New Paltz
STATE UNIVERSITY OF NEW YORK

Division of Engineering Programs

SENIOR DESIGN PROJECT REPORT

Spring 2019

May 6th 2019

Multipurpose End of Arm Tooling

Student name	Student Major	Grade Received
NAME OMITTED	ME	?
George Dagis	CE	A
NAME OMITTED	ME	?
NAME OMITTED	ME	?
NAME OMITTED	ME	?

Advisor's name	Advisor's affiliation
Ping-Chuan Wang	SUNY New Paltz
Mike Otis	SUNY New Paltz
Aaron Phipps	MPI

Abstract

The goal of our team's senior design project was to develop a dynamic End of Arm Tooling (EOAT) that can be programmed to adapt to different sized objects. This idea comes from the need of a local company, MPI Inc., whose capabilities are currently limited by needing a separate tool for each wax object that the robotic arm is transporting. With a dynamic gripper, the company will no longer have to engineer a new tool for each wax object, and can instead use the singular tool created by our group. In the first semester, a feasibility study was conducted to develop a tool that used sensor feedback in order to determine where to grab the wax object as well as how much pressure the gripper will apply to the object. Several issues limited the feasibility of this design, including its reliance on many 3D printed parts. After deciding that the initial concept would not fulfill the requirements of the project scope, a new design was developed. The Spring 2019 semester involved developing a completely new concept that did not use sensor feedback and instead moved suction cups to pre-programmed positions in order to pick up the different wax parts. With this design, the reliance on 3D printing was also reduced, as more off-the-shelf components were incorporated. Some issues occurred with this design including binding between different components. A final design was created to fix these issues, and the prototype proved to be much more successful than the product from Fall 2018. With more time, more reliable components, and a method of communication between the tool and the robot, a final product could be developed that would meet all requirements set by the customer. This report provides useful information to the customer or any other team that may further develop this project.

TABLE OF CONTENTS

1. About This Project

This project was brought to SUNY New Paltz by MPI Inc. They requested that a new end of arm tooling (EOAT) be designed for a robot operating within an automated cell, with the objective of reducing costs.

1.1 Motivation

The EOAT Project was proposed to SUNY New Paltz by MPI Inc., a company in Poughkeepsie that designs and manufactures wax injection equipment. The project is based on improving MPI's previous gripper designs by creating a multi-purpose tooling. The end of arm tools that MPI currently uses are static, meaning each tool only has the ability to pick up only the wax pattern that it is specified for. With this project, the goal is to develop a gripper that will pick up any pattern that they currently make or will make within a specific size range.

1.2 Design Objective

The purpose of this project is to create a multi purpose gripper that can pick up an infinite amount of different wax patterns. The gripper should be able to conform to any geometry and size that the manufacturing line can output. In this case, MPI limited it to the wax patterns for a bottle opener and a flowback preventer flap. The gripper has to be able to pick those objects up without causing any permanent deformation (damage) to the parts.

1.3 Need for Project

The creation of a multi-purpose end of arm tooling is important to MPI because it would eliminate the need for the company to produce additional tooling for wax patterns that are created. Currently, the creation of a new end of arm tool costs the company anywhere from $3,000 to $15,000. With the creation of one universal tool, this cost would be reduced significantly as well as the time used to create each tool. Furthermore, by having one gripper in the automated cell for multiple parts instead of one gripper per potential part, some of the cells could be created without the gripper rack that holds all the unused tools. This would further decrease the time and money needed to produce the cells for MPI. This space might be used more efficiently, e.g. by expanding the space around the dye through relocating the silicone sprayer.

1.4 Project Scope

MPI desires a gripper that can pick up two objects having different geometries from a die. One of the required objects can be viewed in Figure 1. This object consists of four wax flaps with a total width of 200mm. The other object is a wax bottle opener with width of which can be shown in Figure 2. The gripper itself must be capable of adjusting to these two sizes and pick them up without dropping or otherwise damaging them.

The temperature of the wax is also a constraint. Because the wax will still be warm when the gripper comes in contact with it, the wax is still relatively soft. This knowledge drives the design of the gripper to be made from a soft material where it is in contact with the wax so no deformation occurs.

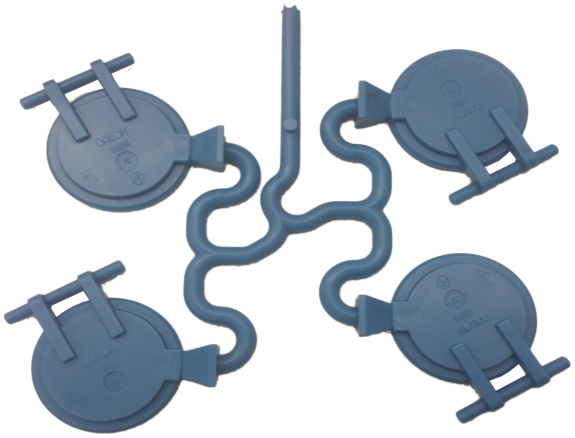


Figure 1: Wax Flaps Attached to Sprue

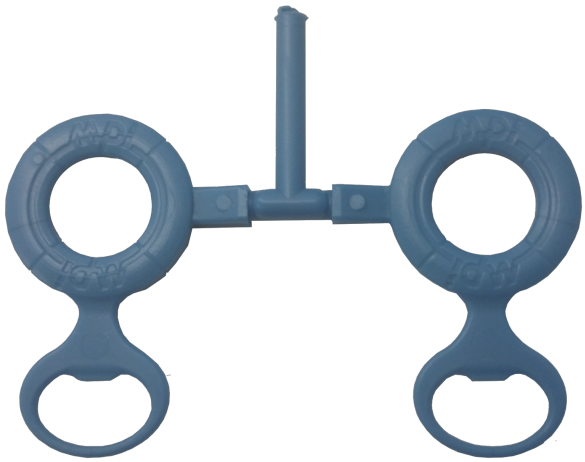


Figure 2: Wax Bottle Opener Attached to Sprue

Since the gripper must be able to work with the Machines produces by MPI, The MPI Internal standards will be followed for throughout the design of this gripper[1].

The properties of the wax used will be determined through preliminary testing.

2. Functional Description Of Design

The diagram below illustrates the flowchart which is controlled using python code on a Raspberry Pi 3 Model B+ microcontroller. The program often inquires from the user if they want to continue their current process as a confirmation and to avoid mistakes. Most of these confirmations are not shown within the diagram in order to avoid redundancy.

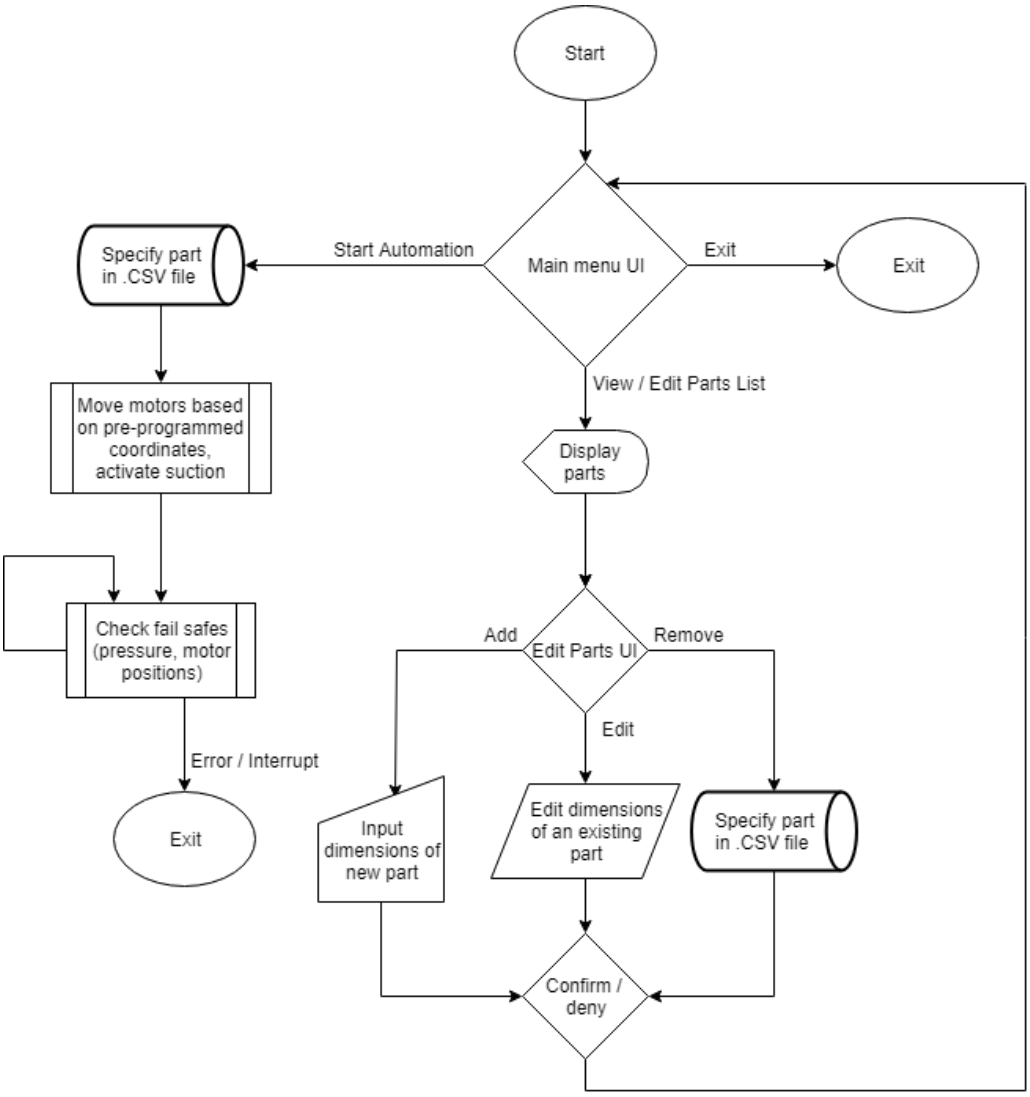


Figure 3: Logic Flowchart

The programs' initial state keeps all components stationary and immediately moves to first decision making block, the main menu UI. The UI displays a text menu for easy navigation. This UI serves as the central hub for the user, allowing them to either start automation on an existing part, view or edit a parts list, or exiting the program.

If the start automation option is selected, the program accesses a text file which is responsible for holding data regarding each unique wax pattern. This data includes a name as well as the parts' positioning information. The positioning information that is responsible for directing the stepper motors are defined by 3 distances (given in inches). That is the distance, x, corresponds to the distance between the two leftmost brackets and the two rightmost brackets. The other distances found in the files, y1 and y2, correspond to the distances between the top brackets and the bottom brackets. Since the wax patterns are not always symmetrical, two y values are needed.

After retrieving the positional data for the suction cups, the stepper motors are then able to step the appropriate amount of steps. The conversion from distance to steps is fairly simple and the derived formula for finding the number of steps to achieve a distance is shown below:

$$steps = distance\ (inches) * pitch * steps\ /\ revolution$$

The threaded rod has 20 threads/inch, and the stepper motor has a step angle of 1.8°, or 200 steps/revolution. Finally, we must factor in the optical rotary encoder which has 300 cycles/rotation rather than 200. These constants can be multiplied together in order to construct the final formula is shown below:

$$steps = distance\ (inches) * 6000$$

In an ideal system, all fail-safes such as pressure sensors and motor positioning is checked in order to avoid issues. The pressure must be checked in order to ensure a part is being gripped when it is being transported and otherwise there must be no pressure. It would also be important to be able to gauge pressure relative to the suction cups in order to avoid running an automation cycle despite a part being dropped or never picked up to begin with. This check could also ensure that a part has been delivered to the right location in the cell. In the case of encoder failure, a limit switch could be implemented in order to prevent high-speed collisions with the frame.

3. Preliminary Designs And Results

3.1 Fall 2018

The initial design phase started out by brainstorming a multitude of ways/designs for a tool to pick up wax and transport it without deformation. Each member of the team was tasked with delivering 3 designs. To select the appropriate design, a decision matrix in conjunction with a pairwise comparison chart was created. The pairwise comparison chart(shown in Table A-1 in the appendix) helped narrow down on the main focuses for the overall design by weighing each criteria that would be assessed in the decision matrix which ultimately selected the design(shown in Table A-2 in the appendix). Significant emphasis was placed on leveraging 3D printing, due to its availability on campus, it being suggested in the project proposal, and all members of the group being experienced with the technology. It was also proposed that testing and demonstrating a design that used suction would be more difficult than a design that used grippers. Since the objective was to quickly build and test a prototype using readily-available tools, the decision was made to use grippers rather than suction cups in the chosen design.

Having determined that the design would implement a gripper, it was decided that a pressure sensor would be used to determine the force being applied to the wax pattern during transport. Knowing this would allow the tool to avoid deforming the wax by adjusting the gripper appropriately. To determine acceptable levels of force, testing was performed on the wax. Experiments were conducted in November of 2018 to determine the elastic modulus and elastic limit of the wax used in MPIs automated cell. The properties of the wax change rapidly as it cools; as such, Stress vs Strain curves were created for temperatures at which MPI would most likely extract the wax from the die.

This experiment was performed by utilizing an Instron testing apparatus to apply a load to a sample of cooling wax(set up shown in Figure A-2). A tool was designed to be connected to the Instron with a tip that had a known area as shown in Figure A-1. With this relationship, the equation shown below was used to determine the maximum stress of the wax before deformation:

$$Pressure = \frac{Force}{Area}$$

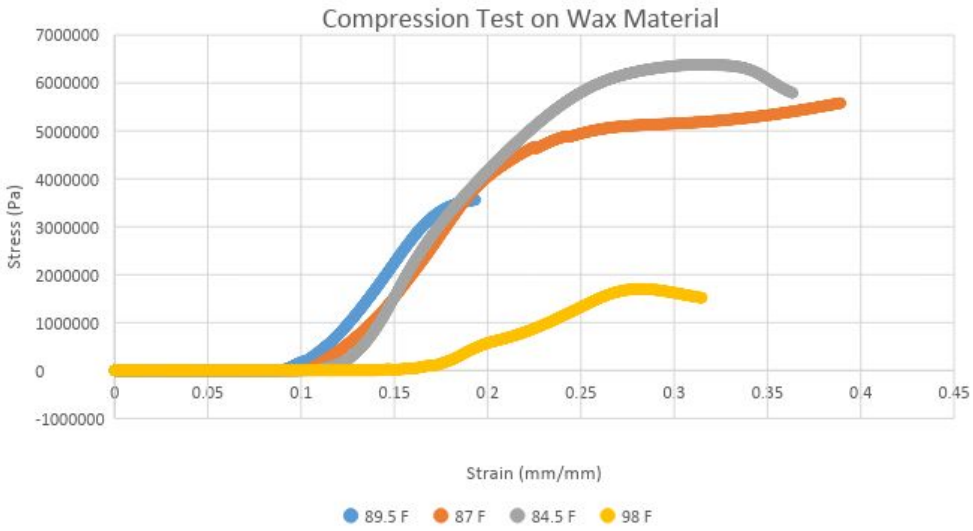


Figure 4: Stress-Strain-Curve Generated of Wax at Various Temperatures

The data in Figure 4 suggests that the wax is much more likely to deform at higher temperatures than at lower temperatures, as expected. However, it was realized after testing that the tool used to apply a load to the wax also experienced deformation. This introduced significant error to the data, making it useless for determining specific material properties.

The EOAT that was designed incorporated four independent grippers that would move according to a distance sensor reading to grab the wax pattern while regulating the pressure applied by the gripper with a seperate sensor. To get the grippers to move in a linear direction in an easy way, the use of rack and pinion gears was investigated. This lead to a design shown in the Figures below(Figure 6 and Figure 6). To alleviate the friction generated from the plastic on plastic interaction between the arm and the rail, bearings were installed on specific locations that would allow the plastic on the arm to remain untouched by the plastic on the rail. This proved to be helpful but because of the inconsistent tolerancing of the 3D printing method, friction was still a major limiting factor of the arm's linear motion.

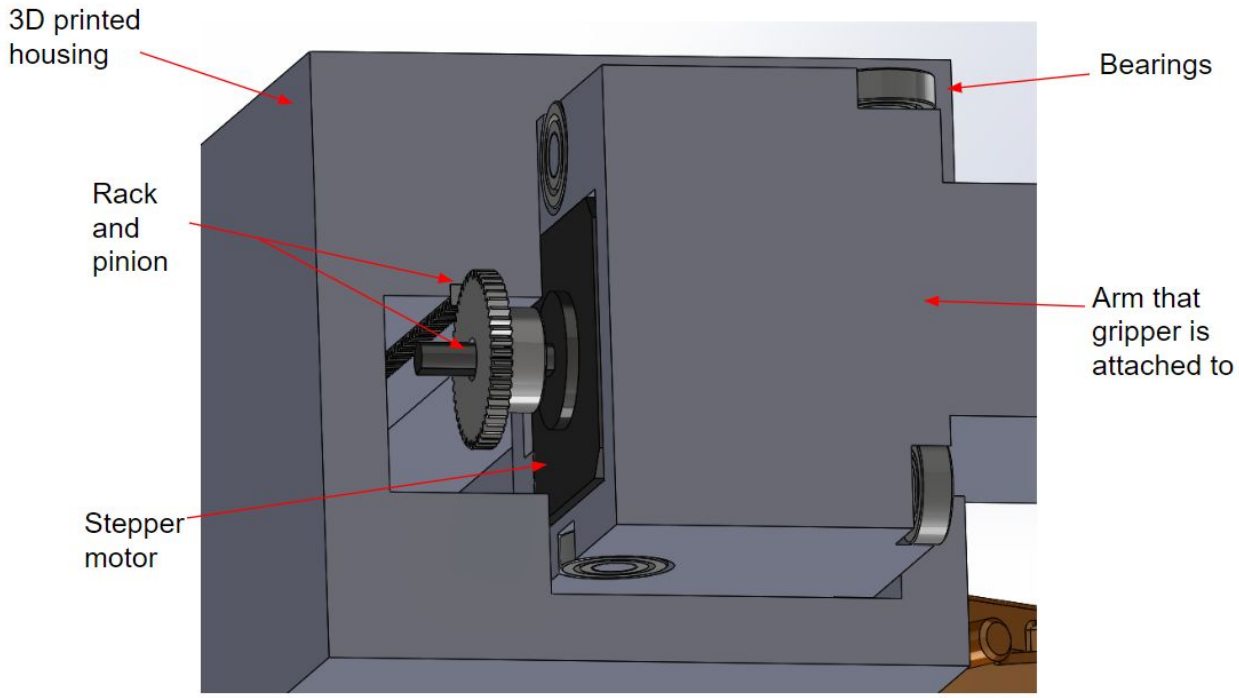


Figure 5: Features of the Initial Design Demonstrated by SolidWorks

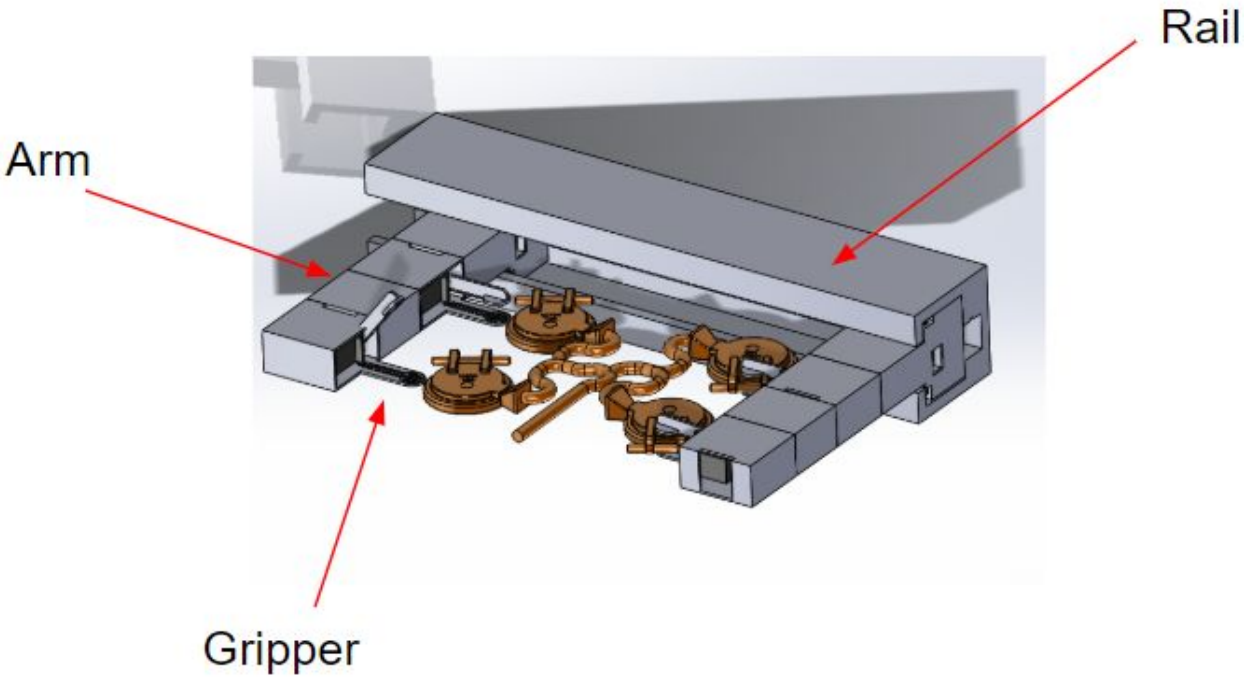


Figure 6: SolidWorks Model of the Initial Design

A rack is placed on the inside of the 3D printed housing to mesh with a gear that is attached to a stepper motor. Depending on the reading from the distance sensor, the arm will move back and forth along the rail. The gripper is comprised of a long piece of plastic attached to a servo motor. The servo motor would then in turn push down on the platform of the arm allowing the object to be gripped.

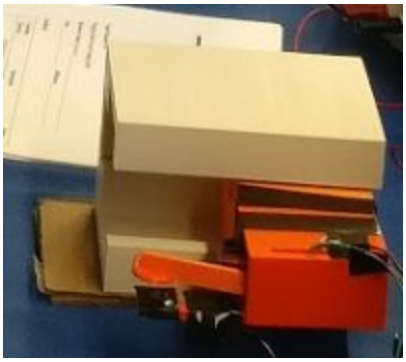


Figure 7: Initial Design ¼ Model Prototype

Shown in Figure 7, is a 3D printed section of the initial design with the pressure sensor mounted on the gripper and the distance sensor mounted below the gripping platform.

The design created in the first semester contained a number of shortcomings. Excessive reliance on 3D printing resulted in a number of these problems. First, it resulted in an oversized prototype that would not fit in MPI's automated robotic cell. Also, limitations in the accuracy of the 3D printers resulted in poor mating between the rack and pinion. Furthermore, the quality of individual prints is unreliable, which would often necessitate reprinting parts. Lastly, the surface finish of 3D printed parts generated too much friction when moving the arms along the rail. Other shortcomings included delayed response from the sensors and a lack of precise control over the force applied by the gripper.

3.2 Spring 2019

Realizing the shortcomings of the previous design, a decision was made to start from scratch with a new general approach. New objectives included reducing the overall size of the part and putting much more consideration into how the prototype would be built. It was determined that use of 3D printing should be limited to components that needed to be custom-made and would still perform effectively despite being 3D printed. Making use of off-the-shelf components was favored, as it was expected that doing so would accelerate the creation of a new prototype and decrease costs.

MPI is currently using suction cups with a majority of their end of arm tooling solutions. The suctions cups have the advantage of being gentle on the wax parts as to not permanently deform the material. With those factors in mind, the new design switched to suction cups.

To get the suction cups to move independently of each other, an idea using threaded rods and smooth rods was developed. The brackets holding the suction cups would have the corresponding nut on the inside to allow the suction cup to change locations. To keep this idea simple, one bracket would remain stationary to not interfere with the other bracket. An alternative idea is to have the motors attached to the bracket itself but this seemed cumbersome and would add excess weight on the bracket.

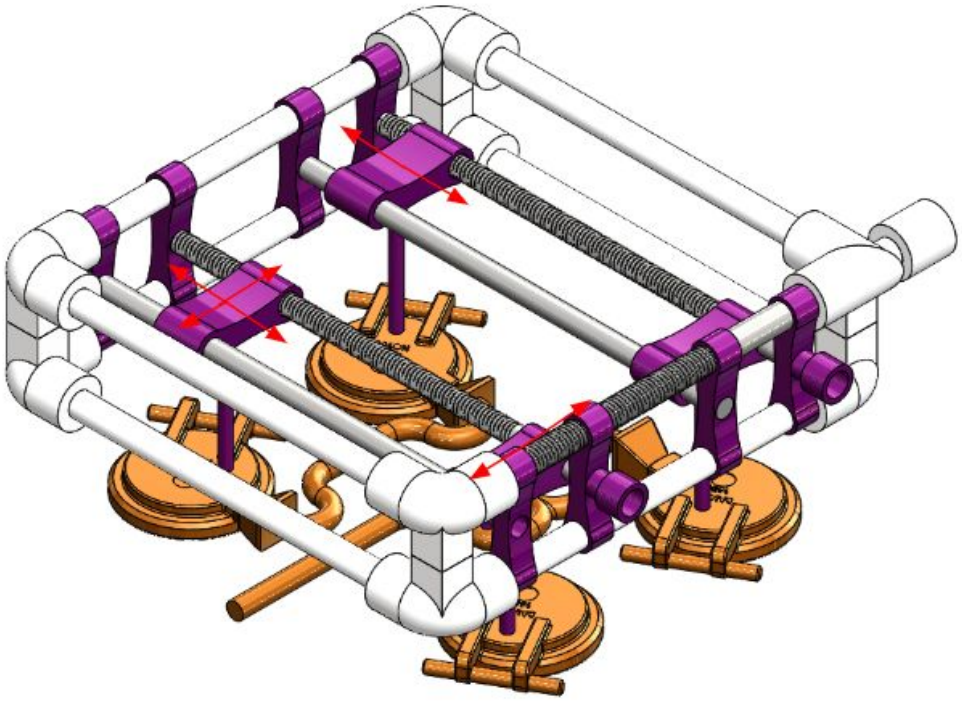


Figure 8: First Iteration of Spring 2019 Design Indicating the Movement Possible by Each Bracket

As shown in Figure 8, the allowable direction of each brackets movement is shown indicated by the red arrows. One of the brackets remained completely stationary. The stationary brackets would have a bearing between the point of contact from the threaded rod to the bracket, keeping the bracket in place. A stepper motor would apply a torque to the threaded rod causing it to rotate and ultimately move the bracket with the nut along it. The suction cups do not change in height because the robotic arm that it is attached to will control this direction.

4. Final Design And Results

4.1 Fixes to Second Iteration

The initial iteration of the design created in the Spring semester had major binding issues. The first mode of binding occurred in the vertical plane between the top and bottom of each outer bracket. The second mode of binding occurred in the horizontal plane between two brackets on opposing sides of the frame. These binding methods were investigated by partially assembling the model and turning the threaded rod by hand to determine when and where the 3D printed brackets were binding with the PVC piping. Spacers were investigated between brackets as well as adding tension with an elastic band between brackets. Both testing methods improved the binding. To alleviate both modes of binding with a more permanent and aesthetically pleasing solution, the brackets that are sliding on the PVC were widened, increasing the area of contact between the two materials. A second threaded rod was also added to alleviate the binding of opposing brackets.

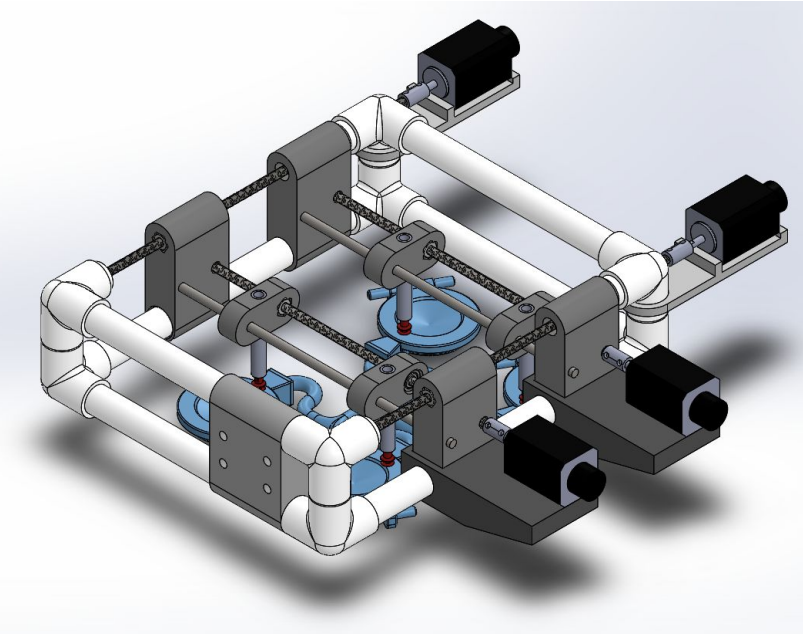


Figure 9: Final Design

The decision to add an additional threaded rod and to widen the brackets was guided by researching the binding principle and developing the relationship below in Figure 10. This relationship states that the ratio of the distance between the brackets (D) and the size of the brackets (L) must be less than or equal to the reciprocal of two times the coefficient of friction between the materials that are interacting. A graph of this relationship is seen below, where any point within the yellow area represents a theoretical point where no binding occurs.

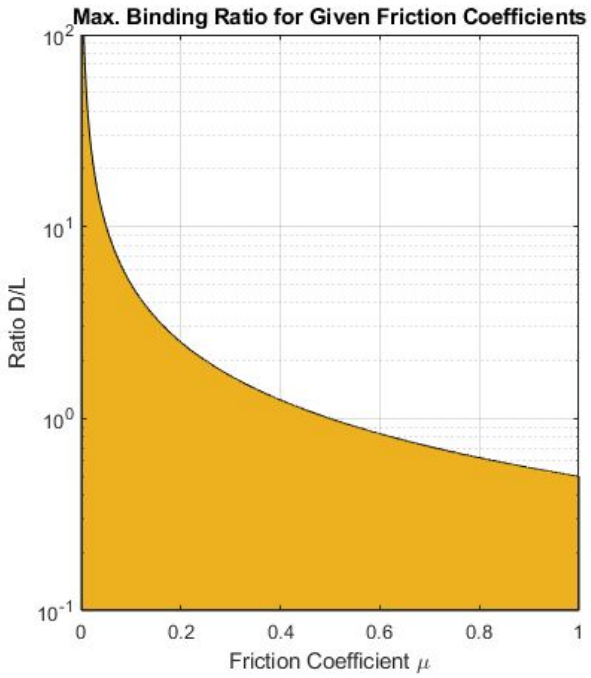


Figure 10: D/L vs μ

Calculations for this relationship can be found in Figure A-3. It should be noted that these calculations make significant simplifications, such as centering the force applied to the bracket and treating applied forces as point loads.

4.2 Raspberry Pi Programming

After assembling the final design, the feedback from the encoders was tested to determine whether or not the motors would achieve the number of steps needed to move each of the suction cups the correct amount.

In order to make sure that both motors that moved the outer brackets were synchronous, a program would need to be developed that would stop one motor is steps were missed and allow the other to catch up. Since this code was not developed prior to the completion of the project, the brackets experienced major binding when either of the stepper motors missed a significant amount of steps. This binding would bring the movement of the brackets to a sudden halt and

eliminated the possibility of displaying the functionality of the tool's dynamic capabilities. With the proper changes in the programming, however, the relocation of each of the suction cups would involve less binding.

4.3 Results

Although the final design and prototype had issues, the overall objective was achieved much more more closely than the concept from the Fall 2018 semester.

The encoders proved to be very useful and almost necessary in being able to transport the suction cups with any level of accuracy. If the stepper motor is not performing under optimal electrical conditions, it may skip steps. With the encoders, the specific number of steps that the shaft of the motor has rotated is provided as feedback, so the actual amount of rotation is recorded in a failsafe manner.

Within the testing phase of the final design, however, a few of the stepper motors exhibited unexpected characteristics and inconsistent performance. One of the motors would randomly change direction and then at another point would not reverse direction when the code specified it to. A possible explanation for these issues is "noise" in the system and it could be fixed using an opto-isolator. An opto-isolator transfers electrical signal between two components using light. With this component in place, electrical noise would not be able to affect the motors and would increase the likelihood of the motors acting without issue.

While untested, it is assumed that the final design would be unable to place parts into a slotted tray. Not only is the tool too large to flawlessly maneuver in MPI's robotic cell, but the need to have simultaneous vacuum on all four suction cups limits the ability to place the wax parts in a single orientation.

5. Recommendations For Future Work

5.1 Size

One of the biggest points to improve on is the size of the gripper. The space in the automated cell's die is very limited. Due to the way the cell is constructed, the dye is in one of the corners. This restricts the gripper's space on two sides. Additionally, the third side of the dye is blocked by a shield. As stated by MPI's Vice President of Sales and Marketing and one of MPI's gripper designer, size is the biggest constraint for grippers. The automated cell is overall fairly limited in size. Many of the steps that the gripper is involved in deal with size restriction from various different directions.

To combat this issue, another iteration of the gripper can be built from materials that combine a better strength and rigidity while simultaneously greatly reducing the overall dimensions. A custom fabricated aluminum frame would certainly improve this issue. This will require more sophisticated fabrication methods and thus cost more than what is realistic on a Senior Design budget.

5.2 Improving Suction

Although the suction proved to be sufficient for the project as needed, improving it would make the gripper more "future-proof". It would ensure that the gripper can pick up heavier objects with ease, as well as picking up structure that poses more difficult shapes and surface textures. This would also improve the safety during steps like cutting the sprues off or tray placement.

5.3 Cutting Off Sprues

This big step in the overall process is needed as a precursor for the tray placement. Cutting the sprues separates the big pieces into its individual components. It also frees up the gates' bottoms so that the parts can be placed gate down.

With this gripper, the sprue will be cut off with three overall steps. First, the sprue needs to be cut in half. This helps alleviating the load on each of the parts that remain attached to the sprue when others get cut lose, because the sprue is only supported by the parts that it is attached to. By cutting it in half, it will reduce that load on the parts by half. This can be seen in Figure 11. Next, the sprue has to be cut off every part's gate. This should be done in such a pattern that the overall torque caused by the hanging sprue on the part that it is still attached to is minimized.

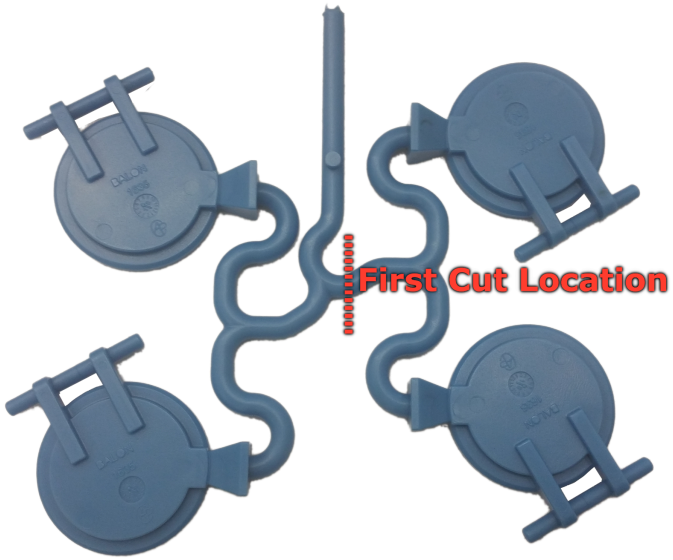


Figure 11: Cutting Position

5.4 Valves

Implement individual valves for each of the suction cups. This will enable the gripper to pick up and set down any configuration of individual parts, whether it is one, two, three, or all four. That way, the gripper will be more universal, in that it can adjust to different configurations with less individual parts after the separation process than the two assigned ones.

5.5 Tray Placement

The tray placement can be implemented by allowing the robot and gripper to follow an algorithm that will be laid out in detail with images that can be found in the appendix.

Unlike placing the part in the foam tray, which means just simply placing them the same, unaltered way the sprue has to be removed and the gates need to face down.

After the sprue has been cut off, the part then needs to be placed down for an intermediate regripping step. This should not cause any increase in cycle time, as the wax cooling is usually the longest of all steps. The gripper will then adjust in such a way that the suction cups row will move to the outermost position in terms of left-right-distribution. This can be seen in Figure 12. The robot will have to move the gripper left and right such that the left row will pick up the right parts and vice versa. This causes the parts to be attached to the gripper with the gates facing outward (Figure 13). Next, the robot will turn the gripper upright on one of the sides and place the parts in the tray. Turning the gripper upside down will face the other set of gates down, and the last two parts can be placed in the tray.

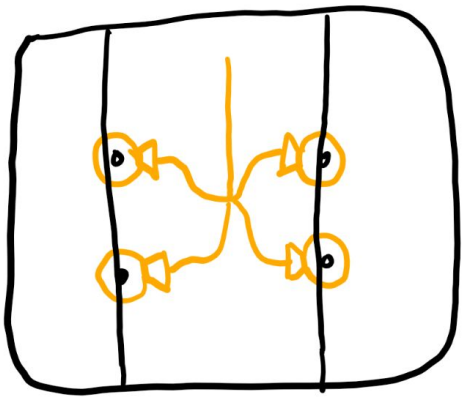


Figure 12: Before Cutting the Runner Off

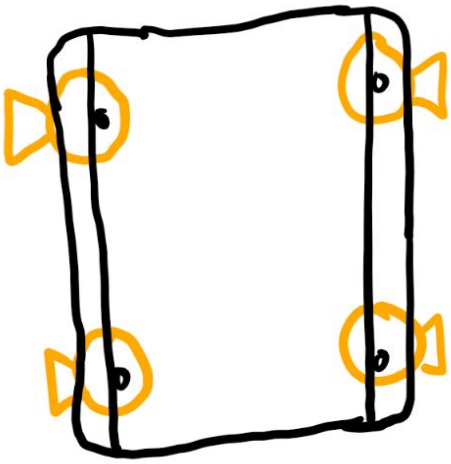


Figure 13: Regripped Configuration With Gates Facing Out

5.6 Establishing Communications With The Robot

In order for MPI to implement the tool into their automated injection cell, there must be a system developed to establish communication between the robotic arm and tool. Communication is especially important for the more complex processes such as turning off a singular suction cup or moving the suction cups in order to incorporate specific tray placement.

MPI currently uses an Omron CJ2 programmable logic controller (PLC) using ladder logic in order to communicate information. This PLC communicates via Ethernet using the EtherNet/IP Data Link function. Eventually the logic within the Pi could be transferred over to the control system at MPI or to a more adept and industrial microcontroller such as a RevPi connect or Compulab IoT Industrial Gateway. These microcontrollers have been tested in industry conditions and have passed tests to ensure their quality.

5.7 Orientation/gripper attaching to robot/gripper rack

The current way of attaching the gripper to the robot causes an orientation problem such that the gripper cannot be hung on the gripper rack in the automated cell by the robot. Usually, grippers can all be stored on a rack in the automated cell. Whenever the dye is changed for a new cast, or when a specific part needs multiple different grippers for all handling steps within the overall process, the robot takes a gripper from the rack or places one on it. This will allow for the overall process to run more autonomously without any human interference in the cell. This gripper is unfortunately in the current state not able to be hung on the rack.

To remedy this problem, another metal connector (the aluminum piece marked on figure 14) that connect the gripper to the orange base plate need to be manufactured. This part can be as custom made as needed in shape and size.

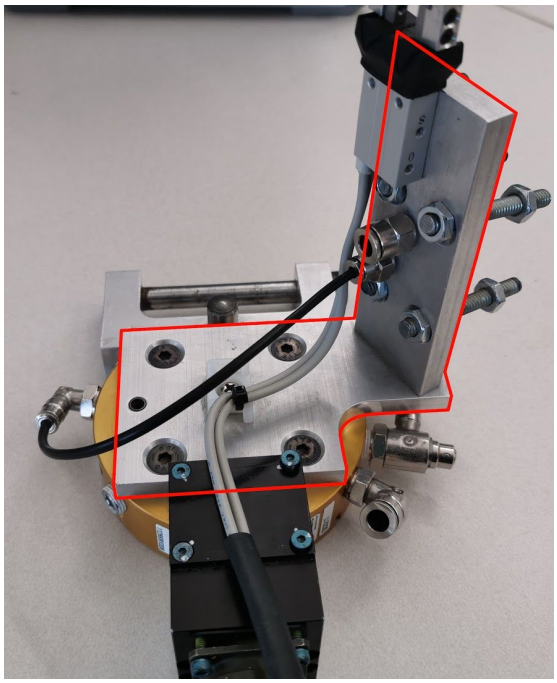


Figure 14: Metal Bracket to Be Remachined

Another way to look at it: when the gripper has undergone the above mentioned improvements, it will be universal enough that no other grippers are required, thus rendering the feature of placing it in the gripper rack useless, because it will not ever need to be changed.

5.8 Adjusting Speed

The current setup uses easily available parts from local hardware stores. This results in the use of threaded rods which have a relatively high pitch of 20 threads per inch. The initial design only required sizing adjustments when switching to different parts. If the suggested improvement for the tray placement (5.5) is implemented, the suction cups need to be able to move a lot faster from one position to another. This can be achieved by two different adjustments. Replacing the existing threaded rods with different rods that are dedicated for a worm-gear-like power transmission have a significantly lower pitch. This means that with the same amount of revolution the brackets would travel further.

Alternatively, this problem can be solved by using different motors that can offer greater rotational speeds. However, this might introduce more problems related to acceleration and deceleration of the sliders due to the motors high rpm. A combination of both of these solutions might prove to be the best option.

6. Lessons Learned

6.1 Research

The first major lesson learned over the course of this project is the importance of gathering as much information relevant to the project as early in the project development as possible. Doing so allows the team to create more informed design decisions and prevent setbacks that could have been avoided by being better informed.

Furthermore, it is important to research what others have done to solve similar problems. If an acceptable solution has already been found for some element of the, it should be used instead of spending time developing one's own solution. This is especially true if the solution is commercially available and within the project's budget.

In the semester of Fall 2018, a design was chosen based on the idea of being able to create a feasible prototype from that concept. If, before the initial design had been chosen, the team had done more research and had asked more detail oriented questions to the project stakeholder, it would have been realized much sooner that the initial concept would not fulfill the scope of the project entirely.

6.2 Project Management

To more effectively conduct a complex group project such as this one, tasks should be split up according to strengths of the individuals in the groups. In the first semester, a majority of the group would attempt to tackle a single problem together. This proved to be very ineffective for a number of reasons. Organizing a meeting between five people was difficult and limited the time that could be spent working. Making decisions required time-consuming deliberation. The group would lose sight of the project in its entirety and instead work on the singular task until it was completed.

In the second semester, the mechanical engineering problems were broken up into groups of two. This improved the turnaround time of specific project tasks while still giving team members collaborative engineering opportunities.

6.3 Tolerances

A problem that has occurred both semesters were issues regarding tolerances and the tolerance stacking. Although the 3D printers published tolerances are 0.5mm that has proven to not be reliable. Whenever any parts were chosen/designed that needed to fit together, that were not already pre-designed to be paired up (threaded rod and nuts), the tolerances added prove to be

cumbersome. In most cases, the parts fit was too lose. This caused issues such as the suction cup pipes being too big for the actual vacuum lines, or bearings and nut falling out of the brackets.

6.4 No Need to Reinvent the Wheel

For the first design, there was a big push to use 3D printing for as many parts as possible. The aforementioned 3D printing related problems carried through. The parts would not match up well because of inaccuracies in the printing process, as well as, tolerances. Also, despite HVAMC's greatly reduced printing rates for students, printing the parts still proved to be a very costly undertaking.

For the second design, the process was greatly reshaped. This time around, there was a push to use as many off-the-shelf parts as possible. Utilize the fact that someone else has already put all the research and manufacturing efforts into making a product. The frame was entirely built from PVC piping. This material was easy to be manipulated and the entire frame was only a fraction of the cost of the old 3D printed frame. The same goes for the threaded rods and nuts. Of course, there is still some play, but these parts have been manufactured to fit together well.

6.5 The Customer's Knowledge

When tasked with solving a problem, it is key to investigate the constraints. In the case of the end of arm tooling, there were some issues that did not become apparent until further down the road. Ask the customer as many questions as possible, even if they may seem too obvious. How much space is there in the dye cavity? Is this space restricted in other way? How big is the robot's wrist?

Furthermore, see if the customer has already thoroughly exhausted a certain route? Make use of that knowledge. Not only is it good to know that this solution (allegedly) does not work, but also why? This can sometimes save a lot of time and work by not going in a wrong direction, or maybe by seeing where the customer went wrong, and how it could be fixed. A fresh set of eyes can sometimes see things that are hidden in plain sight.

7. Conclusions

The tooling developed in the first semester was inadequate because of sizing and frictional concerns inherent to the 3D printing method as well as inconsistencies of the sensors that were incorporated. Although the first iteration developed in the second semester had issues of its own, solutions were created to further develop the product into its final form. Even though the final product will need further development to meet all requirements set by the customer, the research and testing done by the team provides useful information to MPI and any other individuals that may be interested in continuing the project. Specifically, any other team that may further develop this project may find it useful to know which concepts were successful and why others were abandoned.

8. Bibliography

[1] MPI Inc., "MPI Automated Pattern Assembly System 20-14 (APAS 20-14) Tooling Design Reference," MPI Inc., Poughkeepsie, 2018.

9. Different Disciplines Used In This Project

The first column of Tables 9.1, 9.2 and 9.3 show that different disciplines used in this project. More specifically, the second column shows the course(s) within the disciplines whose learning outcomes were used in the project:

Table 9.1: Electrical Engineering (EE) Disciplines and Courses within discipline in the EE program

Discipline	Course in the discipline	Check if used
D1) Computers	EGC251 C/C++ Programming	X
	EGE331 Computer Simulation (MATLAB)	
	EGC331/EGC332 Microprocessors + Lab	X
D2) Analog Electronics	EGE200/EGE201 Circuit Analysis + Lab	X
	EGE320/EGE322 Electronics I + Lab	X
	EGE321/EGE323 Electronics II + Lab	
D1) Digital Systems	EGE311 Signals and Systems	
	EGE416 Control Systems	
	EGE417 Digital Control Systems	
	EGE412 Communication Systems Theory	
	EGE493 Applied Digital Signal Processing	
D4) Electromagnetism	EGE340 Applied Electromagnetics	
	EGE445 Antenna Systems	
	EGE493 Intro to MEMS	
D5) Energy Systems	EGE350/EGE351 Electric Energy Systems + Lab	
	EGE452 Electric Power Systems	

Table 9.2: Computer Engineering (CE) Disciplines and Courses within Discipline in the CE program

Discipline	Course in the discipline	Check if used
D1) Software	EGC251 C/C++ Programming	X
	CPS210 Computer Science I	
	CPS310 Computer Science II (Data Structures)	
	CPS353 Software Engineering	
D2) Computer Systems	EGC331 Microprocessors + EGC332 Microprocessors Lab	X
	EGC433 Embedded Systems	X
	EGC442 Computer Architecture	
	EGC451 Real-Time Systems	
D3) Digital Systems	EGC220 Digital Logic Fundamentals + EGC221 Digital Logic Lab	X
	EGC320 Digital System Design	
	EGC445 VLSI Design + EGC446 VLSI Lab	
	EGC447 Functional Verification	

Table 9.3: Mechanical Engineering (ME) Discipline and Courses within Discipline in the ME program

Discipline	Course in the discipline	Check if used
D1) Computers	EGE331 Computer Simulations	
	EGM302 Finite Element Analysis	
	EGM393 Advanced Computer Aided Design	
D2) Mechanics & Machines	EGM211 Statics	X
	EGM212 Dynamics	
	EGM311 Kinematics of Machines	

	EGM312 System Dynamics	
	EGM393 Biomechanics	
D3) Thermodynamics / Fluid Dynamics	EGM331 Thermodynamics	
	EGM332 Fluid Mechanics + EGM333 Fluid Mechanics Lab	X
	EGM334 Heat Transfer	X
	EGM335 Thermosystems Design	
D4) Materials	EGM221 Engineering Materials	X
	EGM322 Mechanics of Materials + EGM323 Materials Lab	X
	EGM393 Composite Materials	
	EGM393 Design of Machine Elements	

10. Design Constraints That Drove This Project

Table 9.1 shows the design constraints that drove this project

Table 9.1: Design constraints that drove this project

Design Constraint	Check if applied to this project
Economic	
Manufacturability	X
Social and Political	
Environmental	
Ethical	
Sustainability	
Health and Safety	
Other (explain)	

The objective of the project proposed by MPI was to create an EOAT solution that could be mounted to one of their robots. As such, the product would have to be practical to manufacture if it would be implemented in their automated cell. This includes being able to specify a viable production method for each component and reasonable tolerances.

11. Engineering Standards Used In This Project

Table 10.1 shows the engineering standards used in this project (this can be used as an introductory statement):

:

Table 10.1: Engineering standards used in this project

Engineering Standard	Where it was used

Only adhered to MPI's internal standards for designing EAOT

12. Appendix

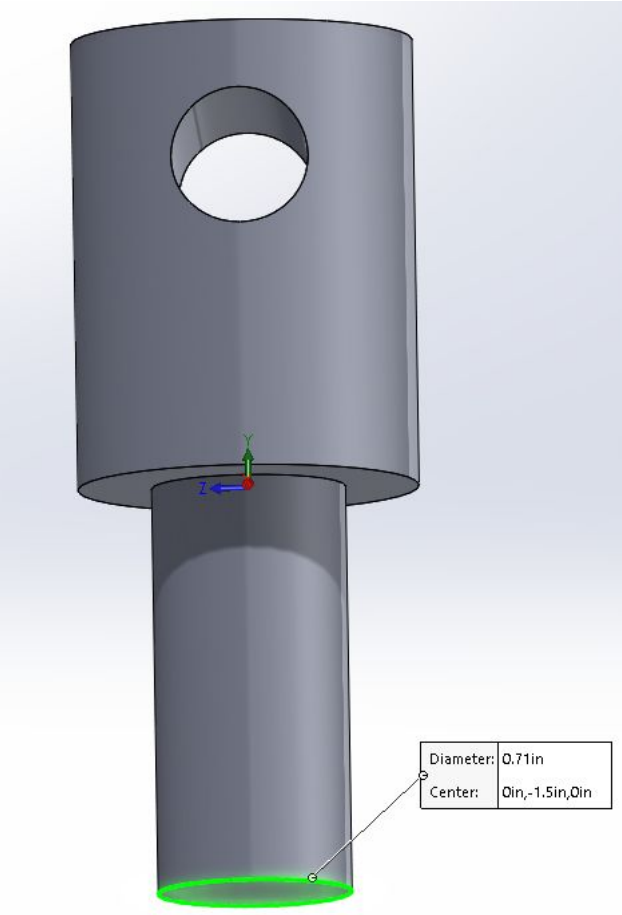


Figure A-1: 3D printed tool for the instron.

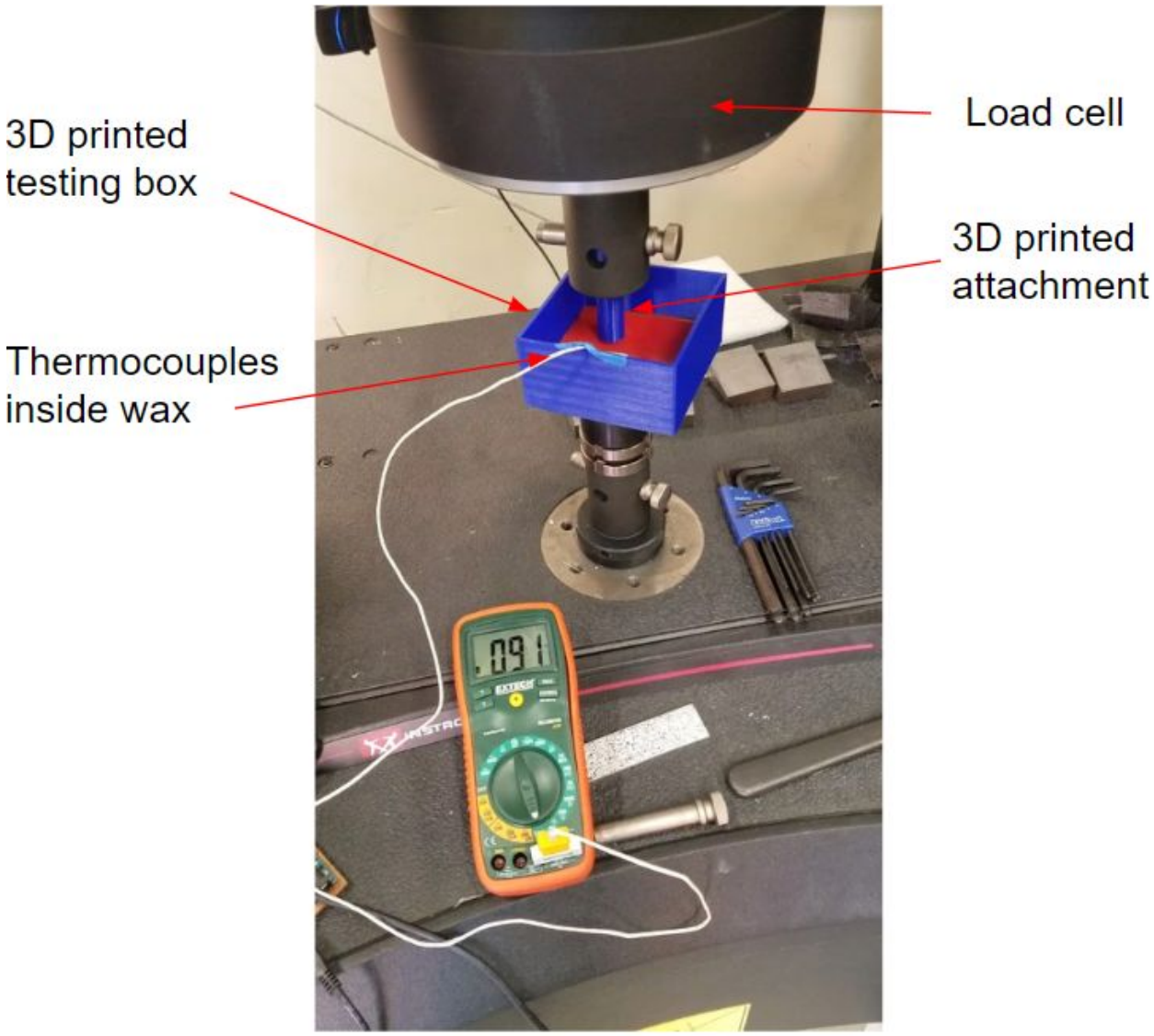


Figure A-2: Testing Apparatus for Wax Properties Experimentation

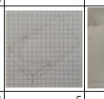
Figure A-2 shows the setup to test the properties of the wax. Wax was melted into a 3D printed box. Thermocouples were placed on the inside of the wax at a location that would not interfere with the tool. Due to the 3D printed testing box having a larger area than the 3d printed attachment, after a test was performed, a different location on the same wax was tested. This increased the efficiency of the experiment by reducing the time between the tests.

Table A-1: Pairwise Comparison Chart For Initial Design

Goal	Feasibility	Scalability (Size of Parts)	Multiple Parts (Different Parts)	Precision	"Smartness" (Sensing)	Cost	Score
Feasibility (Can we make it?)	x	1	1	1	1	1	5
Scalability (Size of Parts)	0	x	0	0	1	1	2
Multiple Parts (Different Parts)	0	1	x	0	1	1	3
Precision	0	1	1	x	1	1	4
"Smartness" (Sensing)	0	0	0	0	x	1	1
Cost	0	0	0	0	0	x	0

Table A-1 displays the comparison chart that ranked the important goals that each of the designs were judged against.

Table A-2: Decision Matrix

SCALE 0-5				Design 1	Design 2	Design 3	Design 4	Design 5	Design 6	Design 7	Design 8	Design 9	Design 10
		Name	ORIGINAL BOTTLEOPENER	Gripper / Cutter in One	The Crab (PREprogeramed)	Double-Pronged Pitchfork (PREprogeramed)	Spatula	Carnival Quad-Gripper	Suction-Cup Array	Inflateable balloon gripper	Suction cup on slidable grabber (PREprogeramed)	Multitool	The Blob
	Weight	Image											
Feasability (Can we make it?)	5		5	3	4	4	5	3	3	3	4	1	2
Scalability (Size of Parts)	2		0	3	4	2	3	1	5	4	3	2	4
Multiple parts (Different Parts)	3		0	4	2	2	1	2	4	2	3	5	4
Precision	4		5	1	4	4	1	1	2	2	4	5	2
"Smartness" (Sensing)	1		0	2	4	3	1	1	4	2	2	1	1
Cost	0		2	3	3	4	5	2	1	3	3	1	4
Sum			45	39	54	49	39	28	49	39	53	45	39

GRADING TECHNIQUE			
Scalability- Reprint, bigger size	5	YES, no issues	Excellent
	4	mostly yes, some issues	Great
	3	probably yes, but definit obstacles	Good
	2	probably not, many obstacles	Alright
	1	mostly not, many issues	meh
	0	impossible	Forget It

Table A-2 shows the decision matrix associated with the selection of the initial iteration. Each member of the team presented their idea and then judged against the criteria. This result yielded Design 2 as the chosen design.

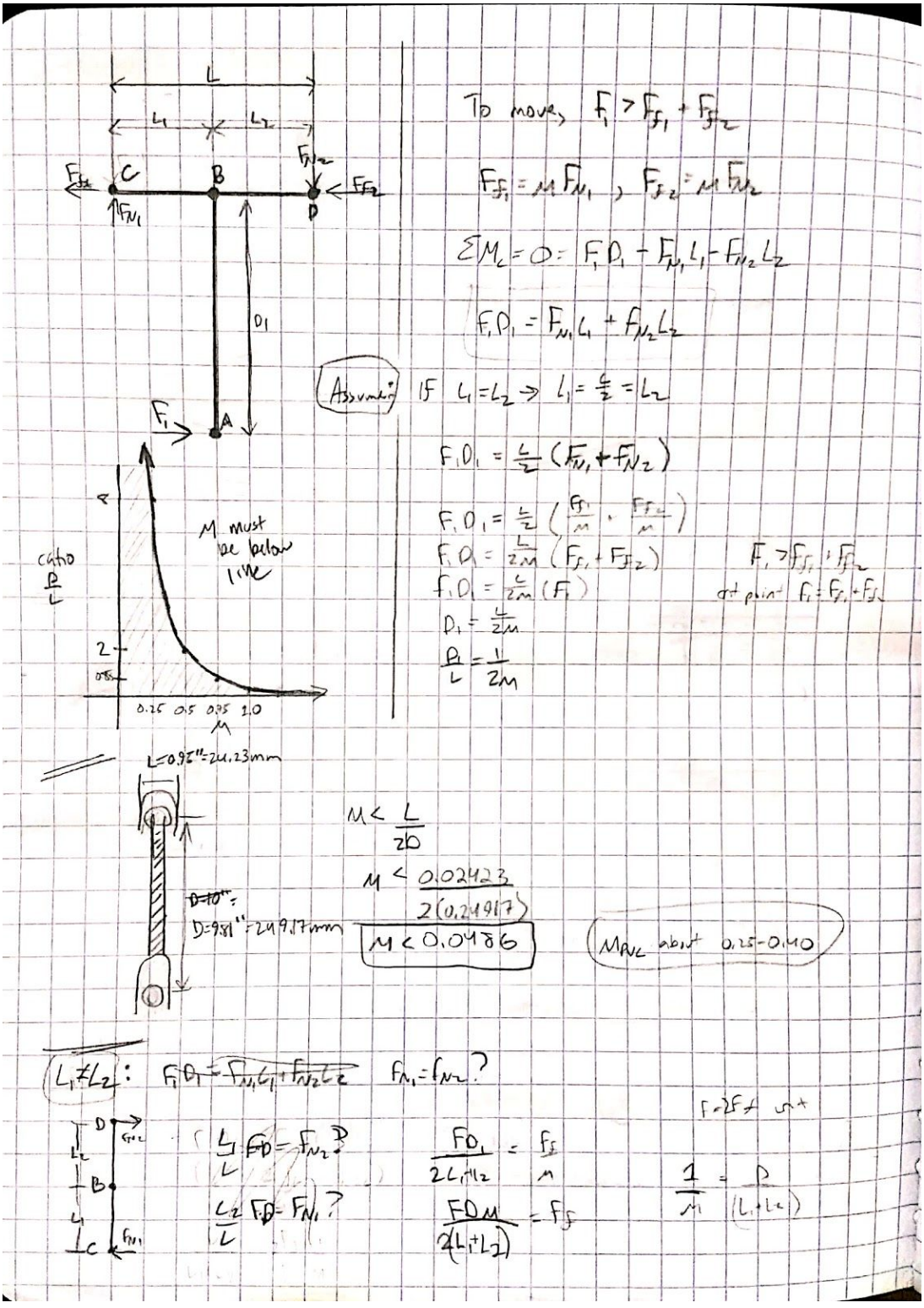


Figure A-3: Binding Calculations

12.1 Code:

main.py

```
# PWM on Raspberry Pi
# https://sourceforge.net/p/raspberry-gpio-python/wiki/PWM/

# Working with A4988 Motor Driver
# https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

# Project Utilizes Four NEMA 14 Bipolar Stepper Motors
# https://www.omc-stepperonline.com/download/14HS20-1504D-E22-300.pdf

# Motors are Driven Using A4988 Motor Drivers
# https://www.pololu.com/file/0J450/A4988.pdf

# Program is built to control the Multi-purpose Gripper End of Arm Tooling (EOAT) project for SUNY New Paltz Senior Design II

"""main.py :Main controller for EOAT """

# system exits
import sys
import os

# re.findall for file handling & line deletion
import re
import RPi.GPIO as GPIO
import time

# run parallel processes
import multiprocessing

__author__ = "George Dagis", "NAME OMITTED"
__copyright__ = "Copyright 2019, Sr. Design II EOAT"
__version__ = "1.0"
__email__ = "dagisg1@hawkmail.newpaltz.edu"
__status__ = "Prototype"

# Establish a running frequency and an optimal frequency
runningfreq = 5000

# NEMA 14 motor runs best at 9kHz but this is adjustable
# optimalfreq = 9000

# Refer to GPIO by GPIO#
GPIO.setmode(GPIO.BCM)

# Motor 1 is topmost on breadboard (screws on bottom)
Motor1StepPin = 27
Motor1DirPin = 22
Motor2StepPin = 17
Motor2DirPin = 4
Motor3StepPin = 5
Motor3DirPin = 19
Motor4StepPin = 26
Motor4DirPin = 13

# Set up encoder channels
Encoder1ChA = 23
Encoder1ChB = 24
Encoder2ChA = 18
Encoder2ChB = 25
```

```
Encoder3ChA = 12
Encoder3ChB = 20
Encoder4ChA = 16
Encoder4ChB = 21

GPIO.setwarnings(False)

# Motor GPIO
GPIO.setup(Motor1DirPin, GPIO.OUT)
GPIO.setup(Motor1StepPin, GPIO.OUT)
GPIO.setup(Motor2StepPin, GPIO.OUT)
GPIO.setup(Motor2DirPin, GPIO.OUT)
GPIO.setup(Motor3StepPin, GPIO.OUT)
GPIO.setup(Motor3DirPin, GPIO.OUT)
GPIO.setup(Motor4StepPin, GPIO.OUT)
GPIO.setup(Motor4DirPin, GPIO.OUT)

# Encoder GPIO
GPIO.setup(Encoder1ChA, GPIO.IN)
GPIO.setup(Encoder1ChB, GPIO.IN)
GPIO.setup(Encoder2ChA, GPIO.IN)
GPIO.setup(Encoder2ChB, GPIO.IN)
GPIO.setup(Encoder3ChA, GPIO.IN)
GPIO.setup(Encoder3ChB, GPIO.IN)
GPIO.setup(Encoder4ChA, GPIO.IN)
GPIO.setup(Encoder4ChB, GPIO.IN)

# Set the step pins and starting frequencies for motors
stepper1 = GPIO.PWM(27, runningfreq)
stepper2 = GPIO.PWM(17, runningfreq)
stepper3 = GPIO.PWM(5, runningfreq)
stepper4 = GPIO.PWM(26, runningfreq)

# Main method
def main():

  # Main prompt for user
  print()
  print("SELECT A DESIRED FUNCTION: ")
  print("1. START AUTOMATION WITH AN EXISTING PART.")
  print("2. VIEW / EDIT PARTS LIST.")
  print("3. EXIT PROGRAM.\n")

  # Register input
  menuSel= int(input("SELECTION: "))

  # Start automation
  if menuSel == 1:
     startautomation()

  # View / edit parts list
  elif menuSel == 2:
     vieweditpartslist()

  # Exit Program
  elif menuSel == 3:
     print("PROGRAM EXITING\n")
     sys.exit()

  else:
     print("UNACCEPTABLE INPUT. PLEASE TRY AGAIN")
     main()

# Starts automation on tool
def startautomation():
  print("SELECT PATTERN TO WORK ON: ")
```

```
printpartslistwithnumbers()
print()

# Get the desired part and convert input from str to type 'int' (will be compared later)
print("INPUT THE CORRESPONDING LINE NUMBER (Ex '3')")
print("YOU MAY ALSO QUIT WITH '0'")
parttoworkon = int(input("SELECTION: "))
print()

# Returns number of parts in PartsList.txt (make sure declaration is valid)
num_lines = sum(1 for line in open('PartsList.txt'))

# Quit
if parttoworkon == 0:
   print("QUITTING PROCESS. RETURNING TO MAIN MENU")
   main()

# Ensure a part on that line in the file exists
# Automation continue
elif parttoworkon > 0 and parttoworkon <= num_lines:

   # Decrement one, otherwise while loop will get the part after the desired line
   parttoworkon = parttoworkon - 1

   # Prints all integers on separate lines
   with open('PartsList.txt') as f:

      # keep track of lines skipped
      linecounter = 0

      # skip lines until desired line is the current line
      while linecounter != parttoworkon:

         # read (skip) line
         f.readline()

         # increment counter
         linecounter = linecounter+1

      # first line is now a str
      s = f.readline()
      print("ARE YOU SURE YOU'D LIKE TO START AUTOMATION ON THE FOLLOWING PART? (Y/N)",s)
      confirmautomation = input("SELECTION: ")

      if confirmautomation == 'Y' or confirmautomation == 'y':

         # extract all integers from string and put into a list
         coordinates = (re.findall(r'\d+', s))
         f.close

         # Assign values to cup position variable
         # Cast to int for comparison
         # x, y2
         Cup_A_X = int(coordinates[0])
         Cup_A_Y = int(coordinates[1])

         # 0, y1
         Cup_B_X = int(coordinates[2])
         Cup_B_Y = int(coordinates[3])

         # x, 0
         Cup_C_X = int(coordinates[4])
         Cup_C_Y = int(coordinates[5])

         # Always 0,0
         Cup_D_X = int(coordinates[6])
```

```
Cup_D_Y = int(coordinates[7])

# A: x, y1
# B: 0, y2
# C: x, 0
# D: 0, 0

#x = Cup_A_X
#y1 = Cup_A_Y
#y2 = Cup_B_Y

                # Step each motor based different number of steps
                # Steppers 1 and 2 always doing the same thing
                steps1 = Cup_A_X
                steps2 = Cup_C_X
                steps3 = Cup_B_Y
                steps4 = Cup_A_Y

                # Convert from inches to steps (20 threads/inch * 200 steps/rev * 1.5 encoder cycles/motor cycle
                steps1 = steps1*6000
                steps2 = steps2*6000
                steps3 = steps3*6000
                steps4 = steps4*6000

                # Ensure that x/y1/y2 are within bounds

if (Cup_A_X == Cup_C_X) and (Cup_A_X >= 0) and (Cup_A_X <= 14000) and (
      Cup_A_Y >= 0 and Cup_A_Y <= 20000) and (Cup_B_Y >= 0 and Cup_B_Y <= 20000):

   print("AUTOMATION WILL NOW BEGIN")

   # Step motors towards part
   GPIO.output(Motor1DirPin, 0)
   GPIO.output(Motor2DirPin, 0)
   GPIO.output(Motor3DirPin, 0)
   GPIO.output(Motor4DirPin, 0)

   # Step motors
   proc1 = multiprocessing.Process(target=stepmotor, args=(steps1,stepper1,Encoder1ChA,Encoder1ChB))
   proc2 = multiprocessing.Process(target=stepmotor, args=(steps2,stepper2,Encoder2ChA,Encoder2ChB))
   proc3 = multiprocessing.Process(target=stepmotor, args=(steps3,stepper3,Encoder3ChA,Encoder3ChB))
   proc4 = multiprocessing.Process(target=stepmotor, args=(steps4,stepper4,Encoder4ChA,Encoder4ChB))

   proc1.start()
   proc2.start()
   proc3.start()
   proc4.start()

   proc1.join()
   proc2.join()
   proc3.join()
   proc4.join()

   # Reverse direction; step motors away from part
   GPIO.output(Motor1DirPin, 1)
   GPIO.output(Motor2DirPin, 1)
   GPIO.output(Motor3DirPin, 1)
   GPIO.output(Motor4DirPin, 1)

   # Step motors
   procr1 = multiprocessing.Process(target=stepmotor, args=(steps1,stepper1,Encoder1ChA,Encoder1ChB))
   procr2 = multiprocessing.Process(target=stepmotor, args=(steps2,stepper2,Encoder2ChA,Encoder2ChB))
   procr3 = multiprocessing.Process(target=stepmotor, args=(steps3,stepper3,Encoder3ChA,Encoder3ChB))
   procr4 = multiprocessing.Process(target=stepmotor, args=(steps4,stepper4,Encoder4ChA,Encoder4ChB))
```

```
                procr1.start()
                procr2.start()
                procr3.start()
                procr4.start()

                procr1.join()
                procr2.join()
                procr3.join()
                procr4.join()

                # Exit program
                os._exit(1)

            # Coordinates not valid
            else:
                print("UNACCEPTABLE PART COORDINATES. PLEASE INSPECT THE PART AND MAKE SURE THEY FOLLOW THE FOLLOWING RULES:")
                print("CUP A X COORDINATE = CUP C X COORDINATE, >= 0 AND <= 14000")
                print("CUP A Y COORDINATE >= 0 AND <= 20000")
                print("CUP B Y COORDINATE >= 0 AND <= 20000")
                print("RETURNING TO MAIN MENU")
                main()

    # Part number out of bounds (specified a number greater than the number of parts in the file)
    elif parttoworkon < 0 or parttoworkon > num_lines:
        print("ERROR: PART NUMBER OUT OF BOUNDS. RETURNING TO MAIN MENU")
        main()

    # Unacceptable input
    else:
        print("UNACCEPTABLE INPUT. RETURNING TO MAIN MENU")
        main()

# Hub to view / edit parts list file
def vieweditpartslist():

    # Print PartsList
    printpartslist()

    # Print menu
    print("SELECT A DESIRED FUNCTION: ")
    print("0. GO BACK TO MAIN MENU")
    print("1. ADD NEW PART")
    print("2. REMOVE AN EXISTING PART")
    print()

    # Register input
    editSel = int(input("SELECTION: "))
    print()

    # Go back
    if editSel == 0:
        main()

    # Add a new part
    if editSel == 1:
        addpart()

    # Remove an existing part
    elif editSel == 2:
        removepart()

    # Unnacceptable input
    else:
        print("UNACCEPTABLE INPUT. GOING BACK TO MAIN MENU")
        main()
```

```
# Add a new part
def addpart():
  # Confirm with user
  print("ARE YOU SURE YOU WANT TO ADD A NEW PART? (Y/N): ")

  # Register input
  confirmAdd = input("SELECTION: ")
  print()

  # Confirmation
  if confirmAdd == 'Y' or confirmAdd == 'y':
    print("ADDING A NEW PART\n")

    f = open("PartsList.txt", "a")

    # Get name of part
    NewPartName = input("ENTER A NAME FOR THE NEW PART: ")

    # Get cup positions
    # CupA: x, y2
    NewPartCupA_X = int(input("ENTER X COORDINATE OF CUP A: "))
    NewPartCupA_Y = int(input("ENTER Y COORDINATE OF CUP A: "))

    # CupB: 0, y1
    #NewPartCupB_X = input("ENTER X COORDINATE OF CUP B: ")
    NewPartCupB_X = 0
    NewPartCupB_Y = int(input("ENTER Y COORDINATE OF CUP B: "))

    # CupC: x, 0
    NewPartCupC_X = int(input("ENTER X COORDINATE OF CUP C: "))
    NewPartCupC_Y = 0
    #NewPartCupC_Y = input("ENTER Y COORDINATE OF CUP C: ")

    # CupD: 0, 0
    #NewPartCupD_X = input("ENTER X COORDINATE OF CUP D: ")
    #NewPartCupD_Y = input("ENTER Y COORDINATE OF CUP D: ")
    NewPartCupD_X = 0
    NewPartCupD_Y = 0
    print()

    # Ensure that x/y1/y2 are within bounds
      if (NewPartCupA_X == NewPartCupC_X) and (NewPartCupA_X >= 0) and (NewPartCupA_X <= 14000) and (NewPartCupA_Y >= 0 and
NewPartCupA_Y <= 20000) and (NewPartCupB_Y >= 0 and NewPartCupB_Y <= 20000):

        # Confirm with user
        print("PLEASE REVIEW THE INFORMATION BELOW:")
        print()
        print("PART NAME: '",NewPartName,"'")
        print("CUP A X COORDINATE:", NewPartCupA_X)
        print("CUP A Y COORDINATE:", NewPartCupA_Y)
        print("CUP B X COORDINATE:", NewPartCupB_X)
        print("CUP B Y COORDINATE:", NewPartCupB_Y)
        print("CUP C X COORDINATE:", NewPartCupC_X)
        print("CUP C Y COORDINATE:", NewPartCupC_Y)
        print("CUP D X COORDINATE:", NewPartCupD_X)
        print("CUP D Y COORDINATE:", NewPartCupD_Y)
        addpartconfirmation = input("ARE YOU SURE THIS INFORMATION IS CORRECT? (Y/N): ")

        # Confirmation
        if addpartconfirmation == 'Y' or addpartconfirmation == 'y':

          # Concatenation of inputs
          NewPartFinal = NewPartName + " " + str(NewPartCupA_X) + " " + str(NewPartCupA_Y) + " " + str(NewPartCupB_X) + " " \
                         + str(NewPartCupB_Y) + " " + str(NewPartCupC_X) + " " + str(NewPartCupC_Y) + " " + str(NewPartCupD_X) + " " +
str(NewPartCupD_Y)
```

```
            # Make sure that it also starts a new line
            f.write("%s\n" % NewPartFinal)

            # Inform user a new part has been added
            print("",NewPartName,"' HAS NOW BEEN ADDED TO PARTS LIST FILE")

            # Ask user if they want to add another part
            addanotherpart = input("WOULD YOU LIKE TO ADD ANOTHER PART? (Y/N): ")

            # Yes, add another part
            if addanotherpart == 'Y' or addanotherpart == 'y':
                addpart()

            # Do not add another part
            elif addanotherpart == 'N' or addanotherpart == 'n':
                f.close()
                print("PROCESS CANCELLED. RETURNING TO MAIN MENU")
                main()

            # Else, unrecognized input
            else:
                f.close()
                print("UNACCEPTABLE INPUT. RETURNING TO MAIN MENU")
                main()

        # Go back to main menu
        elif addpartconfirmation == 'N' or addpartconfirmation == 'n':
            f.close()
            addpartcontinue = input("PROCESS CANCELLED. WOULD YOU LIKE TO TRY ADDING A NEW PART AGAIN? (Y/N): ")

            # Confirmation
            if addpartcontinue == 'Y' or addpartcontinue == 'y':
                addpart()

            elif addpartcontinue == 'N' or addpartcontinue == 'n':
                print("PROCESS CANCELLED. EXITING TO MAIN MENU")
                main()

            # Exit
            else:
                print("UNACCEPTABLE INPUT. GOING BACK TO MAIN MENU")
                main()

        # Exit
        else:
            f.close()
            print("UNACCEPTABLE INPUT. RETURNING TO MAIN MENU")
            main()

    # Coordinates not valid
    else:
            print("UNACCEPTABLE INPUT. PLEASE REVIEW YOUR INPUTS AND MAKE SURE THEY FOLLOW THE FOLLOWING
RULES:")
        print("CUP A X COORDINATE = CUP C X COORDINATE, >= 0 AND <= 14000")
        print("CUP A Y COORDINATE >= 0 AND <= 20000")
        print("CUP B Y COORDINATE >= 0 AND <= 20000")
        print("RETURNING TO MAIN MENU")
        main()

  # Go back to main menu
  elif confirmAdd == 'N' or confirmAdd == 'n':
    print("PROCESS CANCELLED")
    main()

  # Exit
```

```
  else:
    print("UNACCEPTABLE INPUT. RETURNING TO MAIN MENU")
    main()

def removepart():
  printpartslist()
  print()
  print("PLEASE SPECIFY PART TO DELETE (Ex. 'Blue Bottlecap Openers')")
  print("YOU MAY ALSO QUIT WITH 'N'")
  removeSel = input("SELECTION: ")
  print()

  if removeSel == 'N' or removeSel == 'n':
    print("QUITTING PROCESS. RETURNING TO MAIN MENU")
    main()

  else:
    fn = "PartsList.txt"
    f = open(fn)
    output = []
    for line in f:
      if not removeSel in line:
        output.append(line)
    f.close()

    f = open(fn, 'w')
    f.writelines(output)
    f.close
    print("",removeSel,"' HAS BEEN DELETED FROM PARTS LIST (IF IT EXISTS)")

    # Ask user if they want to remove another part
    removeanotherpart = input("WOULD YOU LIKE TO REMOVE ANOTHER PART? (Y/N): ")

    # Yes, remove another part
    if removeanotherpart == 'Y' or removeanotherpart == 'y':
      removepart()

    # Do not remove another part
    elif removeanotherpart == 'N' or removeanotherpart == 'n':
      f.close()
      print("PROCESS CANCELLED. RETURNING TO MAIN MENU")
      main()

    # Else, unrecognized input
    else:
      f.close()
      print("UNACCEPTABLE INPUT. RETURNING TO MAIN MENU")
      main()

# Prints parts list file in alphabetical order
def printpartslist():
  print("CURRENTLY EXISTING PARTS: ")
  print()
  f = open("PartsList.txt", "r")
  PartsList = f.readlines()
  PartsList.sort()
  for x in PartsList:
    print(x)

# Print parts list with line numbers
def printpartslistwithnumbers():
  with open('PartsList.txt') as f:
    for i, line in enumerate(f):

      # Print part on line 1
      while line:
```

```
            # Increment to count from 1 rather than 0
            i=i+1;

            # Convert i to string for concatenation
            linenumber = str(i)

            # Concatenate
            numberedline = linenumber + ". " + line
            print(numberedline)
            break

      # No more lines in file
      else:
         print("*FINISHED READING FILE*")
         line = None

# Function to step specific motor
def stepmotor(stepsm, stepper, EncoderChA, EncoderChB):

   # Duty cycle is 50%
   stepper1.start(50)

   # Define initial values
   currentA = currentB = currentX = lastA = lastB = lastX = count = 0

   # Account for encoder (300 cycles/rotation rather than 200)
   # Start motor
   steps = stepsm*1.5
   stepper.start(50)

   # Encoder steps
   while count < steps:

      # Get current pin values
      currentA = GPIO.input(EncoderChA)
      currentB = GPIO.input(EncoderChB)

      # A XOR B
      currentX = bool(currentA) ^ bool(currentB)

      # Rise edge B + falling edge X = add
      if (currentB*1 > lastB*1) and (currentX*1 < lastX*1):
         count += 1

      # Reset X
      lastX = currentX

   stepper.stop()

if __name__ == '__main__':
   main()
```

PartsList.txt

Below is the contents of a sample PartsList text file. There is an empty line at the end of the file in order to work with file handling in Python.

testpart1 2.5 3.1 0 3.1 2.5 0 0

testpart1 1.2 2.8 0 2.8 1.2 0 0