds} \times \frac{1 \text{ minute}}{60 \text{ seconds}}} = 7.5 \text{ RPM} \\ \text{Motor speed (no load)} &= 19 \text{ RPM} \\ \text{Gear reduction of } 2 : 1 & \\ \text{Output speed (no load)} &= \frac{19 \text{ RPM}}{2} = 9.5 \text{ RPM} \end{aligned}$$

Figure 9: Motor Speed Selection Calculations

We elected not to mount this motor concentric with the pivot bar and designed a gear transition between the two. For this, we set a maximum allowable single stage of gear reduction up to 5:1 due to size constraints in order to minimize footprint. We selected a ServoCity 499:1 Economy motor with a no-load rpm of 19 and a stall torque of 13.9lbf*ft. We designed an additional 2:1 gear reduction into our system to reduce the no load speed to 9.5 RPM and verified that it would meet our torque requirements as can be seen in Figure 10. The motor, 2:1 gear stage, and pivot is shown in Figure 8. These gears

between the motor and pivot were designed in Solidworks to be 3D printed. This allowed us to print the gear attached to the pivot bar in such a way that it attached directly to the 8020 and took up less space. We choose a diametral pitch of 12.7 and a face width of 18mm for these gears. They were also printed with three walls and 25% infill to increase strength. Our operating conditions will not be at either end of the motor curve, so this gear system perfectly fits our torque and speed constraints of slower than 8 rpm and more torque than the 6lbf*ft.

Required Torque at pivot = 3 lbf · ft

Gear reduction of 2 : 1

Required torque from motor must be at least 1.5 lbf · ft

$$T_{\text{stall}} = 2,678.28 \text{ ozf} \cdot \text{in} \times \frac{1 \text{ lbf} \cdot 1 \text{ ft}}{16 \text{ ozf} \cdot 12 \text{ in}} = 13.949 \text{ lbf} \cdot \text{ft}$$

$$T_{\text{stall at pivot}} = 13.949 \text{ lbf} \cdot \text{ft} \times 2 = 27.898 \text{ lbf} \cdot \text{ft} \leftarrow 2 : 1 \text{ gear reduction}$$

$$T_{\text{pivot}} = 13.949 \text{ lbf} \cdot \text{ft} \leftarrow \text{safety factor of 2}$$

Our usable force at the pivot is 13.949 lbf · ft which is over 4.5 times the requirement to lift the plate

Figure 10: Calculation of Usable Torque from Selected Motor

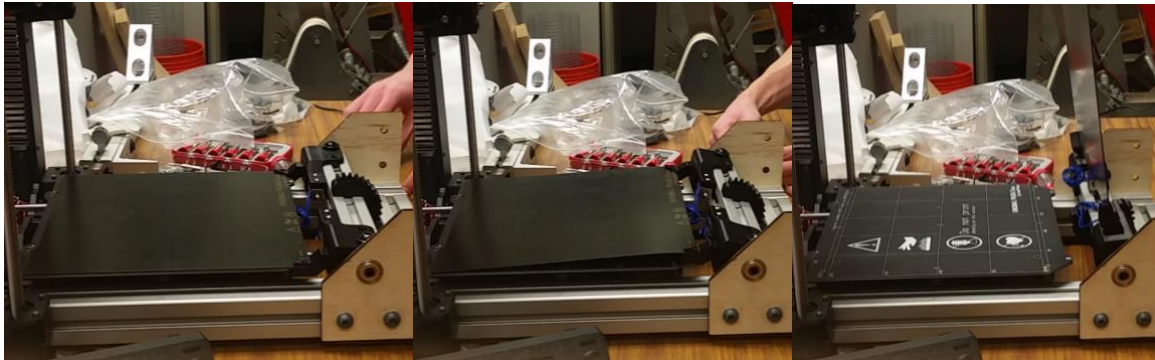


Figure 11: Plate Removal Prototype

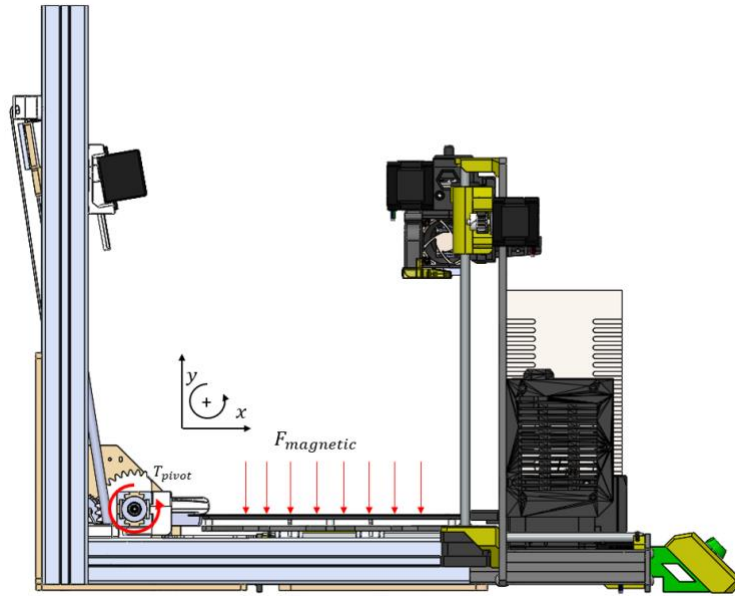


Figure 12: Plate Removal Forces

3.2 Part Removal

3.2.1 Plate Flexing

An initial prototype for plate bend testing was created and can be seen in Figure 13. The build plate sits in slots in the mechanism where to fix it at each of the four corners. Pivoting arms engaged with the at the middle on each side of the plate to flex the it up and down. This prototype tested the viability of removing parts with flexing and was an early part of our decision to move forwards with this method. The results were promising, although we later found that a brush was required to finish the part removal in some cases.



Figure 13: Plate Bending Prototype

As part of our goal to minimize degrees of freedom for this project, we spent a lot of time searching for a plate flexing mechanism that utilized the same actuators as the plate removal system and we made some adjustments for the flexing to be integrated in this way. The main challenge in this process came from the fact that with this pivoting motion, there was not a simple way to flex the plate from the middle without adding in additional actuators. For this type of flexing, the contact point at the middle of the plate needs to be moving in relation to both ends of the plate which must be fixed relative to each other. This means that either the plate needs to be gripped and rotated at the top as well as the bottom, or the whole middle contact point needs to pivot separately from the plate lifting mechanism. Because of the difficulty of achieving that type of motion, alternate bending techniques were explored.

Fortunately, due to the way the spring steel plate flexes, early testing indicated that fixing the plate at one end and rotating it at the other would bend evenly along the plate. This method meant that as long as we could fix the plate at the top when it was pivoted up to a vertical position, it could then be rotated back and forth by the pivot to break parts free. This is achieved using small arms that rotate down on each side of the plate, constraining the top of the plate into a slot as shown in Figure 14. These arms are actuated by small servos. The actuation does not require much torque since the bending occurs in the other direction and there is no load on the arms in the direction of their motion or while they are moving. Since the plate reduces in length as it bends, it was necessary to position this slot a couple centimeters from the top of the plate. This allows the plate to slide and bend, but the plate will not pop out of the slot due to length reduction. The combination of these design choices ensures a reliable and simple part removal system.

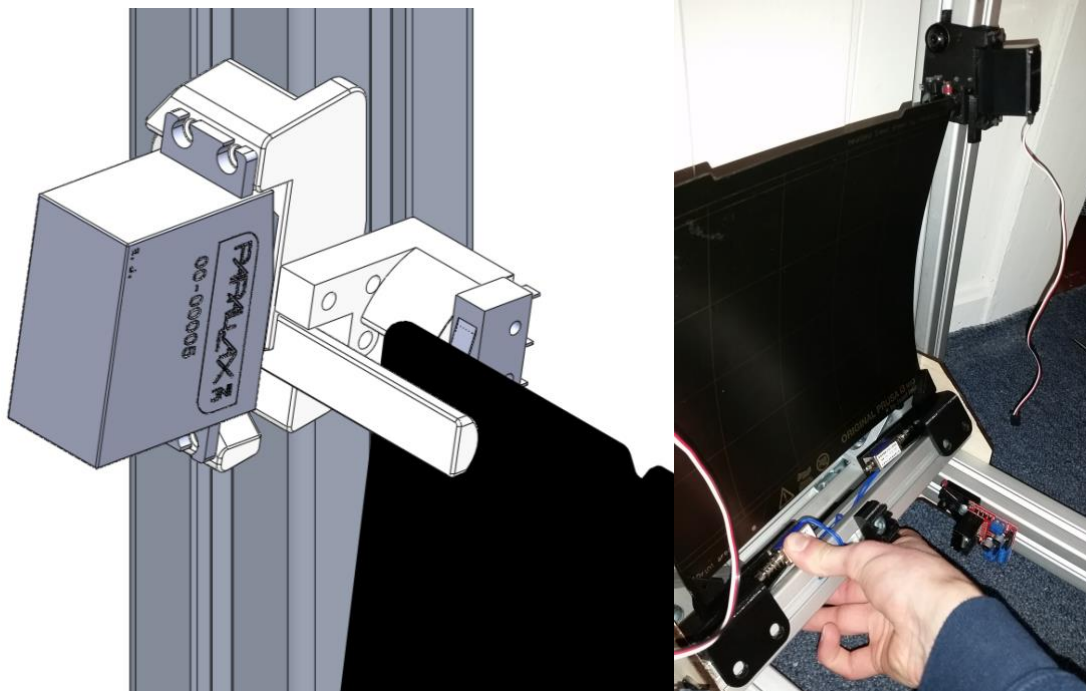


Figure 14: Early Plate Bending Mechanism and Testing

In our initial testing of the bending mechanism, we encountered a couple of unexpected issues. Our mechanism was designed to flex the build plate forwards and then backwards. The mechanism worked well flexing the plate forwards, but had an issue moving backwards. As the plate transitioned between being bent in one direction to the other, the plate bent in the vertical axis, which prevented the plate from being bent in the horizontal direction as we intended. This problem stemmed from only holding the four corners of the plate. There was nothing preventing the plate from bending vertically, and the plastic deposited on the plate biased the bend the wrong way. We observed that we could manually push in on the plate to pop it into the correct direction at which point it could continue to be flexed. To solve this problem, we installed a passive finger (Figure 15) that would rotate into the plate to pop it backwards and hold the center down so it could not bend the wrong way.

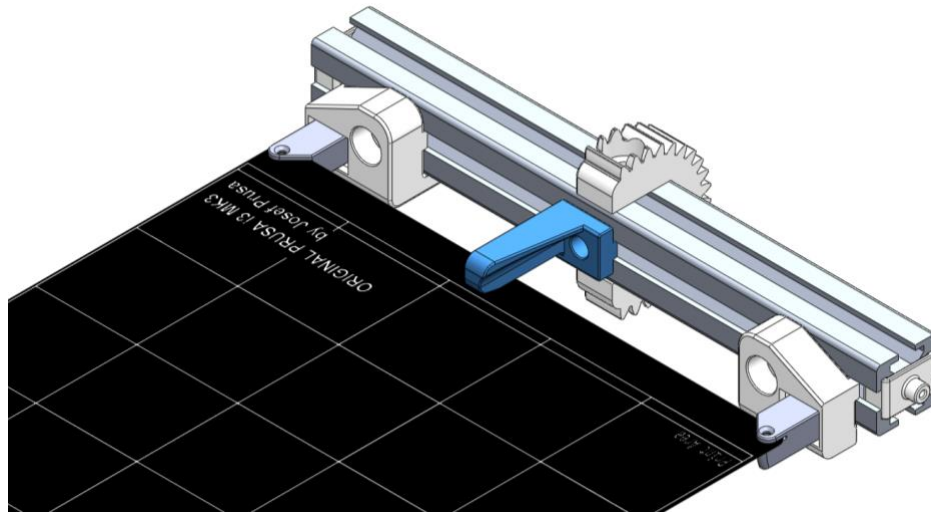


Figure 15: Finger Forcing Plate into Vertical Bend

One issue that we ran into because of the way we lift the build plate, pivoting instead of lifting straight up, is that some parts, particularly small ones, would be dislodged by the bending and then launched off the plate once it broke free of the last magnets. To combat this problem, two small servo mounts were attached to the sides of the printer as shown in Figure 16. On top of each of these servos is a small 3D printed wedge. During the plate removal process, when our system has partially lifted the front of the plate but the back is still attached by magnet, the two servos rotate in, driving the wedge in between the heated bed and the spring steel build plate. This breaks the connection with the plate more gradually to prevent launching parts when we lift the build plate. The mechanism is shown in operation in Figure 17.

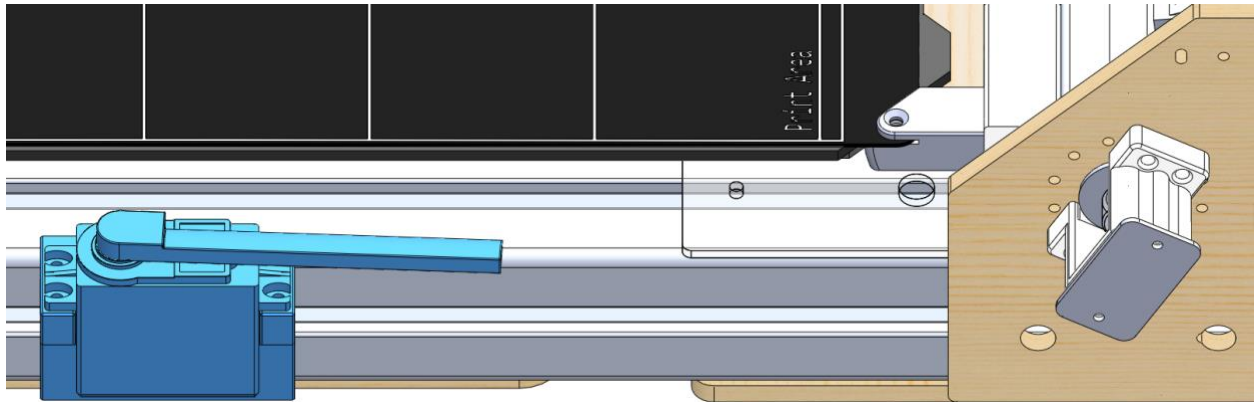


Figure 16: Anti-Catapult



Figure 17: Anti-Catapult Wedge Separating the Build Plate from the Heated Bed

Another issue we encountered was not being able to remove all the parts from the build plate. Bending the plate to extreme angles unstuck the parts from a large area of the plate, but many parts remained adhered to a small strip of the build plate. In early testing, we used a snow brush from a car to manually wipe the parts off of the plates. We determined that a brush mechanism to clean the build plate would be able to remove the remainder of the parts.

We designed a system, that can be seen in Figure 18, to facilitate the brush motion and actuation. It consists of a commercial masonry brush on a linear slide, powered by a geared DC motor. The system uses hardened 8mm rods with LM8UU linear bearings to enable the linear motion. Since the center of the brush mechanism must be open for the parts to fall through, we actuated the brush using a GT2 belt on both sides of the plate, leaving the center open. Both belts have a tensioning mechanism, seen in Figure 19, at the top and connect to the same shaft at the bottom which supplies the rotation to

each of the belts. We use limit switches on the top and bottom of the linear slide to detect the location of the brush.

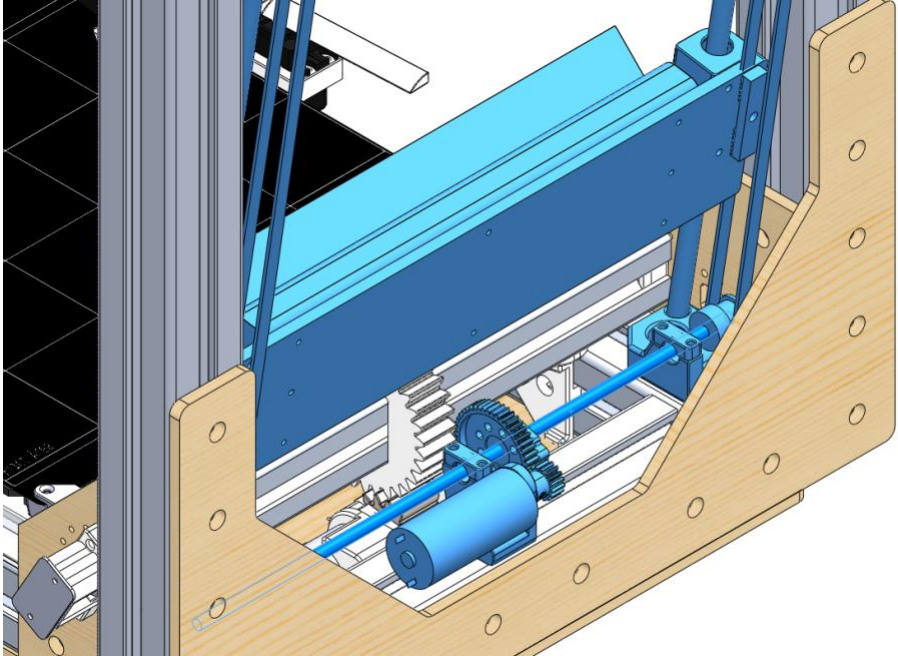


Figure 18: Part Removal Brush Assembly

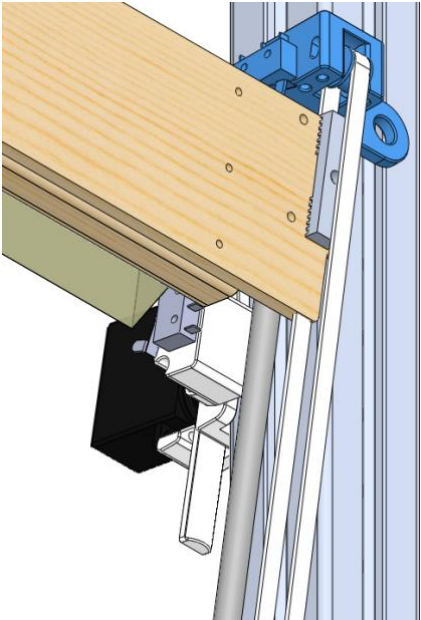


Figure 19: Brush Hard Stop, Limit Switch, and Belt Tensioning

In our transmission design and motor selection for the brush, we ensured it had the proper output speed and torque. The speed of the brush did not need to be fast, our goal was to be able to cycle back and forth the 290mm length in about 12 seconds. As previously mentioned, in our manual testing with the snow brush, the force to remove the part was not significant. We were confident in the

ability of the Servocity economy motor to achieve this motion. The motor is the same one that we used for the pivot, so it has a lot of power for its form factor, and it has many available gearboxes. Using our goal of moving 290mm in 12 seconds, we used the pulley diameter of 12.7mm to calculate the target speed in Figure 20.

Belts and Pulleys are GT2

Pulleys are 20T so have a circumference of 40 mm and radius of 6.3662 mm

Distance = 290 mm

Time = 12 seconds

$$\text{Target Speed} = \frac{290 \text{ mm}}{12 \text{ sec}} \times \frac{60 \text{ sec}}{1 \text{ min}} \times \frac{1 \text{ rev}}{40 \text{ mm}} = 36.25 \text{ RPM}$$

$$\text{Gear reduction of } 3 : 1 \rightarrow 36.25 \text{ RPM} \times 3 = 108.75 \text{ RPM with load}$$

Figure 20: Target Speed Calculations

On the bottom shaft, we use a 48 tooth gear connecting to a 16 tooth gear on the motor. The motor is a 57:1 Economy motor from Servocity with a free speed of 170rpm and 1.6lbf*ft stall torque. In Figure 21 we calculated the brush no-load speed of 37.8 mm/s and The force at the brush with the motor at stall torque to be 228.952 lbs.

Selected motor has no load speed of 170 RPM

$$\text{Brush Speed (no load)} = 170 \text{ RPM} \times \frac{1}{3} \text{ gear ratio} \times \frac{1 \text{ min}}{60 \text{ sec}} \times \frac{40 \text{ mm}}{1 \text{ rev}} = 37.778 \frac{\text{mm}}{\text{s}}$$

Motor has T_{stall} of 1.594 lbf · ft

$$T_{\text{stall}}(\text{at pulley}) = 1.594 \text{ lbf} \cdot \text{ft} \times 3 = 4.782 \text{ lbf} \cdot \text{ft}$$

$$F_{\text{stall}}(\text{at brush}) = 4.782 \text{ lbf} \cdot \text{ft} \times \frac{304.8 \text{ mm}}{1 \text{ ft}} \times \frac{1}{6.366 \text{ mm}} = 228.952 \text{ lbs}$$

Figure 21: Brush Force Calculations

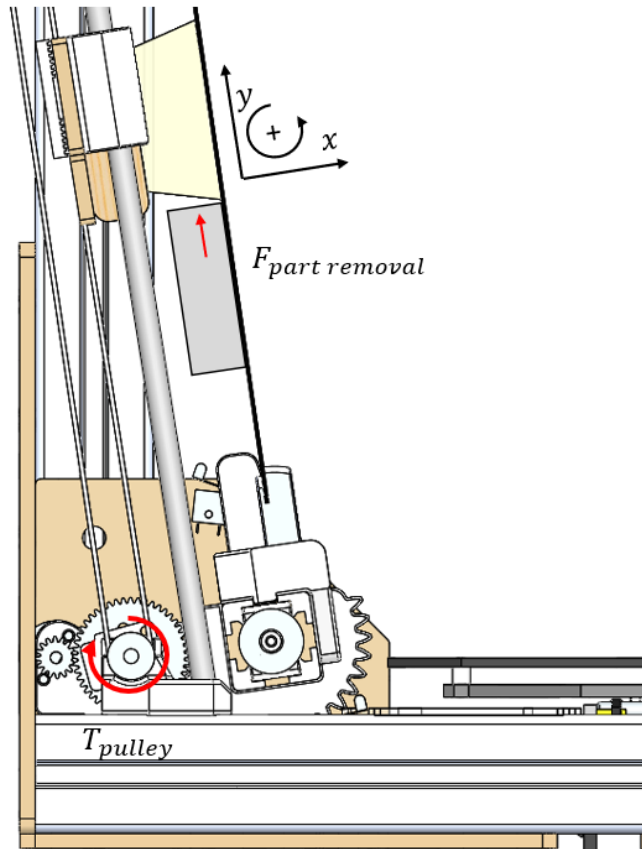


Figure 22: Brush Assembly Forces

3.3 Electrical System

The electrical system for this project is based around extensibility and future development. When selecting the motor controllers and power source, we made sure to select components that exceed the requirements of the current system and could be expanded upon with the development of the project.

Our motors for the pivot mechanism and brush system are both from ServoCity's line of economy spur gear motors. They run off of 12V and have a stall current of 3.9 amps. Both of these motors are driven by a Pololu Dual MCR33926 dual motor controller carrier board. This board allows for control of motors at voltages between 5 and 28 volts and has a high continuous current of 3A with a peak of 5A per channel. Because the motors will not be operating near stall current, this motor driver configuration gave us a lot of headroom and ensured that power was not an issue during testing or usage. Using the same type of motor controller for both of our motors simplifies the system from both an electrical and software standpoint as the wiring and control code is the same for both motors. For the purposes of powering these systems, we selected a 12V 30A switching power supply that easily handles our motors and a 5V 15A voltage regulator to power our servos and control boards. The centerpiece of our control systems is an Arduino Mega board connected over USB serial to the

Raspberry Pi that is managing the whole system. As the Raspberry Pi lacks a large number of PWM outputs or an ADC for analog inputs, we elected to offload the low level control of our motors, sensors, and servos to the Arduino Mega. While it does add in another hardware component to the electrical control system, it greatly simplifies the overall system wiring and control. An additional benefit of the Arduino Mega over other Arduino boards is that it leaves room for the expansion of the system and essentially guarantees that enough open IO pins are being left for a future storage system. Our finalized wiring diagram is shown in Figure 23.

For prototyping and testing purposes the system was originally designed in fritzing and then wired on a breadboard with a number of jumper wires as can be seen in Figure 24. As the system was developed and tested, limit switches and other sensors moved around and port mappings changed which would be a problem with wires cut exactly to length and soldered connections. Instead, wires had slack length built in and connections were made with jumpers and breadboards. Once positions and ports were tested and finalized, wires were cut to length for clean wire management and parts were arranged onto a common electronics board to make the system easier to transport and to reduce the risk of wires being caught in motion components during operation. This arrangement can be seen in Figure 25 with our Arduino pin mapping in Table 1.

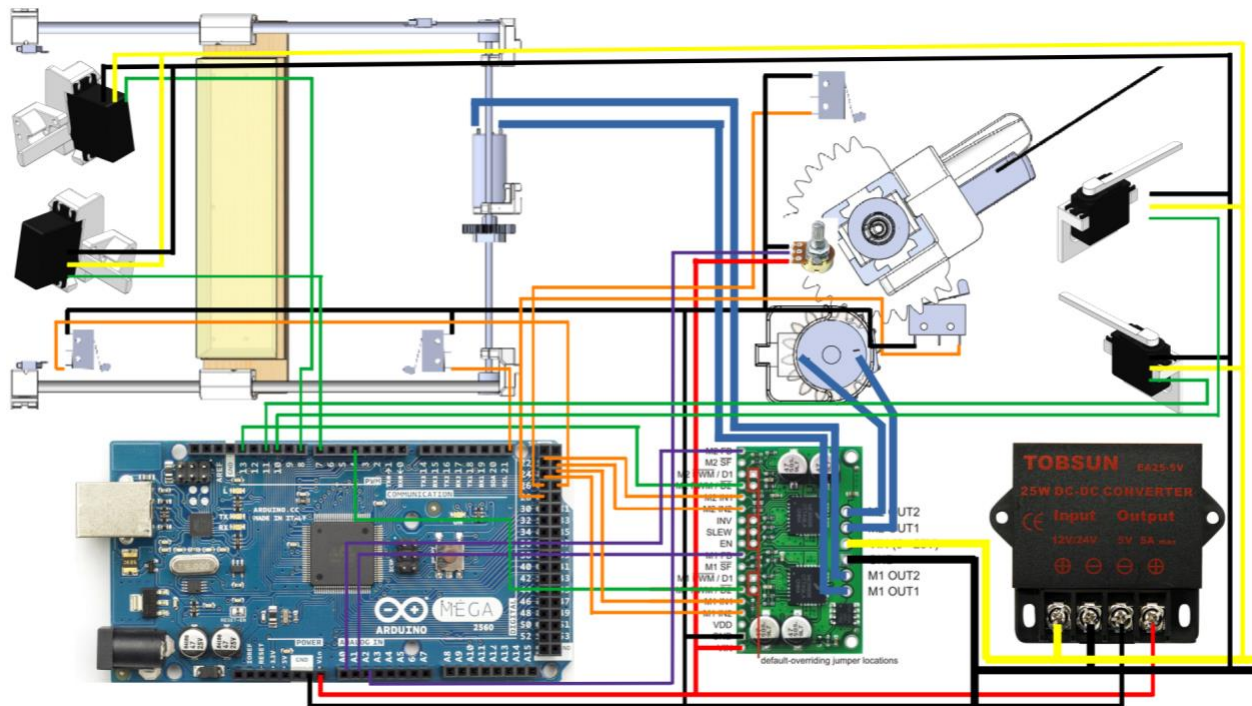


Figure 23: System Schematic

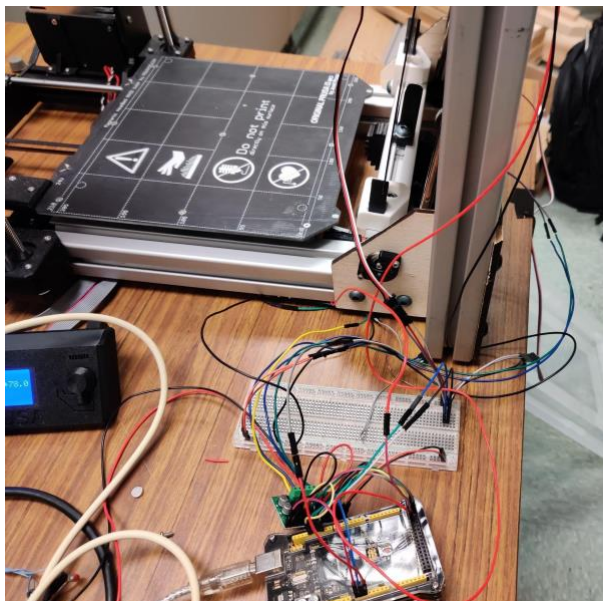


Figure 24: Prototyped Wiring Setup

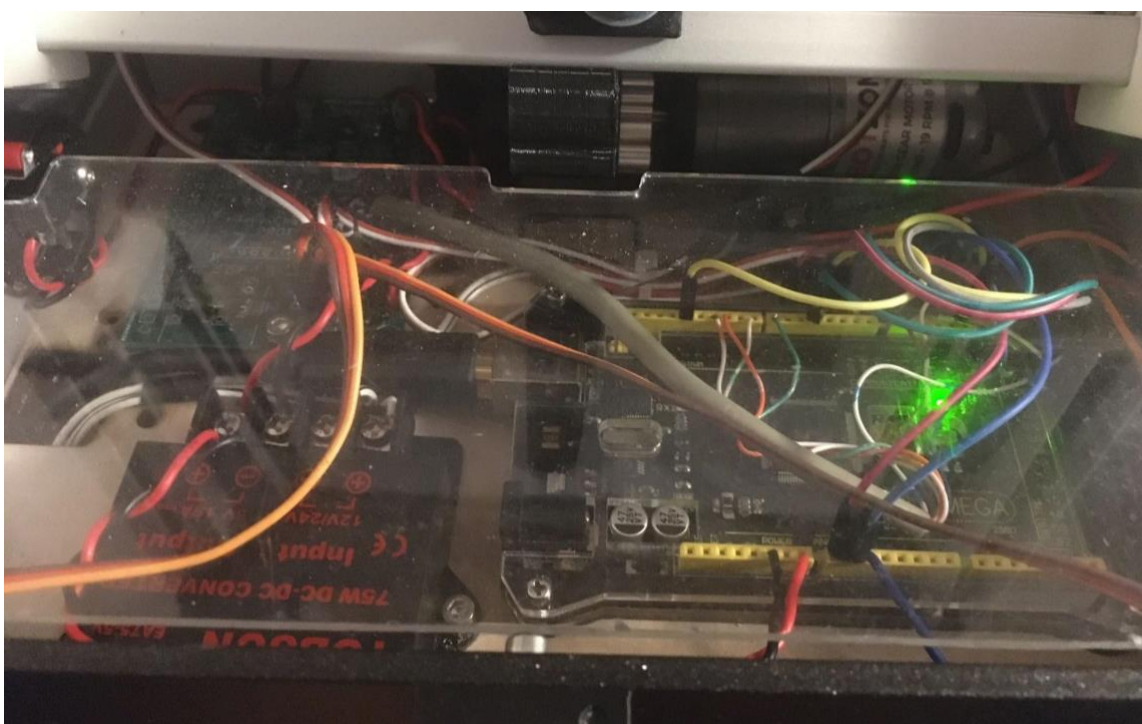


Figure 25: Final Electronics Board

Subsystem	Description	Type	Arduino Pin
Rotation	PWM D2 for Rotation Motor	PWM	D13
	Rotation Motor IN1	Digital IO	D22
	Rotation Motor IN2	Digital IO	D23
	Rotation Motor Current Sense	Analog	A0
	Left Bed Lock Servo	PWM	D11
	Right Bed Lock Servo	PWM	D10
	Left Anticatapult Servo	PWM	D7
	Right Anticatapult Servo	PWM	D8
	Top Rotation Limit Switch	Digital IO	D26
	Bottom Rotation Limit Switch	Digital IO	D28
	Rotation Angle Potentiometer	Analog	A2
Brush	PWM D2 for Brush Drive Motor	PWM	D4
	Brush Motor IN1	Digital IO	D25
	Brush Motor IN2	Digital IO	D24
	Brush Motor Current Sense	Analog	A1
	Bottom Brush Limit Switch	Digital IO	D21
	Top Brush Limit Switch	Digital IO	D27

Table 1: Arduino Pin Mappings

3.4 Queueing and Printer Control Software

The software architecture is broken up into two main components, the client and server. The client software runs on a Raspberry Pi and is in charge of managing the printer and associated automation systems. The client also communicates with the server to receive print jobs to start and update the current status of the machine. The server software runs in the cloud and provides the user interface in the form of a website and manages the upload and scheduling of jobs. Both components are written in python to make development consistent.

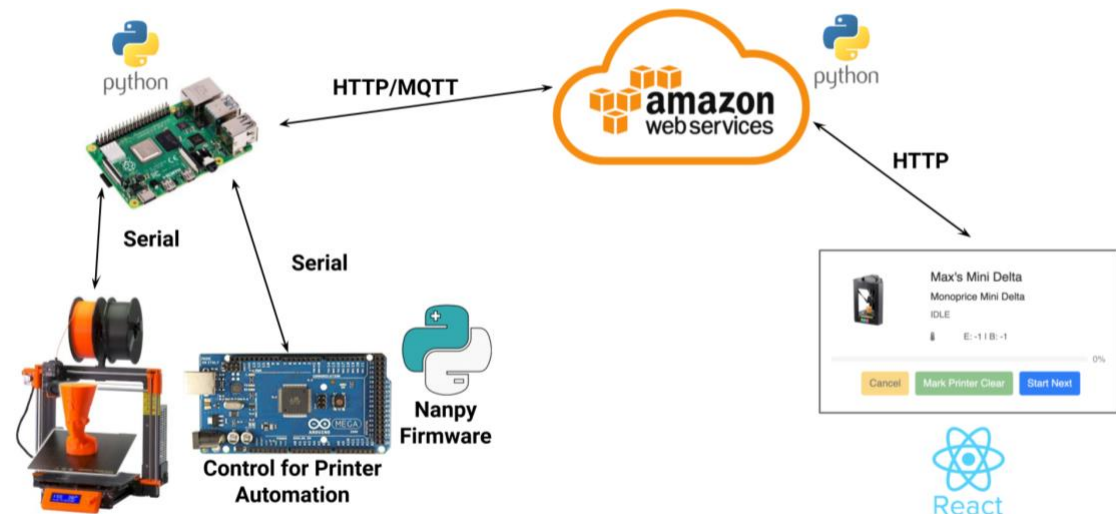


Figure 26: Software System Overview

3.4.1 Client Technologies

The client serves as a middle layer between the server and printer. Commands are sent and received to the server with information about which print should be started. For starting prints, this takes the form of some metadata about the print and a link where the client can download the gcode file locally. This gcode file is then streamed to the printer over USB serial. This operation runs in a separate thread than the web server so that commands can still be received. Prints can also be safely aborted through another request forwarded from the server.

During the print process gcode commands are streamed line by line to the printer. After each command the client sends the printer responds with an acknowledgement that the action has been completed. The client relays information about the current print back to the server periodically. During the heating process the printer echos back the temperature of the bed and extruder which is then read by the client. To continue to get a measurement during the printing process a periodic temperature query signal is sent to the printer. These temperature measurements, along with the system state and progress of current print are sent to the server to be displayed to the user.

The client device also communicates with the custom hardware for the plate removal through serial to an arduino. This communication is handled through a python library called Nanpy. The library uses a custom firmware which runs on the arduino and then allows for basic arduino operations to be run from within python code. This allows for the same code base which runs the printer communication and web server to also do motor and servo control. This was chosen to minimize the amount of components which need to communicate, as the arduino is almost plug and play into python. This does come with a lag caused by the communication over serial for all the IO operations, but due to the slow speed of all the mechanisms it hasn't caused any control issues. If the connection to the arduino fails, there is a risk that the python code will not be able to send a stop command to the arduino, but this was mitigated by restoring the connection as soon the arduino is reconnected.

For setup and troubleshooting, a user interface is accessible to anyone on the same Local Area Network (LAN). This control panel is hosted as a webpage by the client software to display current sensor values, enable direct control of motors, and allow the user to adjust device specific parameters. The sensor values and device status are broadcast live to the browser application from python using Socket.io. The use of WebSockets was important to get live data in an efficient way instead of having the UI constantly poll the server with REST requests which would be inefficient. User requested control commands are sent through a REST API, and queued to the hardware controller. The device specific parameters can be edited in a settings panel, and updated values are also sent through REST requests in JavaScript Object Notation (JSON). To maintain persistence, the values are saved locally to a file on the device running the client software. When the program starts it loads its configuration from the saved file.

3.4.2 Server Technologies

The server acts as the interface for the user and manages the database access for data about print files and the queue. The server is hosted on Amazon Web Services (AWS) using a service designed

for scalable web applications called Elastic Beanstalk. The service automates the process of uploading, deploying, and load-balancing for the application making development faster and simpler. There are two independent environments running for the current stable build and the current development build. These are hosted on 3dprintqueue.com and dev.3dprintqueue.com.

The web server is written in Python and utilizes Flask as the framework. Flask provides a simple lightweight way to create API endpoints for custom URLs. It also allows for integration with the AWS Cognito authentication service to secure certain URLs to require a user to be logged in. Though this framework both user facing pages and the printer client API can be exposed. The server receives status updates from the printer and stores data to be relayed to the client UI.

The server system stores all of its relevant data through various AWS services. S3 (Simple Storage Service) is used for all large files. This includes storage for all printer assets, including the Slic3r instance and associated configuration files, images for the website, and software assets to be deployed. In addition, S3 is also used for all user uploaded files. This includes uploaded STLs as well as any 3MF or gcode files generated from the slicing process. Storing these files in S3 makes it very easy to manage and transfer to the client when ready for printing.

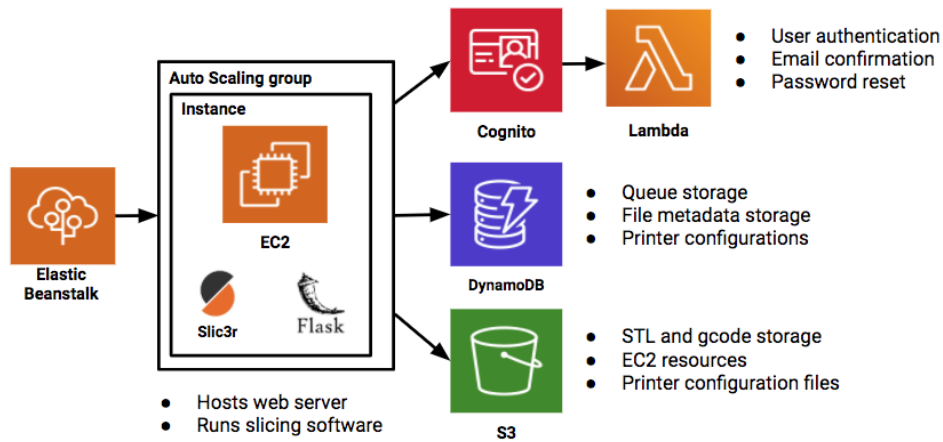


Figure 27: AWS Cloud Structure

The system also has a database for user and printer data which is hosted on AWS DynamoDB. This is a NoSQL database which is simple to use as it doesn't require any server instances to be spun up. As Cognito hands all information about users, this database is mostly used for printer management. Each printer has data stored to tell what user it belongs to, what type of printer it is, and a link to its queue of prints. Every sliced gcode file also has a table entry with its relevant parameters from the slicing process. These two tables join together in the queue which represents all the jobs in line for a given printer.

3.4.3 Client and Server Communications

There were multiple iterations of the communication protocol used for communication between the client and server. The initial implementation was simple and easy to get working but wasn't scalable to actual use outside of the WPI network.

The first iteration of communication utilized REST calls over HTTP between the server and client. The server stored the address of the client attached to each printer in the database so the server would know where to contact the client. Normally it would be difficult for the server to send a message to the client this way because most clients wouldn't be running on a static and easily known address. It is possible to port forward the router the client is running on in order to expose the port used by the webserver on the client so the server could access it. This however adds possibly complex configuration steps and opens up the user to security risks.

The issue of static IP addresses was able to be avoided for our initial testing because the WPI network assigns a publicly accessible domain for all devices on the network. This feature allowed the clients to have a static and publicly visible address which would then point to the webserver running on the device. These domains can be configured through the WPI netreg website and come in the form <HOSTNAME>.dyn.wpi.edu. These domains were useful in development for being able to easily change between which devices were running which software and to have an accessible domain to run the server locally instead of on AWS.

This solution however was tied to the WPI network and not intended to be in the final product. The chosen solution for the final product was to use MQ Telemetry Transport (MQTT) to send messages. MQTT is a publisher subscriber messaging framework used heavily for Internet of Things (IoT) devices due to its lightweight design and reliability. Messages are managed by a MQTT broker which acts as the service to manage the passing of messages. There are many implementations of brokers but for this project we utilized an open source implementation from Eclipse called Mosquitto due to its low footprint and known stability. The broker is set up to run on the same AWS EC2 instance which hosts the web server so the clients can access the broker using the same public domain address as the website. The client and server are able to subscribe and publish messages on channels to each other uniquely identified with each printer's id. The server is able to tell which printers are online through knowing which printers are currently subscribed.

3.4.4 User Interface

The user interface allows the users to upload parts and schedule prints. As a website the UI is cross platform and more easily accessible to users. There are also many front end web frameworks which aid in the development process. Our project utilizes many of them for different portions of the user interface.

For layout and components we decided to use Bootstrap as it is a common front end design framework. It allows to arrange components using its grid system to allow for a responsive website which can work on mobile and desktop. This allows for a useful layout through many different screen

sizes that users may have. The UI uses many components, such as buttons and progress bars, with a consistent theme for various parts of the site.

For the components which interact significantly with the user React is utilized. React is a powerful JavaScript framework developed by Facebook which allows for the creation of interactive components which are aware of their own state. Through this system we were able to develop components modularly to reuse common aspects throughout the site. This helped power the many interactions users have in uploading and slicing parts in their user portal. The queue system is also enabled by a drag and drop React component allowing for the simple dragging of parts to change their order in the print queue.

3.4.5 User Workflow



Figure 28: User Software Workflow

Once users log into the system, they enter the user portal screen, which allows them to complete most of the workflow to print a part. First the user is able to upload parts and group them by project. The organized parts can then be viewed, and sliced for any of the users printers through the portal. Sliced parts can then be added to the queue to print on the desired printer, and queues can be reordered to prioritize a part. When a part is printing, the status and progress of the print is displayed so the user knows when the part will be ready.

3.4.6 Online Slicing

A requirement was for users to be able to slice parts through our system. We implemented a system for the AWS server to slice a part through the user interface. The user interface for this had textboxes to specify a selection of common slicing parameters, and after slicing it would display basic information about the sliced part such as print time, and material used. The actual slicing is done using the Command Line Interface (CLI) of the PrusaSlicer software based on Slic3r, a common open source slicer. To run the slicer online, the configuration of the AWS EC2 instance running our server needed to be updated to support the necessary dependencies. When a slice command is sent from the browser application as a REST API command to the server, it is handled by the flask instance on EC2. The flask application then temporarily copies the requested file from its location on S3, and commands PrusaSlicer to slice it with the requested parameters. The output is then parsed to determine the print time and material usage, then copied to S3 before being locally deleted.

3.4.7 Part Visualization

To allow users to view uploaded parts, we added support for part visualization inside the web application. When each part is uploaded to the server, PrusaSlicer generates a 3mf (3D manufacturing format) file representing the part. The 3mf file type is ideal, because it was designed for 3D printing, and more precisely represents the parts than other formats such as stl. The 3mf file is uploaded to the server, and if present allows users to open a visual in the browser. This is achieved using the popular javascript graphics library three.js. While three.js has built in support for interpreting the 3mf file format, it had to be adapted to fit within a modal in our user interface, and interpret data from s3.

3.4.8 Vision Processing

While the global pandemic at the end of C-Term made it impossible to buy a webcam to integrate into the project, a vision processing system was developed to verify that the build plate was correctly cleared before starting the next print. The system is written in python using openCV, and is used by the client software. Unfortunately, most prints and most build plates are black, but in most cases there is still a small glare on the plate from the filament. This difference in lighting from the build plate is amplified using the sobel derivative. The raw image is converted to grayscale, then blurred to reduce noise. The blurred image is then convoluted by the vertical and horizontal sobel operators and the results are combined to produce an image representing how the brightness changes through the image. From the resulting filtered image the system can easily identify if there is still material on the plate, and wait for it to be cleared before starting the next print.

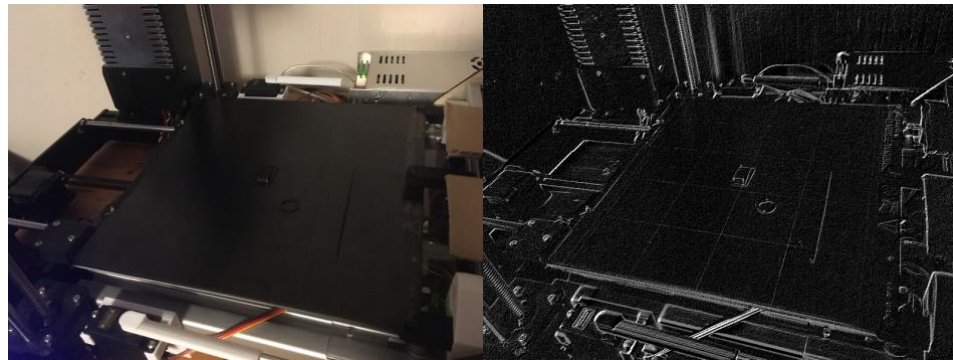


Figure 29: Raw and Filtered Image of a Messy Build Plate

4.0 Results

4.1 Part Removal Results

Overall testing demonstrated that the system is capable of removing most prints, but may struggle with small flat prints. All tests were conducted with PLA filament, a .2mm layer height, 15% infill, and 3 perimeter layers. During the worldwide crisis caused by the spread of COVID-19, many 3D printer owners have been repeatedly printing visors for face shields that could protect medical workers

short on supplies. As this is currently a common situation where a part removal system would be valuable, the visor of a face shield (as shown in Figure 30) was tested extensively as a sample print. As the part covers a large area and is rigid enough that it does not flex with the build plate, the visor consistently separated and fell off the plate without even needing the help of the brush.

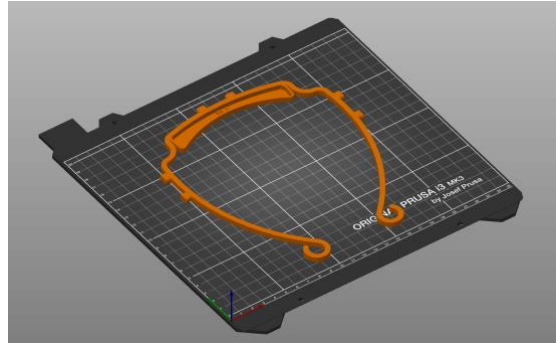


Figure 30: Face Shield Visor Test

To test the effectiveness of objects with a smaller footprint, removal was tested with six 10mm cubes distributed over the plate. The small size of the cubes meant that the flex of the plate was not sufficient to separate the cubes from the plate, but the small area of contact with the plate resulted in low enough adhesion to be easily removed by the brush.

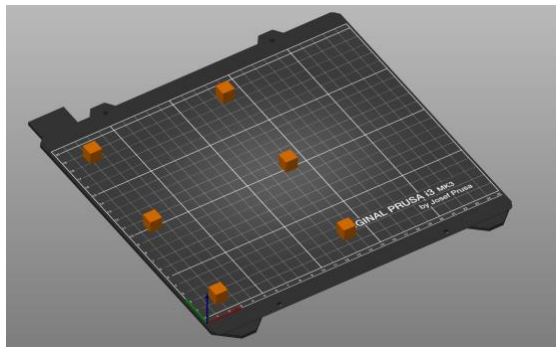


Figure 31: 10mm Cube Test

The largest weakness of this removal system proved to be objects small enough that they would continue to stick to the plate as it was flexed. To determine the point this occurred, a series of flat disks were printed with a range of dimensions as shown in Table 2. This test demonstrated that flat disks needed to be at least 1mm thick and a diameter of at least 20mm to be removed. Objects with a smaller footprint also need a substantial. Although some small prints such as spacers may not be removable using this method, our mechanism does not significantly limit what designs can be printed and removed. Unfortunately, most unsuccessful prints fail on the first few layers, so it is unlikely that this method would be able to clear and restart most failed prints.

Diameter (mm)	Thickness (mm)	Successfully Removed
30	2	Yes

30	1	Yes
30	.5	No
30	.25	No
20	2	Yes
20	1	Yes
15	2	Yes
15	1	No
10	6	Yes
10	5	No
10	4	No
10	3	No
10	2	No
10	1	No

Table 2: Flat Disk Removal Results

By default most slicers generate a strip at the bottom of the plate and a skirt around the part to purge the filament in the chamber. While these are typically discarded it is important that the system can remove them before starting the next print. Due to its flat shape, the default strip at the bottom of the plate was not consistently removed; however, changing the shape and location made it easy to remove. Increasing the height of the strip made it sufficiently easier for the brush to remove the thin strip.

Unfortunately, one minor flaw of the current design is items removed by the brush when moving upward occasionally become stuck in the brush, or move over the top of the plate and end up between the heated bed and the plate. This can be mitigated in software by tilting the plate away when lifting the brush, but this slightly reduces the effectiveness of the brush. This is rarely an issue because most parts are removed on the initial downward motion of the brush, but there is still room for improvement.

4.2 Machine Performance

Our plans changed due to Covid-19, but we were still able to get a majority of the project functional. The most essential parts of the project were the plate removal and part removal. As discussed in the previous section, the device was able to do both of these tasks successfully for most printed parts.

The pivot worked very well. It was built robustly with very little backlash. The motor and gearing we selected had no trouble lifting the build plate off of the magnetic heated bed. The potentiometer measuring the position worked well, and the pivot and build plate properly engaged with the limit switches as designed.

Below, in Figure 32, we plotted the pivot position during the plate and part removal processes. In the first few seconds, the build plate is elevated to eight degrees. This disengaged the front of the plate, and allowed the anti-catapult wiper to remove the back half of the plate off of the magnets without any springing motion. Next, the pivot was moved at a lower voltage up to 95 degrees. Once it has reached 95 degrees, the pivot moves back and forth twice, eight degrees in each direction. The plate moves to 50 degrees and back to help funnel parts off of the plate.

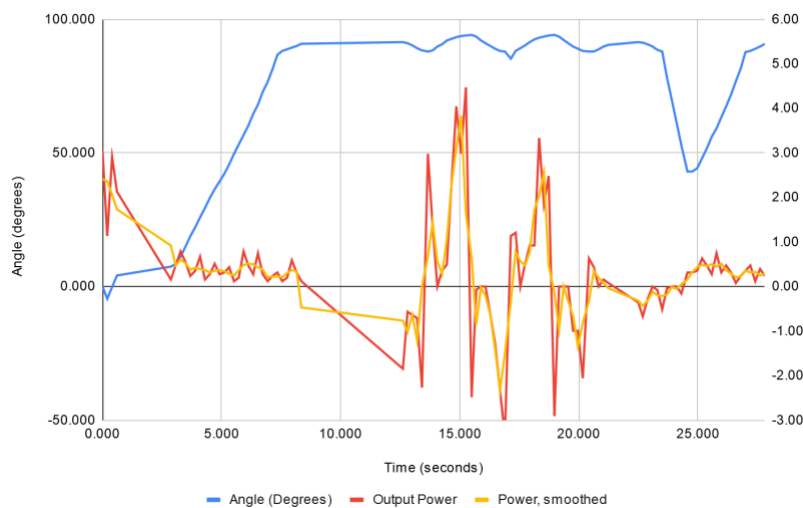


Figure 32: Pivot Position and Power Over Time

Figure 32 also shows the power draw in the plate removal and part removal process. We purchased a motor controller with current sensing, but after reading through the datasheet and some forum posts, the current sensing hardware on our motor controller board is not very accurate and we were unable to source an alternate current sensing solution in time. Fortunately, we did not rely on current sensing for control so the performance of the system was unaffected by this problem with the only downside being less usable data about the actual operation. The plot above shows the proportional power draw of the motor. There is a significant power draw as the plate is being removed from the magnetic bed while the plate is flexed in either direction.

Another smaller system is the anti-catapult system. In our initial testing in December, the build plate had a springing motion when it disengaged from the magnets. In January, we confirmed that parts loosely attached to the build plate would be thrown off of the plate during this process. We designed the anti-catapult mechanism to sweep a wiper in between the heated bed and build plate. This wiper helped remove the plate from the magnets slowly without having to raise the pivot to an extreme angle and deform the build plate. Our anti-catapult system successfully prevented the parts from flying off during the plate removal process.

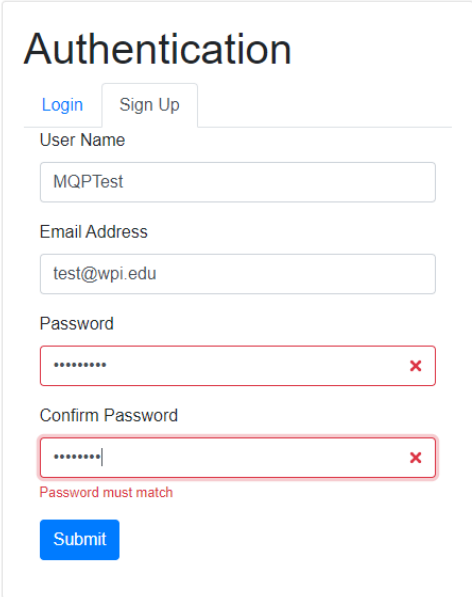
In our final testing, we encountered one problem that we had previously overlooked. Since we did not implement the storage system due to time constraints from the pandemic, we also neglected to design the chute system that would funnel parts from the plate and brush them into the storage system. This resulted in an issue where the build plate would be rotated to vertical, flexed, and then the parts would fall off, but not move out of the printer as intended. In the worst cases, parts could fall into the electronics or pivot gearing, or more likely, be crushed by the brush motion. This is a very solvable problem, though we did not realize it was an issue until in the latest stages of our testing. We made a small ramp out of cardboard and added an additional movement by the build plate between flexing and the brush motion to help dislodge the parts so they would not be crushed by the brush.

4.3 Website Features

The website serves as the main interface between users and their printer. This website was custom built for this project and has a lot of features for helping the user experience. The main page contains information about the team and links to our blog which covered some of our early progress.

4.3.1 User Authentication

Users are able to create an account and login to the website backend to allow access to personal files and printers. This system uses an AWS service to handle all the password hashing and management. The only step to increase security would be to enable HTTPS on the website by acquiring a certificate which was not done for this stage of the project. As shown in Figure 33 the login and signup UI provide live validation for user name, email address and password.



The screenshot shows a web form titled "Authentication" with two tabs: "Login" and "Sign Up". The "Sign Up" tab is selected. The form contains the following fields and elements:

- User Name:** A text input field containing "MQPTest".
- Email Address:** A text input field containing "test@wpi.edu".
- Password:** A password input field with masked characters and a red "x" icon on the right.
- Confirm Password:** A password input field with masked characters and a red "x" icon on the right.
- Error Message:** Below the "Confirm Password" field, there is a red text message: "Password must match".
- Submit Button:** A blue button labeled "Submit" at the bottom of the form.

Figure 33: User Signup UI

Once accounts are created users will be sent an email with a verification link to confirm their email address. This was implemented as an added security step and to allow for features like password

reset. If a user forgets their password they will be able to request a reset which will send an email with a link to click containing a unique code. This link leads to a page on the website to update the password. An example of the email is shown below in Figure 34.

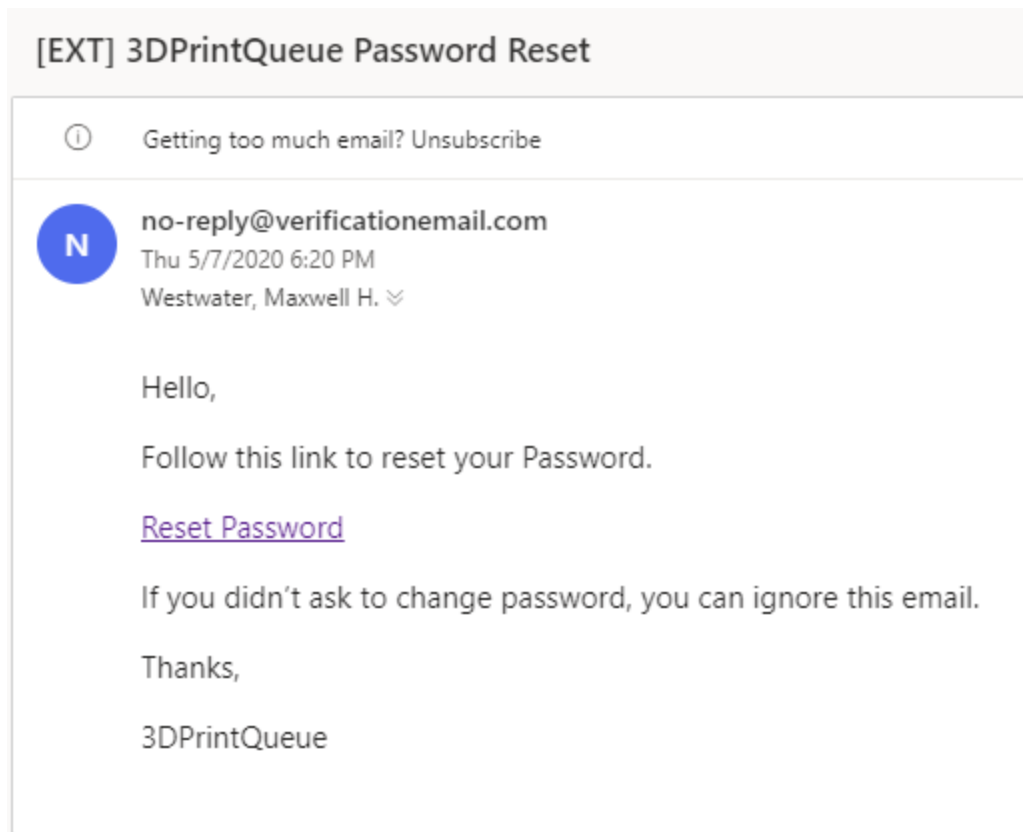


Figure 34: Example Email for Password Reset

4.3.2 Parts Management

Logged in users are given access to their portal which contains the ability to manage different projects containing parts. Users can create a project which is a way to group parts together. Each project then contains parts which are made up of one solid model file (stl/3mf pair) and the gcode files which are sliced for different printers or settings. An example is shown in Figure 35 where “Pokemon” is a project containing files “Bulbasaur” and “Pikachu” each with their respective model and sliced gcode files.



Figure 35: Part Management UI

Users are able to delete anything they no longer want. This can either be done as an entire project, an entire part, or individual files. When hitting the delete button on a specific part the user will be presented with the modal popup in Figure 36 allowing them to delete either a single file or the whole part.

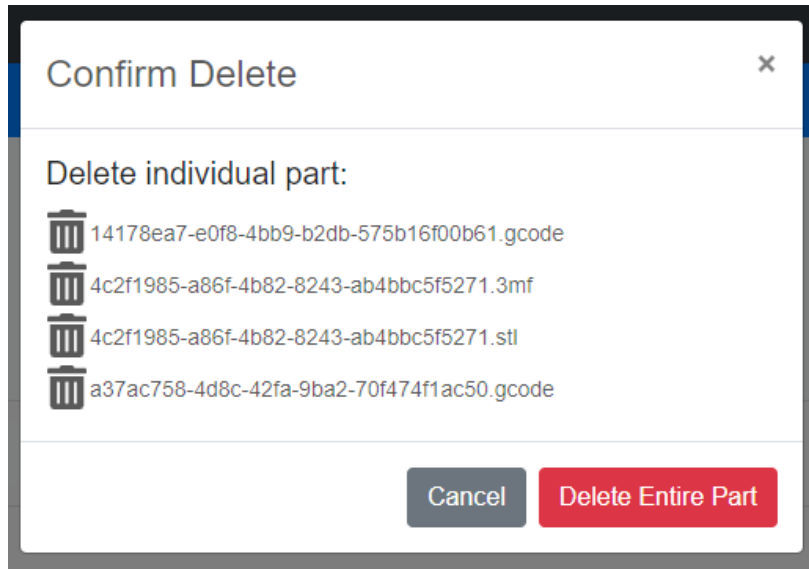


Figure 36: Example of Deleting a Part

Parts can be visualized by clicking on the view button associated with that given part. This button launches a popup which is shown in Figure 37 where the user can pan, rotate, and zoom around the part file.



Figure 37: Example File Preview

4.3.2 Queueing and Printer Management

When a user wants to print a part they hit the queue button for that corresponding part. This will open the popup shown in Figure 38. This lists all the printers a user has in their account and filters out sliced files based on what printer they are for. Once the queue button is hit they're added to the end of the list of files to be printed on that printer.

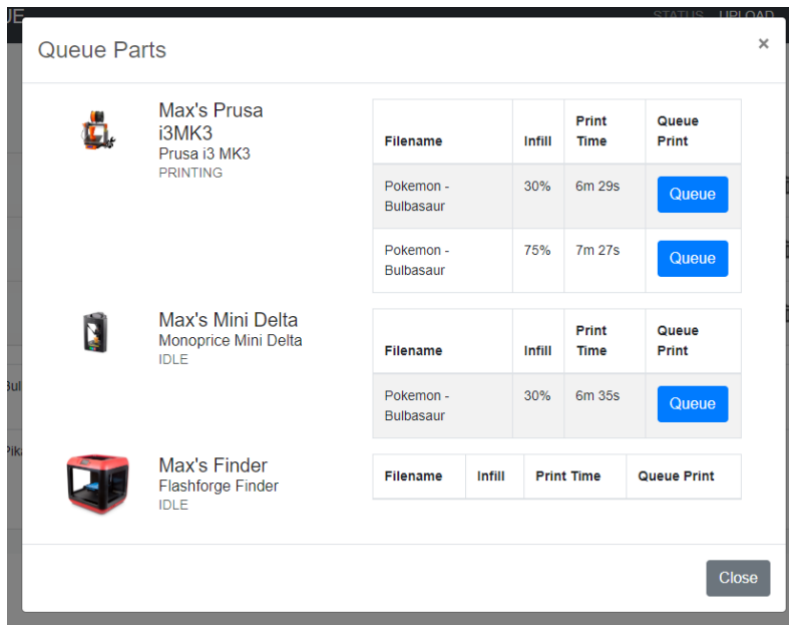


Figure 38: Printer Queueing Interface

Once added to the queue the user can scroll down to the section which shows the information and status for each printer as shown in Figure 39. All the parts in the queue are shown in a list on the left side and can be dragged to change order or removed completely. Currently the preview is a stock image but could be switched to pull a capture from the part preview generated above.

The right side shows information about the printer (name, model, and an image) in addition to the current system state, temperatures, and completion percentage. Depending on the state one of the 3 buttons will be highlighted to click. This system supports standard printers which can't remove parts automatically by having the user manually remove a part and click the "Mark Printer Clear" button to allow for the next part to be printed. For the automated printers this button will only be used for recovering from error states. Once the bed is clear the next print in the queue will either start automatically for an automated printer or when the button is pressed for a standard one.

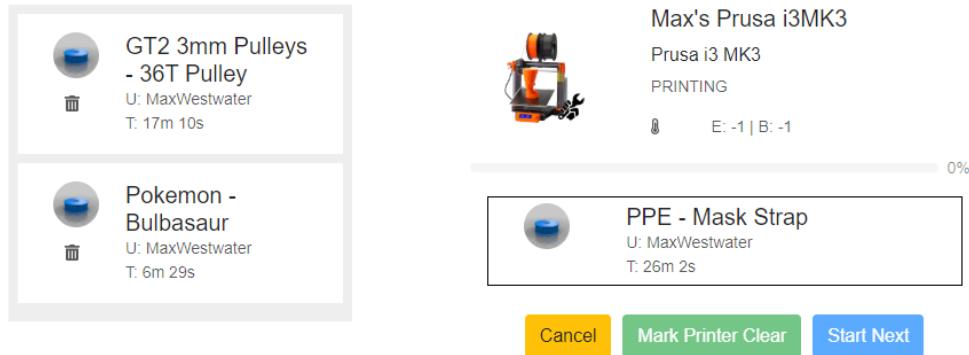


Figure 39: Printer Management Interface

4.4 Printer Client Software

The client software was able to serve as the intermediary between the server, printer, and arduino running the automation hardware. The flexible architecture allowed for the system to work with a wide array of 3D printers and arduino boards if needed. The networking interface is flexible to use either MQTT or HTTP depending on the individual use case. The use of MQTT made it possible to communicate two ways with the server without having to rely on port forwarding or any other networking setup. The connection to the printer was also tested to work with multiple types of printers, as a few different models were used in testing. This allows the system to be generalizable if users want a different or multiple types of printer.

There were some fears about latency and reliability of the Nanpy library used to control the Arduino from Python. Due to the speed of the motors in the system the latency of running the PID controllers from within Python presented no issue. However, there were some issues encountered with reliability of the serial connection. These were mostly all resolved by adding reconnection logic to the application so it would carry on with its task after reestablishing a connection. To make this system more reliable for long operation it would be ideal to implement some kind of failsafe heartbeat signal so that the Arduino could stop all outputs if the connection dies for more than a momentary period.

5.0 Discussion

5.1 The Implications of Global Pandemic

Around the time of the end of C-Term of 2020, the virus COVID 19 began to spread from Asia and Europe into the US. As a result of this outbreak, school administration decided to delay D-Term and move all classes online. This was announced over spring break while we were away from campus and presented a fairly large barrier to the completion of this project. These unusual circumstances required significant adaptation of workflow and a reprioritization of features and goals.

The biggest problem that the remote work during D-Term presented for us was the inability for more than one person to be working on the hardware at a time. Fortunately, the project was small enough for someone to take home but social distancing measures and concerns about the virus forced us to only have one team member work at a time. Additionally, to limit risk we decided that the project would only be able to be passed off once. Tom and Oliver, being the two group members closest to each other and to campus were selected to try to get things working. Tom was given two weeks to bring mechanical and electrical work on the project to a close and prepare the system for the final software work by Oliver.

We had to decide how best to fulfill the core goals of the project taking into account the altered circumstances. The largest change that came as a result of this was the removal of the storage system from our goals for D-Term. While some conceptual work and brainstorming had been done, we had not actually finalized designs or started work on the actual system and this would have no longer been possible to build while still allowing for software integration and troubleshooting before the end of the year. The other place where expectations had to be adjusted was reliability and testing. With only one person able to work on things at a time, the entire process of troubleshooting had to be reworked for D-Term. After the printer was passed off to Oliver for software integration and testing, he was the only one able to work on troubleshooting mechanical or electrical problems. This meant that anytime an issue was encountered, work on integration and other testing had to completely stop until he was able to address the problem. Under normal circumstances we could have worked together with one person working on a fix for the problem while the others continued making progress but that was unfortunately no longer possible.

Due to the timeline the team set, the project was close to being fully functional and only needed some final integration work to accomplish the major goals of the project. Because the bulk of the work unfinished at the end of C-Term was on integration and the development of a storage system, we were able to cut back on storage and still fulfill the main goal of the project, a continuous and automated 3D printing solution.

5.1.1 Utilization of the project due to COVID

The global pandemic caused a national shortage of Personal Protective Equipment (PPE) for front line healthcare workers. The maker community responded by taking action to help produce face masks. The software was modified to support two printers at once and to replace the mechanical plate removal system with a set of status LEDs and mechanical buttons. The LEDs show the current printer progress of heating, printing, and if the build plate needs to be cleared for the next print. The buttons act as a way for the user to easily mark the plate as clear and start the next batch of face shields. The system also sent notifications to the user's phone to know when the parts were done. These changes were easy to implement due to the modular design of many parts of the client software. The system is shown below in Figure 40.



Figure 40: Printers Setup for Printing Face Shields

5.2 Possible Future Work

5.2.1 Part Storage

The part storage system was originally intended for completion during D-Term but had to be put aside as a result of the adjustment to working remotely. This section will discuss our early approach to solving this problem and how it could be expanded on.

5.2.1 Storage Constraints

There were a couple of main objectives for the storage system for this project. First and foremost, the size of the system must be kept to a minimum. If this project is to be something that users can implement themselves, it can't take up an excessive amount of space. We wanted to keep the entire

system to roughly the size of an average workbench or table so the storage system must be designed intelligently to fit within this space. Furthermore, we set a requirement that it utilize some sort of non-trivial sorting solution. Essentially, we don't just want all parts to fall into a bucket that users then have to sort through to find a specific part. Each part doesn't necessarily have to be stored individually but users shouldn't have to look through more than a few parts to find theirs if the system has been running for a while.

5.2.2 Considered Storage Solutions

At the end of C-Term we were exploring a storage system that makes use of a number of small boxes where parts will be stored, perhaps 2 or 3 to a box. This system allows for larger parts to have their own box but smaller parts that don't require that kind of space wouldn't need to take up a whole box. Ideally the system will allow for parts to fall directly down from the part removal process into the storage area. In this case, there would be some sort of ramp or cushioned landing area to prevent parts with delicate features from breaking when they fall. Parts would then be directed or placed into a small bin or box in the storage area, with the bins either forming a row on a linear slide as seen in Figure 41, or a circle on a rotating carousel so that the correct bin can be positioned for the part to drop into. When a user goes to retrieve a particular part, they should be able to see which bin the part is in for easy retrieval.

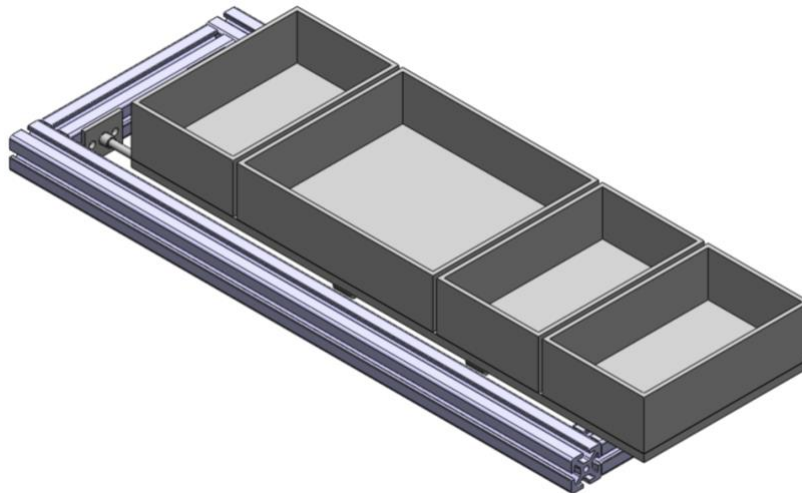


Figure 41: Storage System Design

5.2.3 Possible Improvements to Brush Design

While the brush successfully combines with flexing the plate to remove most common prints, there is room for improvement in the removal of extremely flat prints, and prints that fail on the first layer. It is possible that by experimenting with factors such as the angle, stiffness, and density of the bristles the brush could become more reliable at handling these edge cases.

5.2.4 Software Improvements for Large Print Farms

A concept that we discussed early on was the possibility of a collection system for users without access to the printer itself. Perhaps in a print farm like the makerspace in WPI's Foisie Innovation Studio, it may be desired that someone picking up a print could key in a code or student ID and have their parts delivered without them being able to interact with the system itself. This would require a more elaborate sorting system as well as the implementation of some sort of retrieval mechanism but would greatly expand the utility of an automated 3D printing system.

With such a system in place it might also be advantageous to users to be able to chain multiple automated printers together. A bank of Prusa printers all printing and removing parts into one common storage and retrieval system would make this project much more useful to schools or even manufacturing facilities.

6.0 Conclusion

The final system demonstrated the high potential for a low-cost modification to existing 3D printers for continuous printing. The sequence of removing, bending, and brushing was sufficient to remove parts at least 1mm thick, and 20mm long. The entire user experience for uploading, slicing, visualizing, queuing was streamlined into a simple online interface while automating the tedious elements of printing. The system is no more reliable than the printer itself, but as printers improve automated removal will only become more viable. After integrating the designed storage and computer vision systems, the system could continuously print queued parts without any user input, even overnight. By expanding the hours a printer can be in use and saving time for printer operators, this could have a dramatic impact on the efficiency of 3D printers.

7.0 Works Cited

- [1] Makerbot, "Makerbot for Educators," 2011. [Online]. Available: <http://downloads.makerbot.com/support/pdf/Thing-O-Matic/Docs/Thing-O-Matic%20Build%20Instructions%20for%20Educators.pdf>. [Accessed 2020].
- [2] BlackBelt 3D BV, "BlackBelt 3D," [Online]. Available: <https://blackbelt-3d.com/>. [Accessed May 2020].
- [3] OctoPrint, "OctoPrint," [Online]. Available: <https://octoprint.org/>. [Accessed May 2020].
- [4] 3DPrinterOS, "Store | 3DPrinterOS," [Online]. Available: <https://www.3dprinter-os.com/store/>. [Accessed May 2020].
- [5] Astroprint, "AstroBox Gateway Raspberry Pi 3 Kit," [Online]. Available: <https://store.astroprint.com/collections/accesories/products/astrobox-raspberry-pi-3-kit>. [Accessed May 2020].
- [6] Prusa Research, "The Original Prusa i3 MK3S 3D Printer," [Online]. Available: <https://www.prusa3d.com/original-prusa-i3-mk3/>. [Accessed May 2020].
- [7] J. Kayne, "Prusa i3 MK3 SolidWorks," 7 December 2017. [Online]. Available: <https://grabcad.com/library/prusa-i3-mk3-solidworks-1>. [Accessed May 2020].
- [8] Pololu, "Dual MC33926 Motor Driver Carrier," [Online]. Available: <https://www.pololu.com/product/1213>. [Accessed May 2020].

8.0 Appendices

8.1 Bill of Materials

Name	Material	Size	Quantity	Individual Cost	Total Cost
Frame Parts:					
Prusa MK3 3D Printer	Various		1	\$ 750.00	\$ -
40mm square 8020	Aluminum	2 meters	1	\$ 54.50	\$ 54.50
30mm square 8020	Aluminum	280mm	1	\$ 6.00	\$ 6.00
Front Plate	Aluminum	377x200x5mm	1	\$ 23.50	\$ 23.50
Side Plates	Aluminum	120x120x5mm	2	\$ 2.50	\$ 5.00
Base Plate	Aluminum	286x182mmx5mm	1	\$ 16.20	\$ 16.20
Attachment Plates	Aluminum	180x68mmx5mm	2	\$ 3.80	\$ 7.60
Electronics Protective Cover	Acrylic	286x81x3mm	1	\$ 2.70	\$ 2.70
Brush Plate	Aluminum	292x64x5mm	1	\$ 5.80	\$ 5.80
Masonry Brush	Wood		1	\$ 4.98	\$ 4.98
1kg black PLA plastic	PLA	1kg	1	\$ 19.99	\$ 19.99
M3 x 30mm Bolt	Steel	3x30mm	2	\$ 0.09	\$ 0.18
M3 x 12mm Countersunk Bolt	Steel	3x12mm	2	\$ 0.21	\$ 0.42
M3 x 12mm Bolt	Steel	3x12mm	30	\$ 0.05	\$ 1.50
M8 x 16mm Bolt	Steel	8x12mm	67	\$ 0.26	\$ 17.42
6-32 3/8" Bolts	Steel	3.5x9.5mm	10	\$ 0.09	\$ 0.90
M8 T nut	Steel		40	\$ 0.47	\$ 18.80
M8 double T nut	Steel		12	\$ 0.86	\$ 10.32
Motion:					
ServoCity 499:1 Economy Geared Motor	Steel	25mm	1	\$ 14.99	\$ 14.99
ServoCity 170:1 Economy Geared	Steel	25mm	1	\$ 14.99	\$ 14.99

Motor					
608ZZ Bearing	Steel	8x22x7mm	3	\$ 0.40	\$ 1.20
LM8UU Linear Bearing	Steel	8x15x24mm	2	\$ 2.25	\$ 4.50
8mm Linear Slide	Hardened Steel	325mm	2	\$ 7.00	\$ 14.00
5mm Shaft	Steel	300mm	1	\$ 2.69	\$ 2.69
32T Gear	Aluminum		1	\$ 12.99	\$ 12.99
16T Gear 4mm Bore	Brass		1	\$ 7.99	\$ 7.99
5x11x4mm bearings	Steel		3	\$ 0.59	\$ 1.77
20 Tooth GT2 Pully	Aluminum		2	\$ 1.60	\$ 3.20
16 Tooth GT2 Idler	Aluminum		2	\$ 1.80	\$ 3.60
MG995 Servo Motor	Various		2	\$ 5.99	\$ 11.98
Parralax Standard Servo Motor	Various		2	\$ 10.49	\$ 20.98
5mm Shaft Clamp	Aluminum		1	\$ 5.99	\$ 5.99
4mm Shaft Clamp	Aluminum		1	\$ 5.99	\$ 5.99
Electrical:					\$ -
Pololu Dual MC33926 Motor Driver Carrier	PCB		1	\$ 29.95	\$ 29.95
Raspberry Pi	PCB		1	\$ 42.99	\$ 42.99
Potentiometer	Various		1	\$ 9.99	\$ 9.99
Limit Switch	Plastic		4	\$ 0.89	\$ 3.56
22 AWG Hookup wire	Wire	5x26.2ft	0.2	\$ 13.99	\$ 2.80
5V DC Voltage Regulator	PCB		1	\$ 12.99	\$ 12.99
Arduino Mega	PCB		1	\$ 12.99	\$ 12.99
12V Regulated Switching Power Supply	PCB		1	\$ 18.88	\$ 18.88
Gear Motor Input Board	PCB		2	\$ 0.99	\$ 1.98
3D Printed:					
Plate Tab	PLA		2	\$ -	\$ -
Plate Tab Receptacle	PLA		2	\$ -	\$ -

Plate Finger	PLA		1	\$ -	\$ -
Pivot Gear	PLA		1	\$ -	\$ -
Motor Gear	PLA		1	\$ -	\$ -
Anti-Catapult Servo Mount	PLA		2	\$ -	\$ -
Anti-Catapult Wiper	PLA		2	\$ -	\$ -
Plate Bender Servo Mount	PLA		2	\$ -	\$ -
Plate Bender Finger	PLA		2	\$ -	\$ -
Pivot Motor Mount	PLA		1	\$ -	\$ -
Pivot Motor Shaft Support Bracket	PLA		1	\$ -	\$ -
Brush Shaft Bearing Clamp	PLA		3	\$ -	\$ -
Brush Motor Mount	PLA		1	\$ -	\$ -
Brush Linear Rod Base Mount	PLA		2	\$ -	\$ -
Brush Linear Rod Top Mount	PLA		2	\$ -	\$ -
Brush LM8UU Mount	PLA		2	\$ -	\$ -
Potentiometer Bracket	PLA		1	\$ -	\$ -
				Total:	\$ 458.80

8.2 Project Assembly

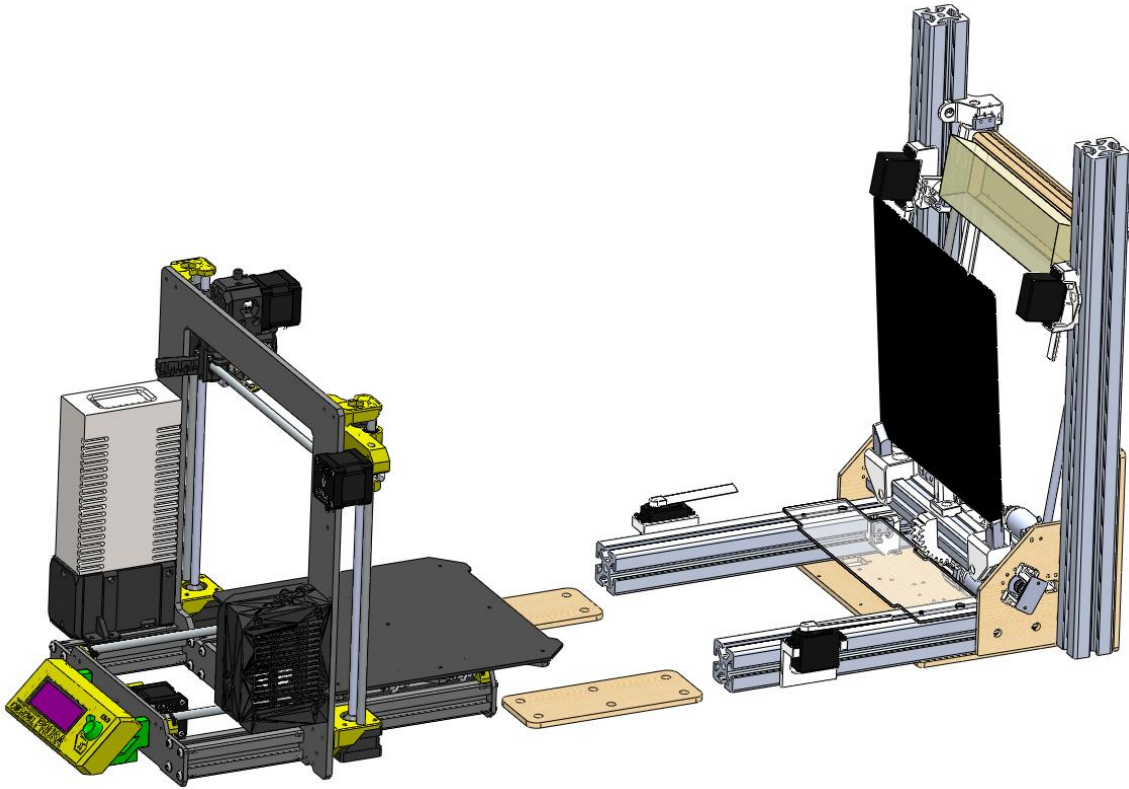


Figure 42: Attachment to Prusa i3 MK3

SolidWorks model of the Prusa i3 MK3 is from Jonathan Kayne on GrabCAD [7].

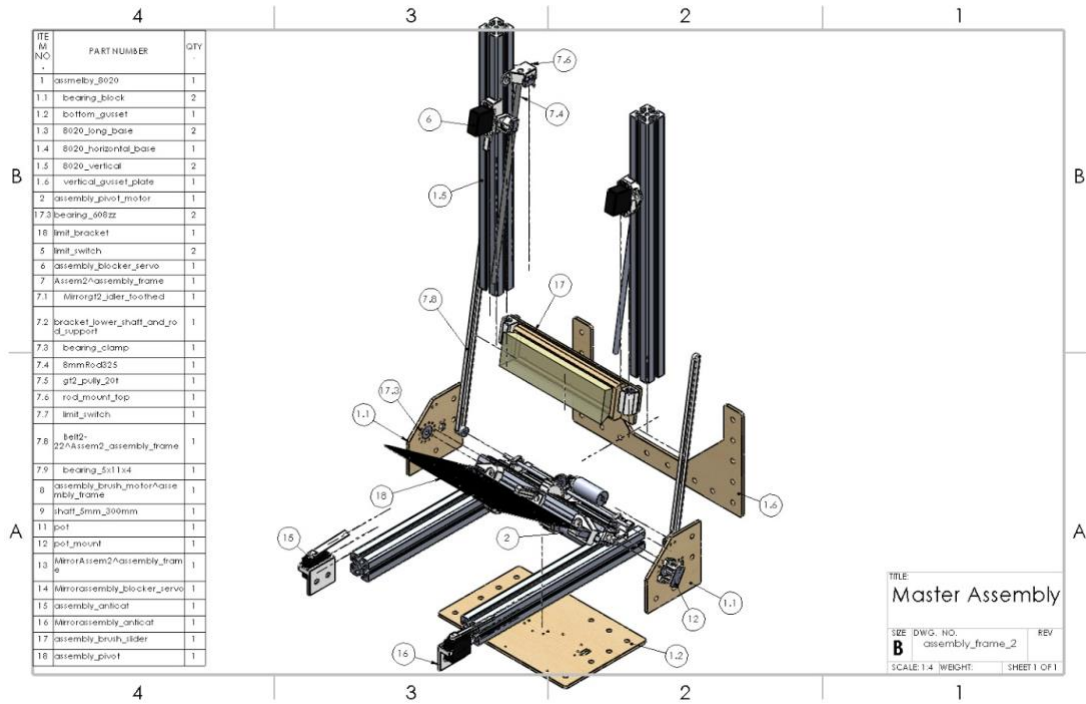


Figure 43: Full Part Removal System Exploded by Subassembly

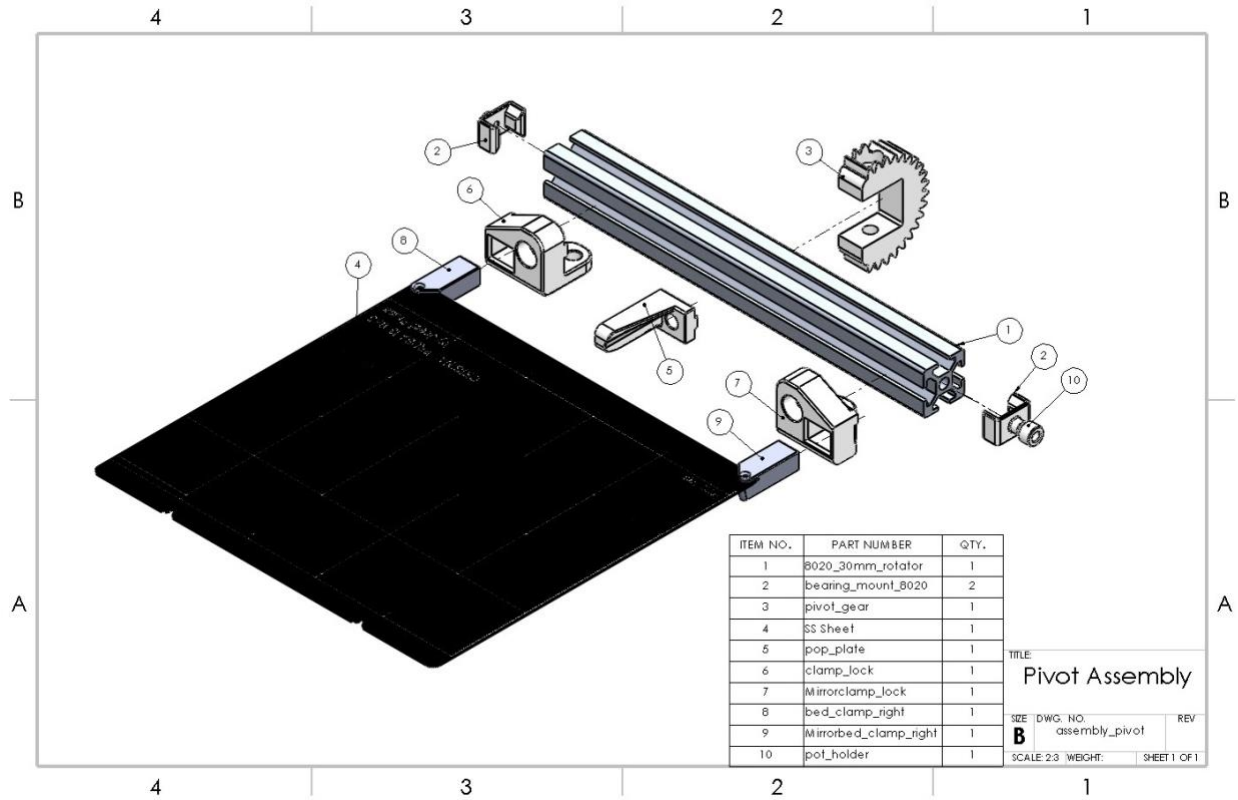


Figure 44: Attachments to Main Pivot Exploded

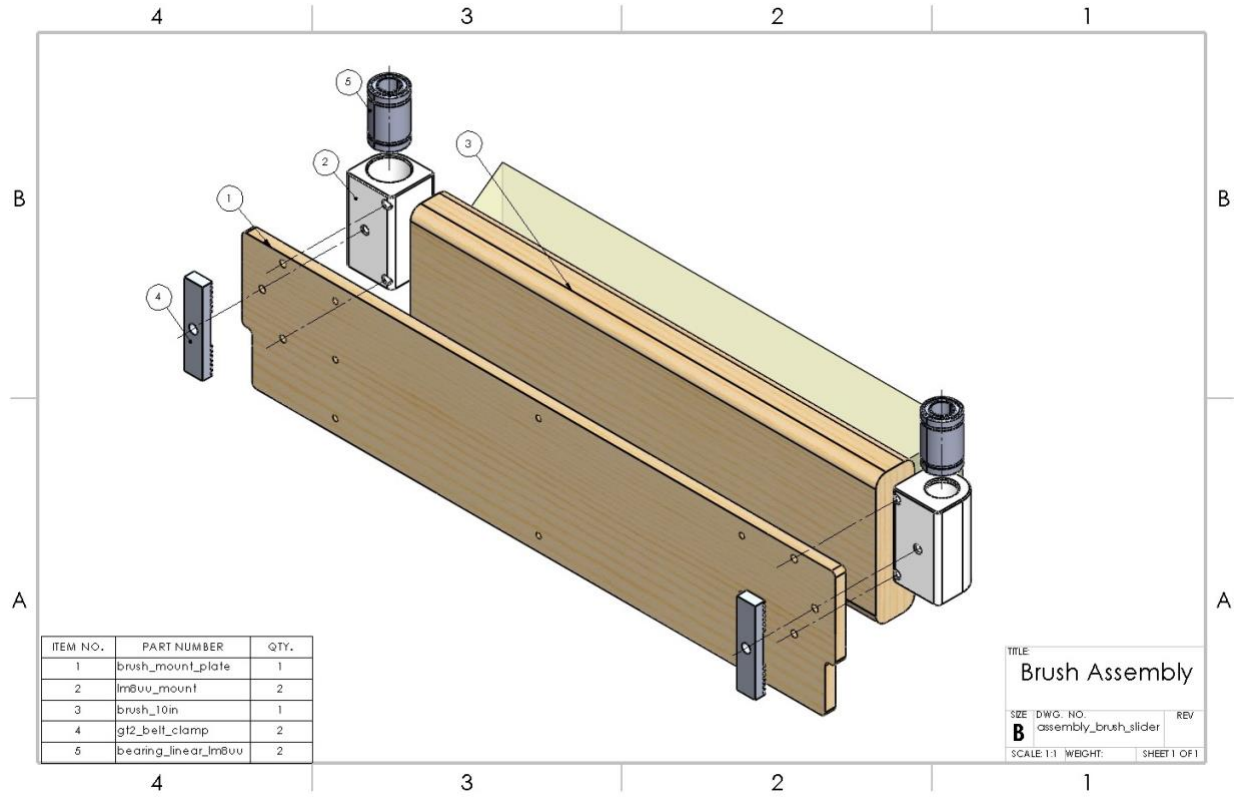


Figure 45: Brush Mount Exploded

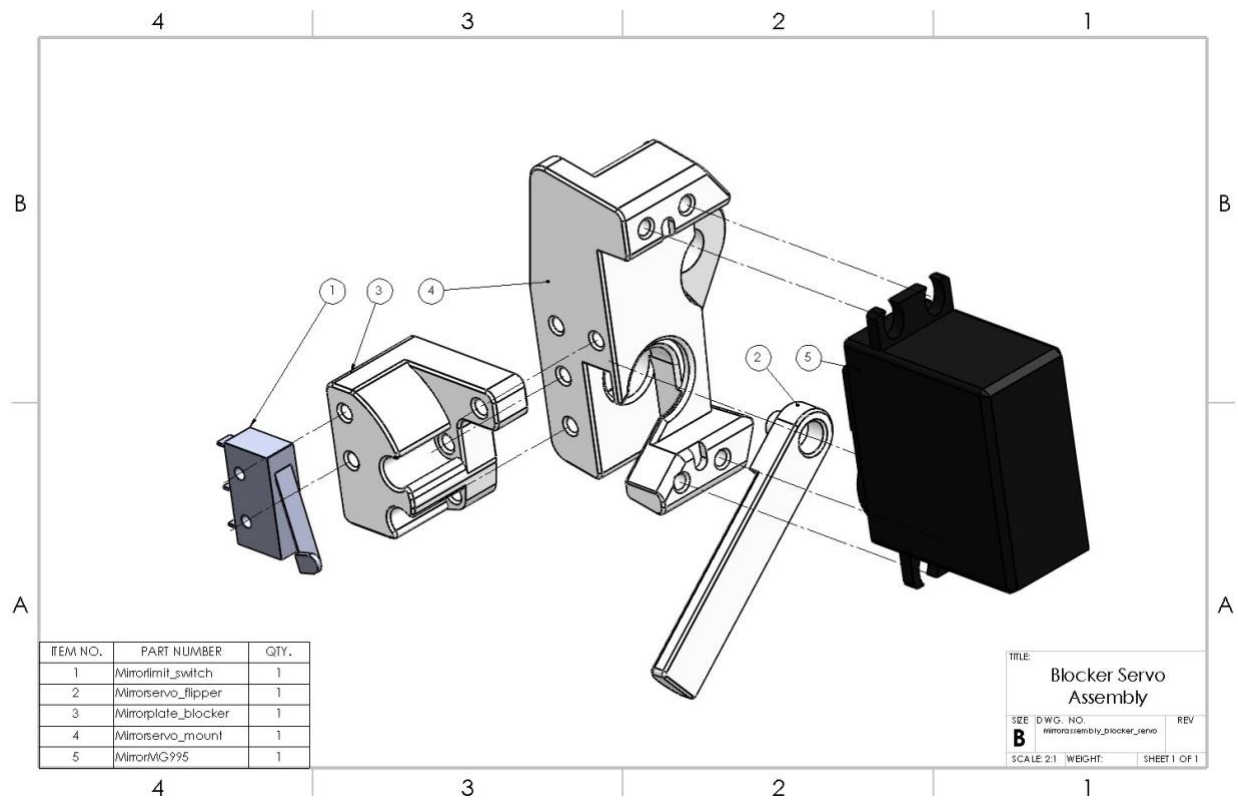


Figure 46: Bed Locking Servo Mount Exploded

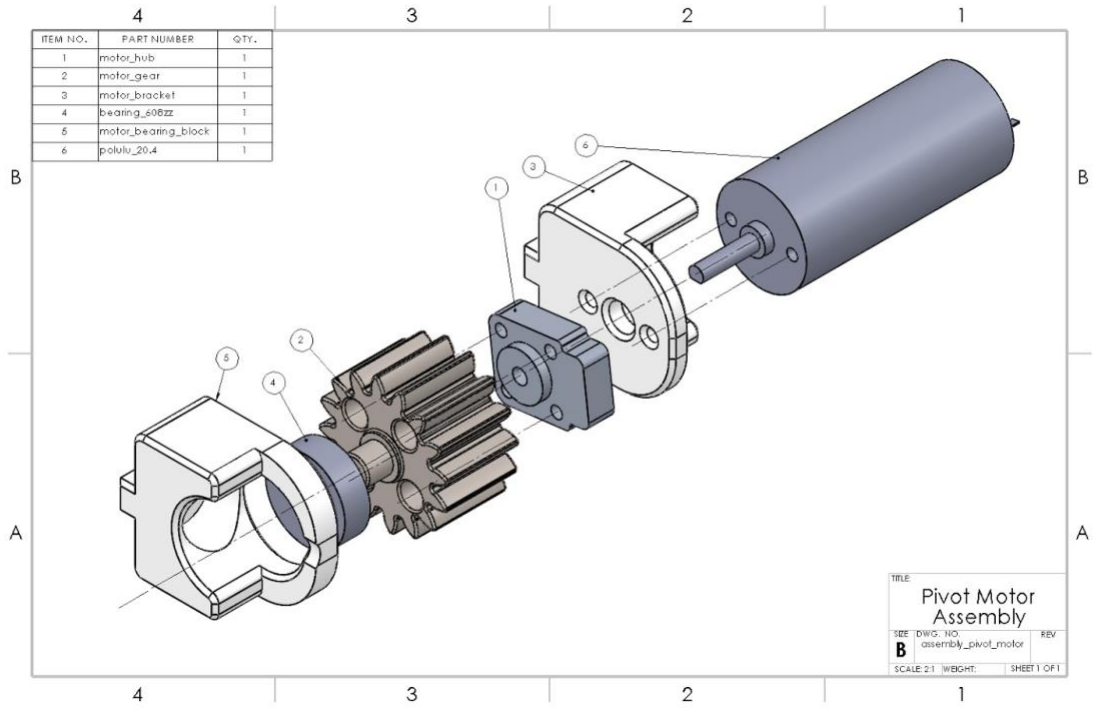


Figure 47: Bed Rotation Motor Mounting Exploded

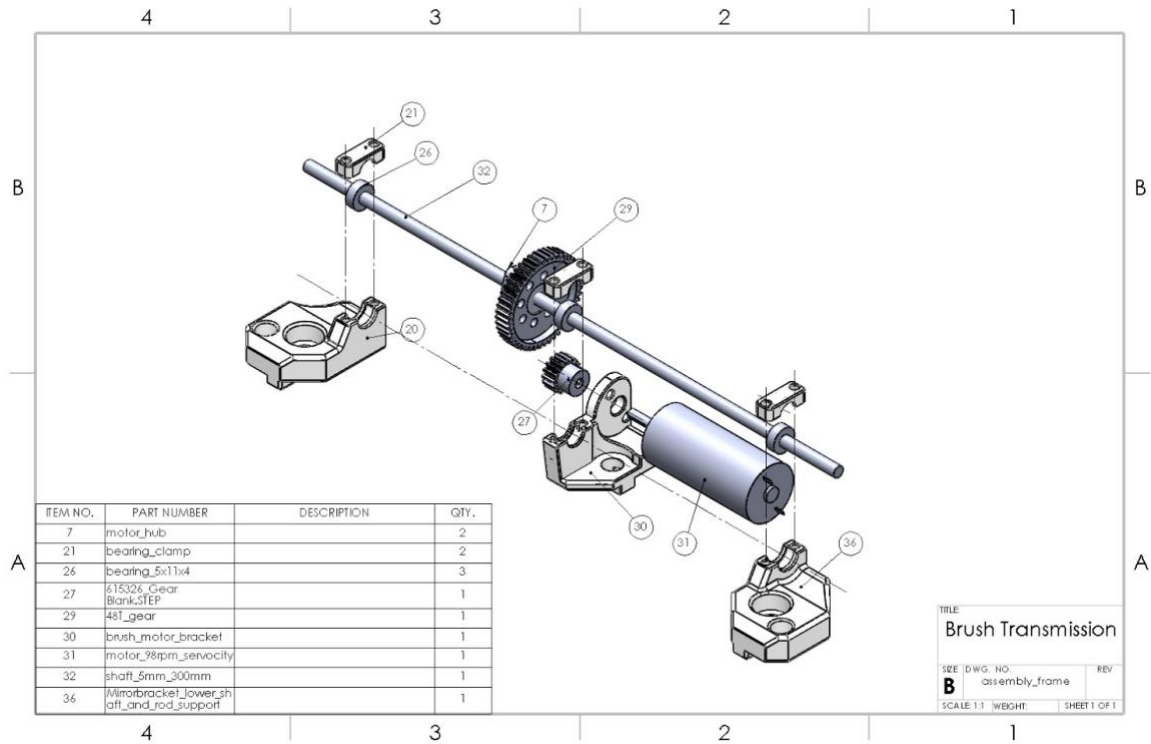


Figure 48: Brush Control Gearing Exploded

8.2 Dual MC33926 Motor Driver Carrier Pin Descriptions

PIN	Default State	Description
VIN	HIGH	This is the main 5 V to 28 V motor power supply connection, which should typically be made to the larger VIN pad. Operation from 5 V to 8 V reduces maximum current output; the device is also protected for transients up to 40 V. The smaller VIN pad can be used to distribute the VIN node to the rest of the application circuit; for lower-current applications, the pin can also be used to power the board and motors.
GND	LOW	Ground connection for logic and motor power supplies.
OUT2	HIGH	The motor output pin controlled by IN2.
OUT1	HIGH	The motor output pin controlled by IN1.
VDD	HIGH	3 V to 5 V logic supply connection. This pin is used only for the \overline{SF} pull-up and default-overriding jumpers; in the rare case where none of those features is used, VDD can be left disconnected.
IN2	HIGH	The logic input control of OUT2. PWM can be applied to this pin (typically done with both disable lines inactive).
IN1	HIGH	The logic input control of OUT1. PWM can be applied to this pin (typically done with both disable lines inactive).
PWM / $\overline{D2}$	LOW	Inverted disable input: when $\overline{D2}$ is low, OUT1 and OUT2 are set to high impedance. A $\overline{D2}$ PWM duty cycle of 70% gives a motor duty cycle of 70%. Typically, only one of the two disable pins is used, but the default is for both disable pins to be active.
PWM / D1	HIGH	Disable input: when D1 is high, OUT1 and OUT2 are set to high impedance. A D1 PWM duty cycle of 70% gives a motor duty cycle of 30%. Typically, only one of the two disable pins is used, but the default is for both disable pins to be active.
\overline{SF}	HIGH	Status flag output: an over-current (short circuit) or over-temperature event will cause \overline{SF} to be latched LOW. If either of the disable pins (D1 or $\overline{D2}$) are disabling the outputs, \overline{SF} will also be LOW. Otherwise, this pin is weakly pulled high. This allows the two \overline{SF} pins on the board to be tied together and connected to a single MCU input.
FB	LOW	The FB output provides analog current-sense feedback of approximately 525 mV per amp (only active while H-bridge is driving).
EN	LOW	Enable input: when EN is LOW, the both motor driver ICs are in a low-current sleep mode.
SLEW	LOW	Output slew rate selection input. A logical LOW results in a slow output rise time (1.5 μ s to 6 μ s). A logical HIGH selects a fast output rise time (0.2 μ s to 1.45 μ s). This pin should be set HIGH for high-frequency (over 10 kHz) PWM. This pin determines the slew rate mode for both motor driver ICs.
INV	LOW	A logical high value inverts the meaning of IN1 and IN2 for both motor drivers.

[8]