Metamorphic Manufacturing v2

Authors:
Sean Barry
Charles Kittler
Jonathan Landay
Jake Mackenzi
Aidan Melgar
Patrick Siegler

Faculty Advisors:
Professor Berk Calli, CS/RBE
Professor Adam C. Powell, ME
Professor Craig B. Putnam, RBE

28 April 2022

Abstract

Metamorphic manufacturing is a type of digital fabrication wherein robotic systems are employed to incrementally deform material into the desired shape. This approach has the potential to reduce material waste compared to subtractive methods such as CNC milling and can achieve material properties superior to what is possible with additive methods like 3D printing. The aim of this project is to develop a prototype metamorphic manufacturing system capable of shaping plasticine clay. Utilizing the 6-axis ABB IRB 1600 robotic arm equipped with a custom end-effector and interchangeable tools, a system was constructed to achieve this objective. A LiDAR camera was explored to capture accurate 3D models of the Plasticine workpiece as it is being shaped. Automated tool-changing was also explored to allow for a fully automatic workflow.

Acknowledgements

We would like to thank our faculty advisors, Professors Putnam, Calli and Powell for their advice and guidance throughout this difficult project. We would also like to thank Arnold Muralt from last year’s team, Professor Bergstrom and the Washburn Shops staff for their help and expertise with the ABB Robot.

Table of Contents

List of Figures

List of Tables

1 Introduction

To improve upon and extend the opportunities posed by additive and subtractive manufacturing, a new approach, known as metamorphic manufacturing, is being developed. This new approach rapidly manufactures complex parts with little to no waste. Incremental forming, which is run in a numerically controlled closed-loop, creates complex shapes. During this process, incremental deformation combines with robotic-machine intelligence to surpass the limitations of previous manufacturing methods. Metamorphic manufacturing is also referred to as robotic blacksmithing as it employs many of the techniques of blacksmithing but replaces the work of a human with that of a robotic system. The five main aspects which create the foundation for metamorphic manufacturing are sensors, actuators and forming tools, robotic manipulation systems, and computation. Sensors provide real-time monitoring of the system. Actuators and forming refer to the tools which are used to form the material, such as hole punching, pressing, or shaping. Robotics systems are the integration of the actuators and forming tools with robotic arms. Lastly, computation controls the sequencing, tool paths, sensors, and decision-making of metamorphic manufacturing. This process presents key benefits such as lower material waste, reduced energy consumption, more complex control, process optimization, increased manufacturing flexibility, and real-time error correction. This project seeks to improve upon the prior work towards creating metamorphic manufacturing methods through research of current manufacturing techniques and systems. For the first year of this MQP, the goal of the project was to create a working prototype that forges, and molds blocks of plasticine clay, using an ABB IRB 1600 robotic arm, fitted with custom tooling. This year focused on improving and implementing these systems. This is meant to be a continuing project, so the team strives to set a good foundation for future improvement upon the systems involved.

2 Background

2.1 Literature Review

There are other computer driven forms of manufacturing that already exist. These can be broken down into two main categories, additive and subtractive manufacturing. These have their own benefits and drawbacks, which may be replaced by or used alongside metamorphic manufacturing.

Additive manufacturing focuses on building a design layer by layer to create an object. There are several forms of additive manufacturing, allowing for the creation of products in many different material types, some examples are thermoplastics, resins and some metals. As a result of being added layer by layer, the surface finish often has layer lines and other imperfections. In some forms of additive manufacturing the end of the process, the product needs to have the support material removed, leaving imperfections, and requiring postprocessing [1]. There is also the problem of having a weaker direction due to weaker shear forces in the direction of the layer lines. For metal additive manufacturing there are several problems with feed material properties as well as grain structure of the finished part [2].

The largest benefits of additive manufacturing are the ability to control volumetric density through materials, as well as the ability to create products with more advanced features, such as internal cavities. Additive manufacturing also produces far less waste than subtractive manufacturing, as the only waste produced is in support materials [1].

Subtractive manufacturing works by taking a piece of stock and removing material until the desired shape is created. Most times, a CNC machine is used to machine the part to the required specifications. There are some advantages to having a part machined rather than 3D printed; Machined parts tend to be more accurately made and be more cost-effective. Another interesting aspect of machining is that it can be used to create and/or finish products. However, as was with 3D printing, machining also has its downsides. Since machining relies on cutting away one block of material, it can waste significant amounts of the stock material [3], [4].

Metamorphic manufacturing provides a different method which utilizes intelligent sensing robotic systems, with deformation of materials to create a nearly entirely waste free system, capable of manufacturing shapes not normally possible with the other forms of manufacturing. Unlike subtractive manufacturing most of the material is used and shaped without needing to cut away or discard extras. This further adds the ability to manipulate certain material properties such as the hardness of the material, through heat treatments such as quenching or annealing. This process can also work with many materials of much higher melting points than additive manufacturing can work with normally, allowing metals to be bent and shaped with the strength of the robots, and in conditions not suited for humans.

2.1.2 Blacksmithing and Robotic Blacksmithing

Blacksmithing is a process that has existed for thousands of years and can be found in all cultures across the globe. It is defined as the trade of a blacksmith, one that forges and shapes iron with an anvil and hammer. Over the years, blacksmithing has evolved to incorporate newly developed machinery to assist in the process, such as a hydraulic presses and power hammer, yet maintains the core properties of forging using simple tools. Tools such as tongs, hammers, smelters, and anvils are all used today in a similar fashion as they were used in earlier times [5]

There are three major processes that encompass blacksmithing: heating, maneuvering, and striking. The heating process consists of exposing metal to a significant heat source for a period, changing the malleable enough to shape into a specific design. The temperature of the metal can be changed based on the length of time the metal is subjected to the heating source but should not be heated to the point of the melting temperature of the metal. As the metal gradually becomes more malleable, the metal can then be deformed. The second process uses tools such as tongs or a vice to maneuver materials into positions for striking. The tools can grip the material in place, as well as reposition a material to a different orientation. Four types of tongs used in blacksmithing: flat tongs, straight lip tongs, rivet tongs, and gad tongs. Tongs allow a blacksmith to grip various heated materials, facilitating easy movement, whether it be to change the orientation of the material or move the material between a forge and anvil without being burned. The third process consists of striking the material with a hammer on an anvil. Hammers are used to hit malleable materials and deform the surface of a workpiece into different shapes and sizes as needed. Types of hammers include drop hammers, power hammers, hand hammers, and sledgehammers. These hammers can be used on all forging materials, regardless of temperature.

Robotic blacksmithing incorporates these three techniques in similar ways but with small variations. Due to the wide range and capabilities of robots, these tools can be used for multiple blacksmithing uses, such as a gripper used as both tongs and a press to move and reshape a material. A wide range of motion and strength eliminates the fatigue of a human consistently striking a material with a hammer, significantly decreasing manufacturing time. A robot can quickly strike materials uniformly, allowing for precise and rapid duplication rates.

The first concept of robotic manufacturing was demonstrated by an Ohio State University team of undergraduates in 2017 for a government-funded consortium called Lightweight Innovations for Tomorrow, or LIFT. This challenge demonstrated the basic processes of metamorphic manufacturing using both hardware and software to deform plasticine [6]. However, for both the hardware and software to communicate simultaneously, the system must be able to understand "the shape, temperature, and condition of the material at each location of the part being formed," as well as "control the temperature to produce the right structure and properties." The LIFT challenge had several shapes which people were to make, the most challenging being a horseshoe [7].

2.1.3 Metamorphic Manufacturing

Metamorphic manufacturing is an innovation in the field of manufacturing that aims to combine material deformation with the precision of intelligent machines and robotic systems. This method of manufacturing has been adopted by various institutions seeking to further test out the benefits that it provides. One example of this would be The Ohio State University's LIFT challenge Team Honey Badger and their endeavors in making a robotic blacksmith [8].

The objective for Team Honey Badger was to construct an autonomous system that could perform a series of deformation operations on a block of material, which in this case was plasticine. The end goal was to form the plasticine into a horseshoe and a bracket. To accomplish this the team opted to build a CNC mill and Arduino based deformation system. This system was built out of parts that were either purchased from a supplier or custom made in a 3-D printer. Once completed the system consisted of the CNC mill, Arduino control system, tool changer assembly, and a thermal heating bed. The mill itself provided the system with basic translational X, Y, and Z movements, while the Arduino controlled the tool changer assembly along with the thermal heating bed for the plasticine. Finally, the system included shaping tools. These parts consisted of a set of jaws to both squeeze and flatten the material as well as to grip the other tools. An example of the other tools that could be picked up by the gripper was a hole punch. At the conclusion of the project, Team Honey Badger was able to form both the horseshoe and bracket using the system they had built. While the result was not exactly like the horseshoe and bracket that had been 3-D printed, the project results displayed how this method of manufacturing could have various applications.

Like The Ohio State University's Team Honey Badger, the metamorphic manufacturing MQP from the 2020-2021 academic year at WPI centered around taking a block of plasticine [9] and utilizing a robotic system to shape it into an object[10]. The team started their approach by testing assorted brands of plasticine and putting them through stress tests which were used to determine a suitable linear actuator. When designing the robotic system, the team utilized Solidworks to design parts. A file-sharing software called Grab CAD Workbench was used to create a more efficient way of sharing CAD. The team was also provided with already existing components such as an ATI QC-11 robot side connector along with several compatible tool side connectors. The connectors were used to allow for pneumatic and signal pass through to the End of Arm Tooling (EOAT) and allowed for the robot to potentially switch out the EOAT automatically. The lab provided the team with a Schunk 3-jaw pneumatic gripper, which was able to close and open on cylindrical or hexagonal shafts when given pneumatic pressure.

The final system that the team produced was based around a central gripper, which served as the primary end effector for the ABB IRB 1600 arm. This gripper part was designed to be able to manipulate the material as well as hold any additional tools. The additional tools were designed for tasks that the gripper jaw alone could not accomplish such as making holes and detailed finishings. Due to the intended automatic nature of the arm, the team produced a passive tool rack where parts would be placed so that they could be switched out as needed. The jaws of the main gripper were designed to be a universal shape that could interface with any of the additional parts that would need to be switched out [10].

2.1.4 Project Equipment and Communications

As this project was intended to develop an overall system capable of metamorphic manufacturing by combining off the shelf components with custom-designed tooling, several separate smart devices which are required to communicate with one another to control the mechanical components and give feedback to the system. Our metamorphic manufacturing robot system consists of two separate mechanical components: a manipulator responsible for the positioning of the working material and the tooling responsible for shaping the component. For this project, the manipulator consists of a 6 Degree-of-Freedom (DOF) robotic arm, while the tooling is a custom-built gripper. Since the two mechanical components are not natively integrated by the original equipment manufacturer (OEM), particular care must be taken to generate appropriate control signals and to read feedback signals.

The manipulator portion of the system is responsible for ensuring the working material is appropriately positioned within the tooling for the molding operation is performed, and several different configurations exist. The first configuration of the system involves static forming, with the material placed upon a 3 DOF platform which moves and orients the material within the working range of the tooling. A second configuration utilizes multiple DOF robotic arms to manipulate the tooling around the static working material to form the desired object. A third configuration involves both a multiple DOF platform and multiple DOF robotic arms working in tandem. An example of this configuration can be seen by Ohio State University's Team Honey Badger in the LIFT Metamorphic Manufacturing Challenge [10]. There are benefits to each configuration ranging from the cost of each subcomponent to the required mobility and versatility of the final mechanism. In our implementation of metamorphic manufacturing, we utilized a 6 DOF ABB IRB 1600 robotic arm (See Figure 1) to manipulate our custom forming device. The IRB 1600 is an industrial robot capable of manipulating 6kg with a reach of 1.2 meters[11], and is controlled by ABB's IRC5 robot controller.

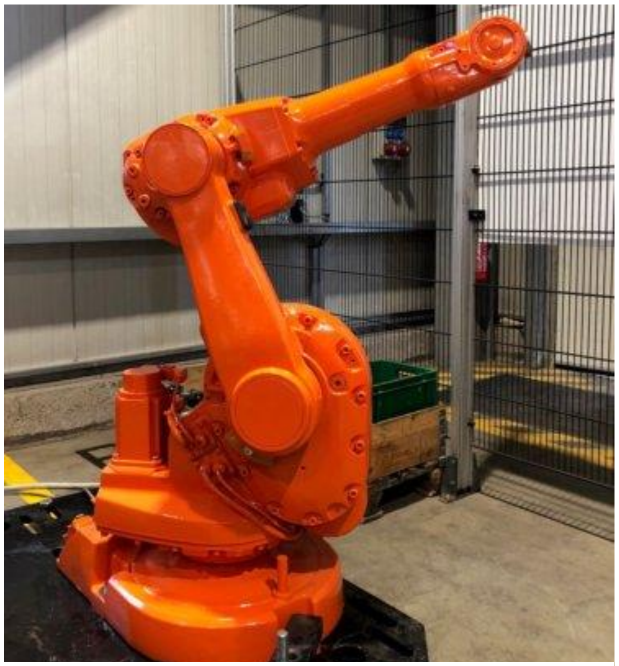


Figure 1: ABB IRB 1600 Serial Manipulator

The forming portion of our project was developed a year prior by another project team [10] and consists of two independent

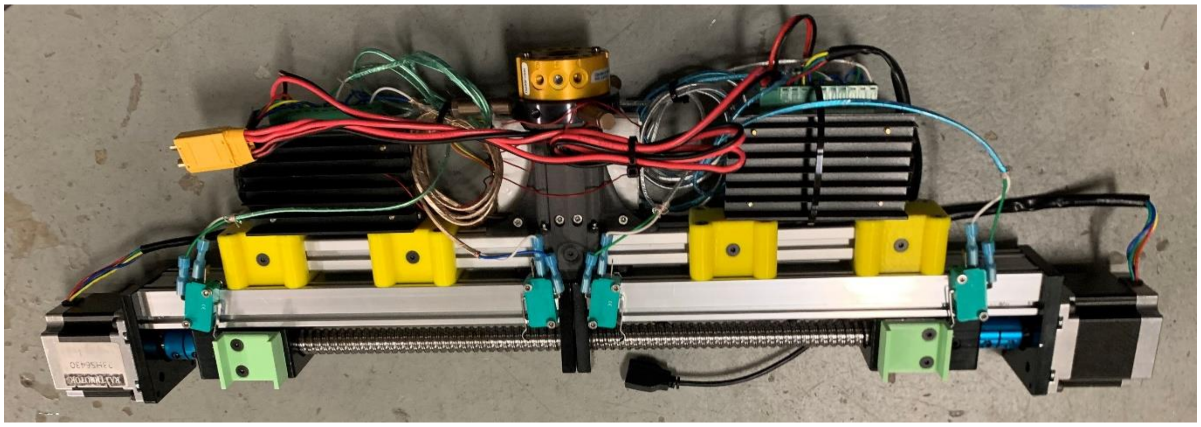


Figure 2: End of Arm Tooling

interchangeable tools on a linear slide driven by a pair of stepper motors as seen in Figure 2. By opening and closing the tools, as well as pressing upon the material in various manners, the working material can be deformed and shaped into the desired final product. As this portion is mounted to the end of the IRB 1600 arm, it functions as an EOAT device. Since the EOAT is not manufactured by ABB, and is not natively interfaced with the IRB 1600, additional modules are necessary to allow the IRC5 to communicate with the EOAT. While the IRC5 is capable of generating the electrical signals and pulses necessary to control the movement of stepper motors, other equipment such as Programmable Logic Controllers and microcontrollers or computers such as Arduinos or Raspberry Pi's are more suited to this task.

Serving as intermediaries between the IRC5 and the EOAT, PLCs and Microcontrollers/Minicomputers serve the same purpose: offloading the generation of appropriate control signals and the interpretation of digital sensors from the overall robot controller. In doing so, the robot controller can focus on the high-level processing necessary to form and position the EOAT, while minutiae can be decided upon independently by an external subservient processing unit that takes high-level commands issued by the robot controller. The industry-standard device fulfilling this purpose is the PLC, a robust and capable device that monitors several inputs and switches outputs depending on logic programmed via a coding language known as "Ladder Logic," [12]These devices are housed in cabinets and are resistant to electrical noise and other interference. Microcontrollers such as the popular Raspberry Pi are capable of many of the same functions, however, are less robust, and more geared towards rapid prototyping, or ease of use. With many General-Purpose Input Output (GPIO) pins available, these devices can be configured and reconfigured with ease and may be programmed in several common languages including the C Family and Python.

2.1.5 Software

In order to establish communication between all the subsystems, we decided to use Robot Operating System (ROS). ROS is a set of software libraries and tools that help to build robot applications, and it is not an operating system. When using ROS, software is implemented as nodes that can be connected using either topics or services. Topics work in a broadcaster and subscriber system. The broadcaster will publish messages of a predetermined type. Services work in an on-demand fashion, by sending a message as a request the receiving node can do something and then send a message back in a one-to-one interaction.

The alternative option that was considered was writing the code in Rapid (ABB’s Proprietary Robot Coding Language) as well as using a PLC that was on site to interface with the robot, as well as being the industry standard for an industrial environment. There was already a small amount of work done on this front from the previous team, but ultimately the problems with this system were deemed too much to be time efficient. The first problem was the lack of experience with either system. Additionally, there seemed to be a problem getting the PLC CIRTIO module to work, the curio module was responsible for generating control signals for the stepper motors. Even in the case where the curio module was working the number of wires that were being run to the end of arm tooling was a concern. The lack of confidence in the system led to it not being implemented.

2.2 Social Implications

The impact of metamorphic manufacturing soon may redefine how we see manufacturing. The first thing metamorphic manufacturing would be able to do is reduce the amount of waste from manufacturing. The two most popular methods of manufacturing, additive and subtractive, both have flaws regarding waste material. With CNC, parts are machined out of a piece of material leaving only the final part left. The excess material must go somewhere, and it usually ends up thrown away or wasted, but in some cases, it is recycled or reused. While subtractive manufacturing is inherently wasteful, additive manufacturing is quite good for material usage. 3d printers are a popular form of additive manufacturing but like CNC, they can have some drawbacks. The main form of material waste with 3d printers comes from failed prints, support material, skirts, brims, etc. Things like support material and brims are necessary, but they are designed to be printed once and thrown away after. Failed prints are sometimes uncontrollable and the waste they generate is variable, which makes it another often unavoidable problem.

Metamorphic manufacturing presents an improved solution to the waste problem as well as some other issues that current manufacturing techniques suffer from. An effective way to save on material is to use all of what is given to you, and that is what metamorphic manufacturing is all about. By reshaping, forming, squishing, and hitting material to morph it into the desired shape little to no material is lost. This means that no material must be thrown out or recycled, which has the potential to save lots of time, money, and resources in the manufacturing process. It also has the potential to save us and our planet. By producing less waste, excess material will not end up in landfills, the wild, and in the oceans. As we move towards more sustainable production techniques and more renewable energy sources, things like metamorphic manufacturing can change far more than just the production of goods.

Another key problem metamorphic manufacturing sets out to solve is production quality. As it stands, current manufacturing techniques do not have a good method of quality control. If a cast part has a defect and is sold, the company is liable to either take it back and fix it or provide the customer with a new one. If a print fails, usually that means you just throw it away. Metamorphic manufacturing does not need quality control after the fact because it is done live by what is producing the part.

2.3 Safety Precautions

With any industrial system such as a metamorphic manufacturing one, there are a few points of safety that need to be addressed. First, the operator, and all personnel near the robot must wear proper personal protective equipment. This includes hearing and eye protection appropriate for a manufacturing setting. Closed toe footwear suitable to protect feet from injury. All jewelry, long hair, and baggy clothes should not be worn when possible or tied up/tucked away. Clothing that fully covers arms and legs is advised when possible.

Second, when the robot is on, or the current state is unknown, it is important that all nearby people keep their distance. The robotic arm that is being used, can move at speed and forces capable of causing significant harm or even death, should it impact or pinch someone. When the robot is operating it is advised that there be a barrier to enclose the robot and its workspace. A physical barrier to separate the team and robot or invisible fence which halts the robot when crossed, are to be always used when the robot is moving autonomously. It is also important to have as many emergency stop buttons as needed so that any person from any reasonable location can stop the robot in a timely manner.

Third, any person that wants to jog the robot or interact with the robot's workspace, while the robot is on, needs to make sure that the robot is in a user interaction safe mode. The robot is to be controlled exclusively by the Flex Pendant, to ensure that the robot moves in a predictable fashion.

Fourth, for all persons that are writing code, or designing systems for the robot, security and safety needs to be a high priority. Systems should be designed in a way that allows the robot to halt immediately upon pushing an emergency stop button or any other safety system is triggered. Care should be made to maintain a secure and closed software ecosystem for the robot and its control, to prevent outside software, or persons from influencing and/or controlling the robot.

Fifth, although it was outside the scope of this project it is important to consider for future projects the environment/workpiece. The robot will eventually have to handle and work in elevated temperature environments. When this happens the level of PPE must be adjusted to handle the more dangerous environment. Also, all robotic systems must be evaluated on their ability to withstand the conditions that they will be subjected to, and all systems that fall short must be adjusted before operating in an elevated temperature environment.

3 Methodology

3.1 Previous Work

Since this MQP is in its second year, we were not starting from scratch and had the previous year's work to build upon. The previous team had focused primarily upon developing the hardware for the first stages of the project, where Plasticine Clay serves as a stand-in for metals heated to forging temperatures [10]. Their solution involved using a custom-built EOAT mounted to the ABB IRB 1600 robotic arm to squeeze the clay. The EOAT was comprised of a pair of lead screws driven by stepper motors (see Figure 3) and a L515 RealSense Lidar camera to provide 3d scans of the workspace.

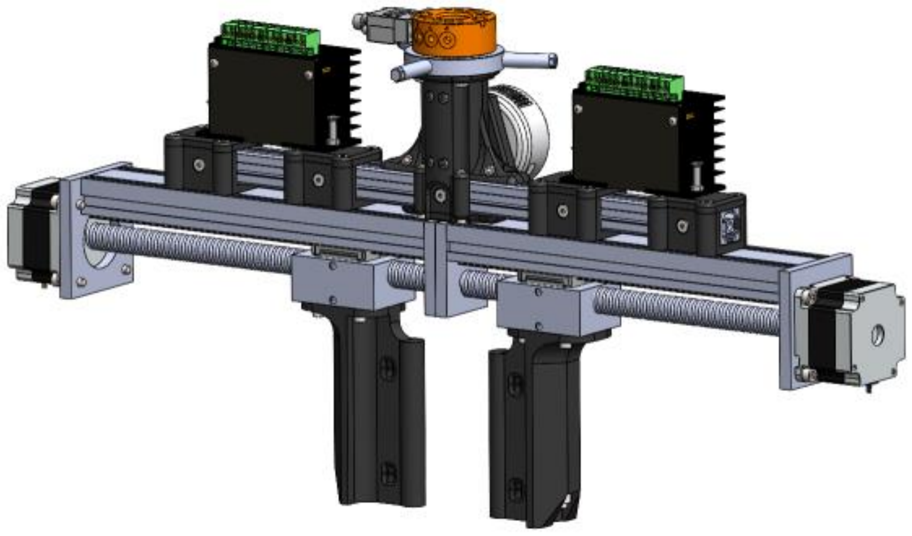


Figure 3: Previous Team's End of Arm Tooling Design

While examining the previous team's work, we identified several areas which we wanted to improve upon including the wiring (See Figure 4), tools and methods of connecting various subsystems. When we first started the project, the EOAT wiring was not functional and thus needed significant attention prior to running the robot due to faulty connections. In addition, the CITRIO module of the programmable logic controller (PLC) the team had intended to use to generate the stepper motor driver control signals was not functioning.

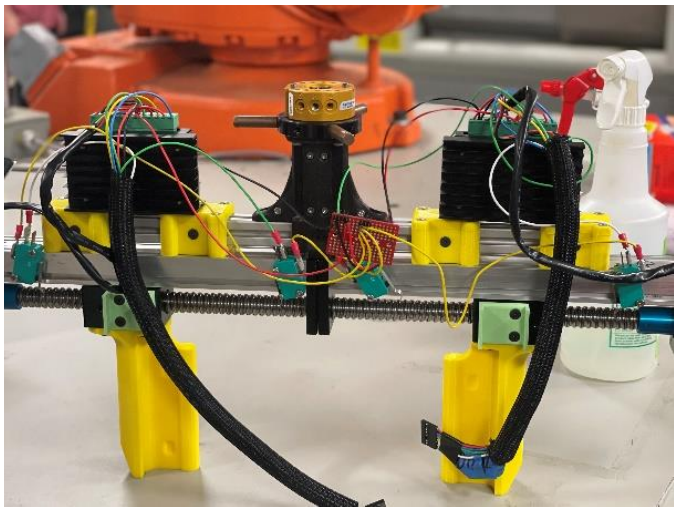


Figure 4: Inherited Wiring

When we started the project, only the tool changers were present on the EOAT, and the other tools developed by the previous team were missing. Although the tool changers were also intended to be usable as tools, when we examined the tool changers, we determined that they were not suitable for manipulating the clay as they would accumulate plasticine over time and become unusable as mounting points for other tools. Additionally, it was determined that the tool changers were unable to hold any of the other tool as the rounded edge of the tool changer did not fit into the angled shape of the provided tool.

In connecting all the subsystems, the previous setup did not provide much of a roadmap for controlling each independent system save using a PLC to generate the signal pulses for the stepper motor controllers on the EOAT. We determined that this was causing several problems as the existing connection across the IRB 1600 were prone to disconnecting as the robot moved, in addition to the module responsible for generating pulse trains on the PLC not functioning.

Knowing these issues, we looked into other systems and settled upon ROS for its modularity. This would allow for the implementation of the other sub systems, multiple computers and any of many prewritten packages. It would also easily allow for the code to be easily upgraded, because old nodes can be exchanged with new ones. The second reason is that there are ROS packages for the IRB 1600, that can be used to path plan and send trajectories to the robotic arm. Another reason for picking ROS was it allowed for networked computers, which would allow for fewer cables to be run to the end of arm tooling. Members of the team were already familiar with working with ROS, and along with the other benefits ROS was the clear choice for this system.

3.2 End of Arm Tooling Design

3.2.1 Design Considerations

The basic design of the overall project we inherited is based on a two-part system: the IRB 1600 robotic arm, and the End of Arm Tooling. Since the EOAT is a custom-built mechanism, it does not directly interface with the IRB 1600’s control system (IRC5), thus requiring the transmission of control signals to and from the tooling, as well as power for the motors.

As the IRB 1600 is shared with other groups and classes, the EOAT is attached to the IRB 1600 by an ATI QC-11 robotic tool changer (see Figure 5), allowing the tooling to be easily detached. This necessitates some manner of electrical connection at the end of the robot arm, where the number of unique contacts depends on the number of control signals and power requirements of the EOAT.

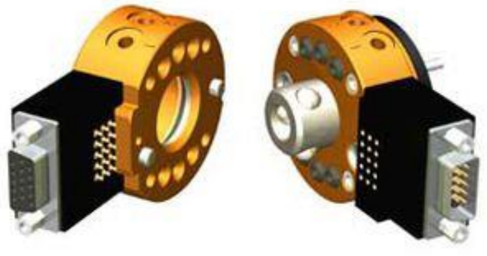


Figure 5: QC-11 Robotic Tool Changer with example electrical passthrough

The EOAT for this project is composed of a pair of stepper motors and accompanying stepper motor drivers. Each stepper motor/driver pair requires a power source between 9 and 48 volts. At 12 volts, each motor draws 3.2 amps, necessitating a total of approximately 7 amps of 12-volt power delivery. In addition, each stepper motor driver requires 3 control signals.

3.2.2 Signal Generation: PLC vs Raspberry Pi

Each stepper motor controller requires 3 unique control signals in order to rotate the motors the desired distance: A direction signal, an enable signal, and a step signal. The direction and enable signals are simple binary logic; if voltage is applied to the direction input, the motor might spin counterclockwise, if no voltage is applied, it spins in the opposite direction. The enable signal turns the motor on and off, but unlike a conventional DC motor, this does not directly translate into motion, but merely turns the current on and off. If the motor is not enabled, the motor is free to spin under any applied torque, whereas if it is enabled, the motor is locked in the current position, or “step.” When a pulse, or switch from no voltage to input voltage (or vice versa depending on the particular stepper motor driver configuration) is applied to the step/pulse input, the motor increments by one “step,” or a fraction of its rotation, thus in order to get continuous rotation, a series of pulses, or pulse train must be applied to the step input. The speed of the motor can be controlled by varying the delay between subsequent pulses.

In conventional industrial settings, such a pulse train can be supplied by a Programmable Logic Controller (PLC). While a PLC is a robust source of clean signal in an electrically noisy shop, it is a sizable module and is not mountable to the end effector of the robot, necessitating the transmission of 6 control signals along the IRB 1600 arm, through the QC-11. Our alternative source of signal generation is the Raspberry Pi microcomputer, a palm sized computer capable of generating digital signals and connecting to wireless networks. Using the Raspberry Pi would eliminate the need to transmit 6 data lines through the connection point (see section 3.2.3 Electrical) as the Raspberry Pi could be mounted directly to the EOAT, and receive instructions wirelessly.

3.2.3 Electrical

As noted in the previous section, while the PLC is an industry-recognized stable solution to the generation of the PWM signals, it necessitates transmission of 6 separate data lines. In addition to the necessary digital signals controlling the stepper motors, the EOAT has 4 limit switches to detect an erroneous position of the manipulators and provide feedback that must return to the high-level controller, adding an additional 4 data lines across the QC-11. When adding the power supply for the stepper motors this creates significant issues routing signals as the existing electrical passthrough can support 8 unique wires and our initial design required 10 (omitting power). In addition, each electrical connection is only rated for 3 amps at 12 volts, requiring us to split the power across 6 lines (3x12v, 3xGround) resulting in a staggering 16 unique electrical connections required as seen in Figure 6

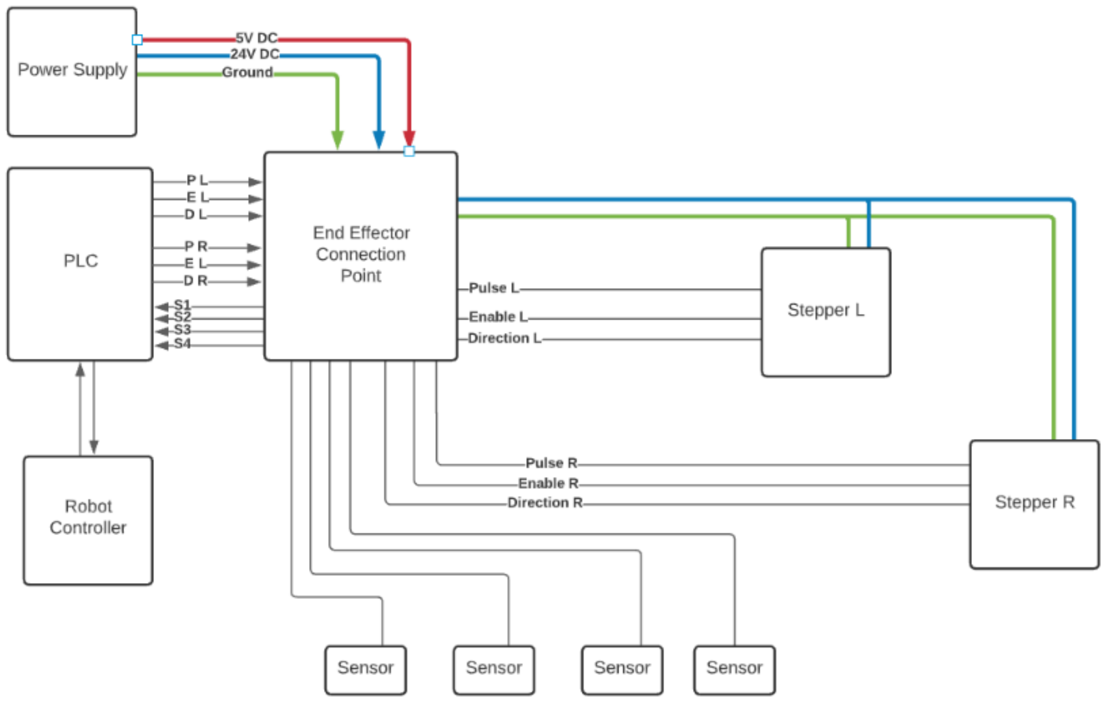


Figure 6: Previous Team's Wiring Diagram

By switching to a Raspberry Pi with a pair of serial communication lines, we were able to reduce the required electrical connections by 6 by routing the 10 data lines from the sensors and motor controllers to the Pi, before forwarding 4 data lines for the Serial Communication protocol. We also removed the need for 4 connections for power by running these lines through an external robust high-power connector (XT-90), thus enabling us to use the existing 8 pin passthrough as seen in Figure 7. We then further refined this design by switching from the native IRC5 programming language of Rapid to a Robot Operating System (ROS) stack, where multiple ROS nodes communicate with each other over the network or Wi-Fi. By running a ROS node on the Raspberry Pi, we could eliminate the need for physical control signal passthrough via the

QC-11 in favor of ROS Messages passed over the network. With this model, we only require 12v and ground as seen in Figure 8

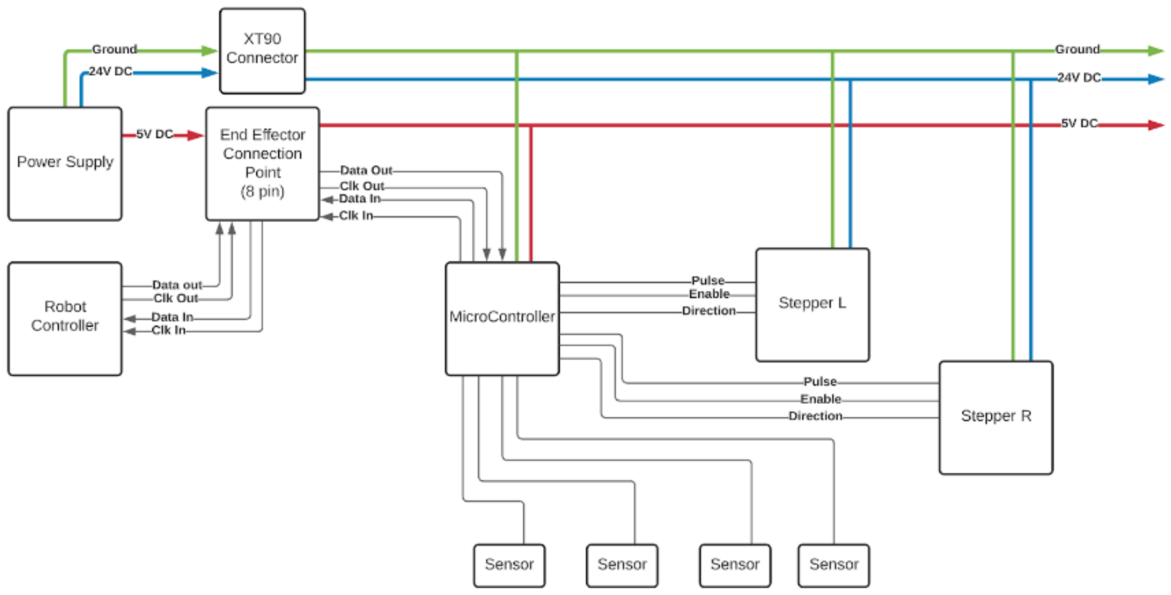


Figure 7: Proposed Updated Wiring Diagram 1

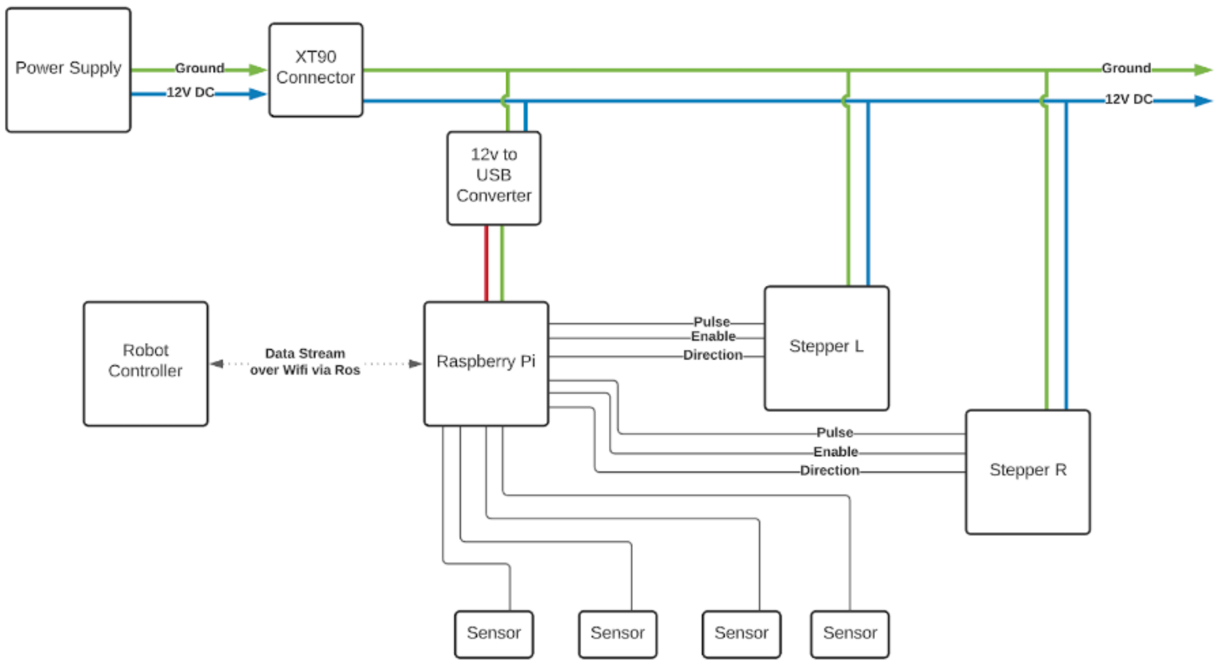


Figure 8: Proposed Updated Wiring Diagram 2

3.2.4 Mounting

With the decision to utilize a Raspberry Pi to control our end of arm tooling we needed a way for it to be mounted on to the end effector. To do this we designed a case for the pi that would attach on to the end effector as well as providing adequate venting for the pi. Because the Pi tends to run hot, it was necessary that the casing had an open top and vents on the sides and bottom to allow for better heat flow. Along with the Pi we also had to account for the Lidar camera, which needed to be attached to the end effector. It was found that last year's MQP group had created, and 3D printed a mount for the camera that could be attached to the end effector. Since the mount had a decent amount of space on it, the possibility of mounting the pi behind the camera was brought into consideration.

3.3 Software Design

3.3.1 Design Considerations

This project requires a handful of subsystems to work reliably and together. The largest constraint is that any solution that is used must be compatible with the ABB IRB 1600 robotic arm. The IRB 1600 is controlled by the IRC5 robot controller which runs a RobotWare operating system and can be controlled with ABB’s RAPID programming language. Other compatibility constraints include the Intel RealSense L515 Lidar camera and the EOAT. Fortunately, the RealSense camera is officially supported on many platforms and there is substantial documentation on these platforms. Unfortunately, the EOAT is not as compatible due to its custom design. Some manner of interfacing with stepper motor controllers was necessary through the development of proprietary code. To allow the project to meet deadlines, a modular, robust, and version controllable system is desirable.

One option that was considered was using a PLC and RAPID code, there was already a minor amount of work done towards this solution (the system was wired up). It was decided not to go for this solution for a few reasons. Firstly, using the PLC required a large number of data lines (See 3.2.3 Electrical). Secondly, no group member had any experience with using a PLC or writing RAPID code. This lack of knowledge was made worse by the fact that the previous group was having problems with the PLC Citrio module, which oversaw generating the pulse train signals for the EOAT. The idea of debugging an unknown system led us to explore other options. On top of that, there was also no supported RAPID API for the Intel RealSense L515 Lidar camera.

The other option we came across was using the open-source robotics middleware suite Robot Operating System ROS. This system was used by the team for multiple reasons. Firstly, being that multiple team members had taken classes that used ROS, and thus were experienced with it. Additionally, there was an ABB IRB 1600 package, Intel RealSense package, and it was easy to write our own packages. ROS also has the advantage of being able to work and communicate between many different machines, over the internet and/or LAN (IPv4 addresses). This will allow computationally intensive tasks to be done on more powerful machines or even at a remote site. ROS can also remove the need to wire subsystems together physically and just control them over Wi-Fi.

3.3.2 Robot Operating System

ROS is made of three groups each to be run on a different machine, which is depicted in Figure 9. The Main Computer is responsible for the bulk of the computational tasks, because it is the most powerful machine in the system. The Main Computer is tasked with the path planning of the arm, sending target positions to the Raspberry Pi, reading the point cloud from the LIDAR camera, and deciding where and how much to deform the plasticine. The deformation logic is depicted in Figure 10.

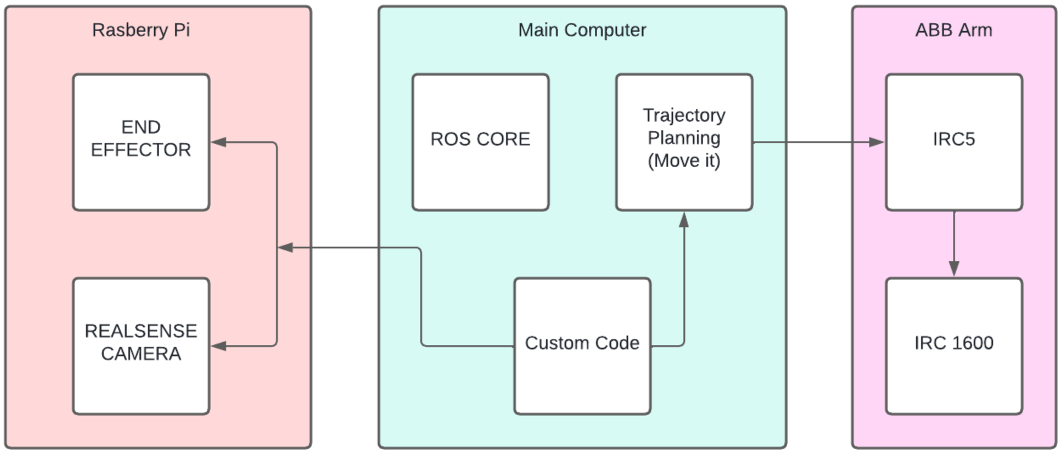


Figure 9: The ROS Flow Chart

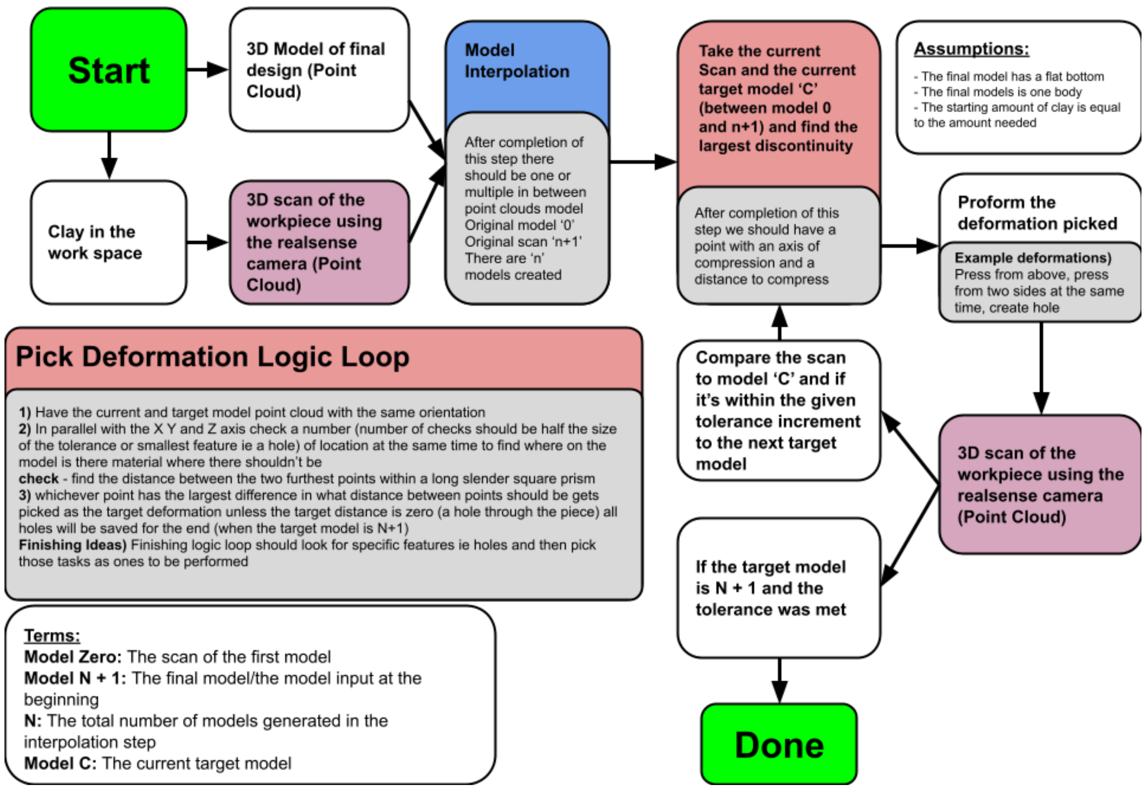

Start
3D Model of final design (Point Cloud)
Model Interpolation
After completion of this step there should be one or multiple in between point clouds model Original model '0' Original scan 'n+1' There are 'n' models created
Take the current Scan and the current target model 'C' (between model 0 and n+1) and find the largest discontinuity
After completion of this step we should have a point with an axis of compression and a distance to compress
Assumptions:
- The final model has a flat bottom
- The final models is one body
- The starting amount of clay is equal to the amount needed
Clay in the work space
3D scan of the workpiece using the realsense camera (Point Cloud)
Proform the deformation picked
Example deformations) Press from above, press from two sides at the same time, create hole
Pick Deformation Logic Loop
1) Have the current and target model point cloud with the same orientation
2) In parallel with the X Y and Z axis check a number (number of checks should be half the size of the tolerance or smallest feature ie a hole) of location at the same time to find where on the model is there material where there shouldn't be
check - find the distance between the two furthest points within a long slender square prism
3) whichever point has the largest difference in what distance between points should be gets picked as the target deformation unless the target distance is zero (a hole through the piece) all holes will be saved for the end (when the target model is N+1)
Finishing Ideas) Finishing logic loop should look for specific features ie holes and then pick those tasks as ones to be performed
Compare the scan to model 'C' and if it's within the given tolerance increment to the next target model
3D scan of the workpiece using the realsense camera (Point Cloud)
If the target model is N + 1 and the tolerance was met
Terms:
Model Zero: The scan of the first model
Model N + 1: The final model/the model input at the beginning
N: The total number of models generated in the interpolation step
Model C: The current target model
Done

Figure 10: Pseudo-Code Flow chart for custom code

To generate and send valid trajectories to the IRB 1600 a few ROS packages were used, these include Moveit, abb_robot_driver, abb_experimental, abb_irb1600_6_145_support and ROS industrial. Moveit is a package designed to control robotic arms, it handles all the trajectory planning and speeds, comes with a graphical interface to control the robot. Moveit can use both python and C++ modules that can interact with the move group node in the Moveit package. Moveit is responsible for generating paths for the IRB 1600 arm as well as reporting the robot's current state over tf (transform) messages.

In order to send the planned trajectories to the IRB 1600, the abb_robot_driver package is required. This package communicates over a local wired network with Robot Ware code installed as a system on the IRC5. The exact arm model used with the system is not supported by the abb_robot_driver package so the abb_experimental package is used because it supports our specific IRB arm along with a Moveit configuration for the 1.2-meter version of the IRB 1600. A modified version of the 1.2 meter package was used to control and communicate with the arm it is called abb_irb1600_6_145_support. This package was designed to reuse configurations from the abb_experimental package, where there were no differences, and have custom files for to reflect the required changes. The ROS Industrial package is required to make both ABB driver packages work, and creates the underlying structure for the Moveit package.

The custom code on the main computer works by using model interpolation and a closed control loop, which are both outlined in section 3.3.4 Model Interpolation and section 3.3.5

Outlier Detection. These two processes work together to determine where the robot should deform the plasticine and correct any unpredicted deformations.

The Raspberry Pi is responsible for taking a EOAT movement command from the main code issued by a ROS Message on a ROS Topic and generating the signal necessary to control the stepper motor drivers and move the manipulator fingers to the desired position as detailed in section 3.3.3 End of Arm Tooling Code. In addition, the Raspberry Pi takes the point cloud from the L515 RealSense camera and sends it back to the main computer.

The ABB arm is responsible for executing the trajectories sent to it by the main computer. To achieve this ROS is installed on the IRC5, which controls IRB 1600. The IRC5 is configured to listen for incoming trajectory messages as a background task, and then there is RAPID code from the abb_robot_driver package that when run on the robot turns the received trajectory into motion.

3.3.3 End of Arm Tooling Code

The EOAT manipulation code is a state machine-based loop running on the Raspberry Pi mounted to the tooling. Originally written in Python 3 due to its high compatibility with ROS and the Raspberry Pi, we found that while it was easy to implement initial features, the code in python quickly became unwieldy due to the timing necessary. Since python is a high-level programming language, it simplifies many functions for the end user at the expense of versatility and fine control, though it compiles down into C or C++. To regain high control over our manipulators, we reworked the code to utilize C++.

We created a manipulator object responsible for knowing its positional limits and the individual move and calibrate commands. Further a tooling object contained two instances of a manipulator: the left and right side. Together, this allowed us to have high level control over both manipulators while maintaining quick control over erroneous conditions such as a user initiated emergency stop, or a manipulator exceeding its intended physical position.

The program is set up such that the developer can choose to require user input to set up the EOAT each time, or automatically setup the EOAT and proceed directly to listening for instructions from another ROS Node. This process is determined in code and cannot be changed after compilation. In the manual mode, which is recommended for persons new to the system, the code will prompt the user to verify the wires are connected appropriately, ensure the area around the EOAT is clear and request permission to proceed before each step. The automatic mode assumes the user has preemptively verified everything is setup appropriately and directly sets up the IO, calibrates the manipulators and immediately begins listening for commands.

The bulk of the code is a continuous loop of checking for new instructions and executing them, verifying at each step that no errors have occurred and transmitting a status update every second via ROS topics (see Table 1 and Table 2). In the event an error has occurred, the issue is transmitted to the higher-level ROS Node and the EOAT recalibrates itself before waiting for new instructions. While any motions are in progress, the Raspberry Pi monitors 4 GPIO inputs connected to limit switches for a manipulator exceeding its design limits. Using the Pigpio library, these lines are monitored using an interrupt service routine to ensure all signals are caught. Upon detecting a tripped switch, the code verifies the validity of the signal by checking that the switch is still in a tripped position 0.1 seconds later. This was an important check as we discovered that the limit switches are not perfect and tended to occasionally trip when the robot was moved.

Table 1: EOAT ROS Messages

Incoming Commands:

Point32 message on topic "mmm_eoat_commad"

Field	Use
x	Left Manipulator Position (mm)
y	Right Manipulator Position (mm)
z	Speed (mm/s) or Command

Outgoing Notifications:

Point32 message on topic "mmm_eoat_position"

Field	Use
x	Left Manipulator Position (mm)
y	Right Manipulator Position (mm)
z	Speed (mm/s) or Command

Once a manipulator is verified to have exceeded its limits, the EOAT disables both manipulators, and reports the error. After reporting the error, the EOAT backs the manipulator off, and then recalibrates. This state machine diagram can be found in Figure 11

We also made extensive use of the C++ signal library to raise the appropriate signals and alarms for executing sensitive or elevated portions of the code as necessary in unusual circumstances. Since changing variables, or even some interrupts were dependent on various portions of code running in a certain order, we elected to raise signals, which must be handled appropriately, or else crash the program. We used SIGABRT to execute an emergency stop as it kills all related functions and terminates the program immediately, and SIGALRM to send a message to the controlling ROS Node every 1 second. Of note though was the necessity of overriding the ROS and Pigpio signal handlers, as each library dealt with certain signals in ways that conflicted with our own signals.

Table 2: EOAT Codes

Command Codes

Code	Meaning
-911	Execute ESTOP
-732	Calibrate EOAT
-111	Set Tool Offsets

Status Codes

Code	Meaning
0	Waiting for Instruction
1	Executing Instruction
2	Tooling exceeded limits
3	EOAT Stopped
4	EOAT Initializing
5	EOAT Testing
6	Tool offsets set to xy
7	Invalid Command
8	EOAT Calibrating
9	Error
911	EOAT ESTOPed

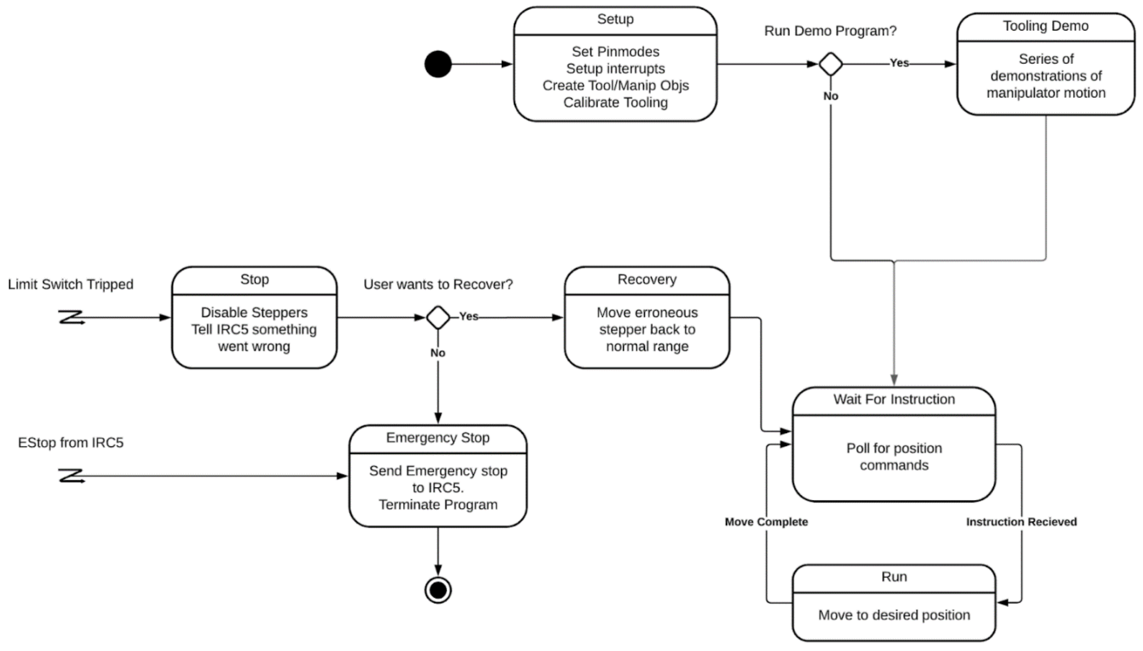
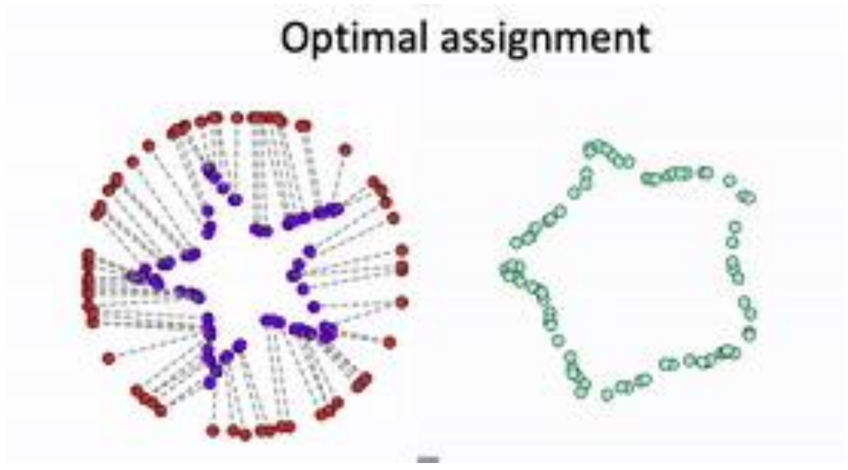


Figure 11: EOAT Code Flowchart

3.3.4 Model Interpolation

To create the desired forms of plasticine, the code needs goals for which to serve as its milestones for outlier detection. These goals help converge the initial shape into the final, desired shape. Rather than attempting to go from the initial condition straight to the end goal, it would be more efficient to develop intermediate goal points that the robot will target in a stepwise process. These iterative goals provide smaller changes and differences to notice. This allows us to target the issues effectively and prevent issues caused by drastic changes that appear due to fewer models. To generate these target models, we decided that model interpolation is the preferable method.



.

Figure 12: Demonstration of Interpolation of a 2D Pointcloud

Model interpolation provides an automated means of generating these intermediate goal models from a given start and end configuration for the plasticine. As shown in Figure 12, to reach the red end configuration from the purple start configuration, each point needs to be attached to its closest corresponding point. This can be seen by the dotted lines between the red and purple dots, which represent the start and goal configurations respectively. Once connected optimally, finding a midpoint between them creates as shown in the teal intermediate model. This allows for a smoother transition between the start and end. The interpolation program then would provide a given number of models, or in the case of this robot, point clouds that when iterated upon make a smooth transition from start to end shape, creating goals which robot can target.

The model interpolator functions by generating a third random point cloud between the two models that need to be interpolated. The algorithm would then calculate the earth mover's distance, or EMD, between the third point cloud and the first two, which is a metric of the distance between two probability distributions [13]. By summing those two EMDs, and minimizing that sum, the program finds the point cloud that is exactly between the original point clouds, which would be the point associated with the optimal assignment from Figure 12, and this process can be iterated upon as many times as necessary to generate the desired number of steps.

3.3.5 Outlier Detection

One major advantage that metamorphic manufacturing has over other manufacturing methods is the ability to correct issues on the fly. We can use our RealSense camera to send a point cloud to the computer and figure out at what location or face there seems to be an outlier. Figure 13 is an example of what an outlier might look like in a part during forming. These outliers may happen given the fact that not all movements and measurements will be perfect, and we do not expect them to be, so we designed this to catch these issues.

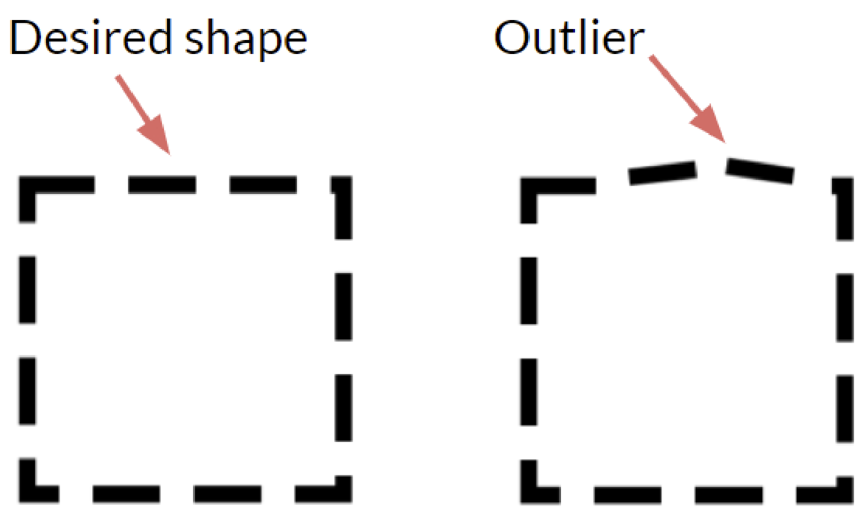


Figure 13: A Visual representation of an outlier compared to a desired shape

The point cloud is sent to and read by a python program that will return a location or face for the robot to fix. This is done by checking each point in the cloud and seeing if they fall outside of a given range specific to the shape that the robot is creating. Each outlier is tracked and will be sent to the robot. Each outlier in the list can also be returned to the robot to fix each one in the same step.

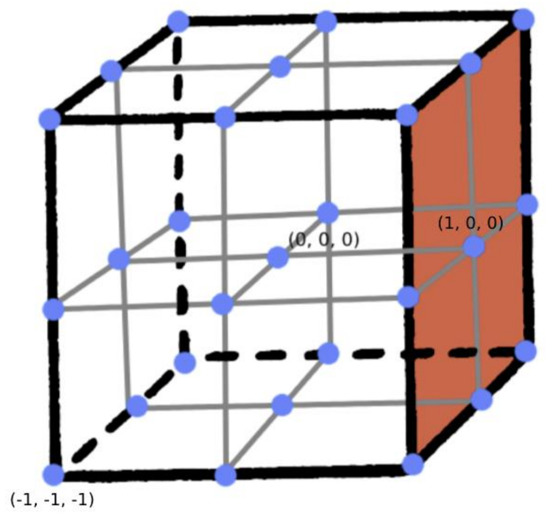


Figure 14: A visual representation of a 2D array of a cube

Figure 14 is an example of a cube with 27 points, represented by a 2d array that the program can read. Each point has its own array of 3 values, and each point is read one by one by the program. The red face is a boundary that would be checked by the program, in this example, it would check if any value in the first position of the array is greater than 1. This would see if any point fell outside of the right face of the shape. If it does, the program adds the point to a list of out of bounds values that can be returned to the robot or the user.

3.3.6 LIDAR Camera and 3D Scanning

For getting real-world data back into the system an Intel L515 RealSense camera was used. The L515 is a Lidar camera that can be integrated with the ROS system that we have been using for the rest of the robotic system. The L515 can generate a series of point clouds that can be stitched together to make a more accurate scan of the workpiece. This is done by processing the point clouds

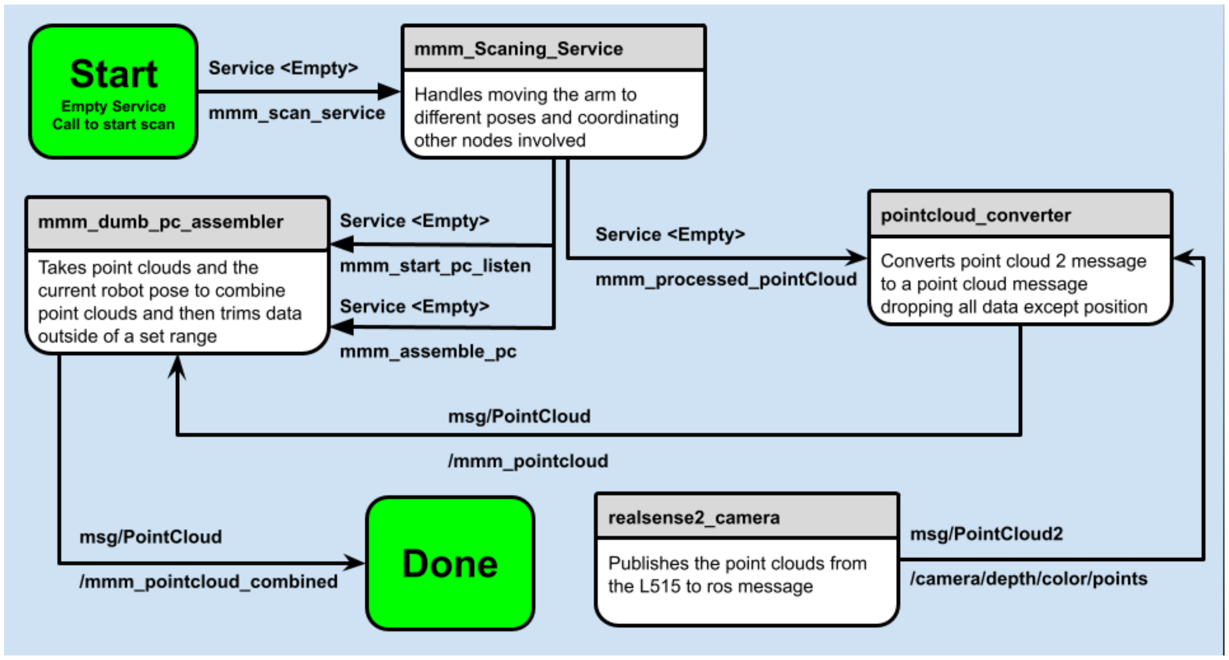

Start
Empty Service Call to start scan
Service <Empty>
mmm_scan_service
mmm_Scaning_Service
Handles moving the arm to different poses and coordinating other nodes involved
mmm_dumb_pc_assembler
Takes point clouds and the current robot pose to combine point clouds and then trims data outside of a set range
Service <Empty>
mmm_start_pc_listen
Service <Empty>
mmm_assemble_pc
Service <Empty>
mmm_processed_pointCloud
pointcloud_converter
Converts point cloud 2 message to a point cloud message dropping all data except position
msg/PointCloud
/mmm_pointcloud
msg/PointCloud
/mmm_pointcloud_combined
Done
realsense2_camera
Publishes the point clouds from the L515 to ros message
msg/PointCloud2
/camera/depth/color/points

Figure 15: Scanning Code Flowchart

The scanning process works as depicted in Figure 15. A scan can be requested at almost any time by making a call to the mmm_scan_service, once the mmm_Scanning_Sercive node receives this service request, it makes a service call to the mmm_start_pc_listen service. This tells the mmm_dumb_pc_assembler to be ready to save incoming point cloud messages that it receives. After telling the assembler node to be ready the scanning node starts moving the camera to poses prepicked to maximize the coverage of the workpiece. Each time the camera gets into the correct pose the scanning node sends an Empty service request to the point cloud converter, telling it to convert then publish the most recent point cloud received from the realsense2_camera node. The assembler node continues to collect the converted point cloud messages until it finally receives a service call on the mmm_assemble_pc service, at which point it combines every saved point cloud, removes the points outside the work area, converts the point reference frame to the robot base and publishes the point cloud.

This scanning works with the model interpolation as well as with the outlier detection subprograms. The first scan generates the initial point cloud for the model interpolation to work with, generating intermediate targets for the overall program. The outlier detection then produces a point cloud that will be compared to the target model made using the model interpolation, and by comparing these two point clouds we can pick the outlier.

3.3.7 Subsystem Communication

By using ROS, we are able to connect all subsystems to each other over the network through ROS Topics. To connect the EOAT and main control loop, we elected to use a bidirectional communication line comprised of two topics, each using the ROS Point32 message. The main control loop issues directions to the EOAT with the x and y coordinates of the message being used as positions for the left and right manipulators respectively, and the z coordinate being used as a code to indicate the instruction. For example, the code might indicate the positions are tool offsets, or that recalibration is necessary. A status update from the EOAT is regularly published via a return message, where the x and y coordinates represent the current position, and the z coordinate again represents a status code, ranging from busy executing an instruction, waiting for instruction, or various errors. In addition to the regularly scheduled status update every 1 second, this line is automatically published with a status in any unexpected circumstance.

3.4 Tooling

3.4.1 Design Considerations

To perform the necessary deformations in the plasticine clay, new tooling was needed. The process began with analyzing the original tooling made for the end effector and understanding the flaws associated with the design. The flaws associated with the original tooling hindered the performance of the tooling and effectiveness in reshaping the clay into the desired shape. Once the flaws were identified, solutions were theorized and generated via Computer Aided Design (CAD) software. The new tooling was then tested and redesigned, when necessary, to improve performance and reliability.

When applying pressure via the tooling onto the plasticine clay, it was found that the clay would stick to the plastic surfaces when attempting to pull the tooling away from the clay. This not only removed the entire clay block from the work area but also prevented the tooling from continuous deformations. This flaw was critical to the original tooling and highlighted a much-needed key feature in the new tooling. To prevent the clay from sticking to the tooling, the tooling would need to have an extremely smooth surface with no blemishes on the surface that will contact the clay. Sanding the surface to a fine grain is a quick and cost-effective solution to smooth the surface, however, as it was found through testing, regardless of the smoothness of the surface, plasticine clay continued to stick to the PLA plastic. Another method theorized was using tempered glass to contact the clay as the glass has a highly polished surface. However, the ability to reshape the glass into the necessary dimensions to attach to the tooling would prove to be far too costly and only hinder the development of the tooling. After researching effective methods of preventing the sticking of clay to all surfaces, it was found that printer paper easily prevented the clay from sticking to surfaces and can be easily adapted to the shape and surface of the new tooling.

The original tooling consisted of a pair of gripper jaws that would, in theory, be used as a universal tool changer to hold and manipulate the tooling for different shaping functions. It was found that these gripper jaws were highly ineffective due to the rounded edge and lack of contact with any of the original tools. The jaws had no form of preventing the tools from rotating when applying pressure to the clay and served no useful purpose. The redesign of the gripper jaw needed to include the following changes: enhanced contact between the tool changer and tooling, easy interchangeable connection between tool changer and tool, and no contact with the plasticine clay. Taking these changes into consideration, the gripper jaw was fully redesigned and named the Universal Tool Changer, allowing simplified connection between the universal tool changer and the tooling, without compromising surface area.

The original tool rack was designed for accommodating the tooling previously created but needed to be redesigned for the new tooling. The rack needed to be able to function directly with the actions of interchangement, allowing the end effector to easily disconnect and reattach to new tooling that would be housed in each of their own racks. Multiple sizing of racks needed to be created in order to hold the differing sizes in tooling. The tool rack must also have a "universal design" that can be replicated and implemented into the design of the tooling so that each of the tools created can be quickly swapped and repositioned when necessary. However, it can be noted that while the smaller tools can fit into the larger tool racks, the larger tools cannot fit onto the small racks.

3.4.2 Universal tool changer

The universal tool changer serves as the primary support system for all tools modeled for this project and as a link between the functions of the end effector to the deformation of the plasticine. The name "universal tool changer" represents the ability to hold any tool designed for the project, regardless of the deformation task needed to be performed.

When creating the first iteration of the universal tool changer, (Figure 31) the design needed to include a few unavoidable details. To connect the UTC to the end effector, four screw holes needed to be implemented into the base. This design comes directly from the previous MQP teams' model for their original tooling. Other factors included in the design feature the large screw down the center of the UTC, as well as a hexagonal cut out that holds the bolt attached directly to the screw. This, in theory, would strengthen the UTC when force is applied during the deformation process. Additionally, taking other ideas from the previous MQP team, a slide and lock mechanism using tabs built into the tooling would in theory allow the UTC to easily connect and disconnect. However, after printing it was found that the tab locking mechanism would not be viable due to the small 3D printed parts being too stiff to bend around the UTC cylindrical shape and lock into the holes.

To improve in the first iteration, the profile of the UTC was changed to a full cylindrical shape. (Figure 32) The lower profile base of the UTC was kept the same, however the hole placement and whole size was modified. One of the biggest changes of the UTC was the locking mechanism that would attach the tooling to the UTC. The cylindrical cut out at the top of the UTC features a. small, extruded cut that would house a small washer shaped magnet. This design would allow the connection and disconnection of the UTC to the tooling to be both strong and reliable as the end effector/ABB arm would only have to apply enough force to separate both magnets from each other, one being on the UTC and one on each individual tool, which will be described later in this paper. The large screw within the center of the UTC was not changed.

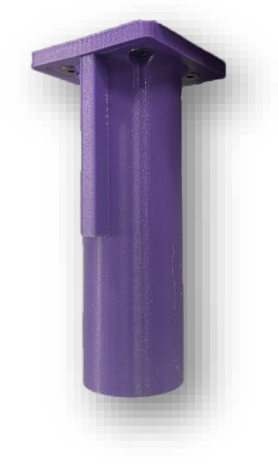


Figure 16: Universal Tool Changer

The final design of the UTC, seen in Figure 16, is the last iteration of the design and the most comprehensive. The cylindrical profile of the UTC was not changed; however, two "key bars" were added. These bars slide directly into cut outs in the tooling and prevent rotation during the deformation process of the clay. This is a major difference between the previous iterations and the final design, as it was one of the main complications that held back the previous design. Additionally, when testing, it was found that the tooling did not have enough support around the base of the cylindrical profile and thus a fillet was added to strengthen the connection.

3.4.3 Manipulation tools

Unique tooling was engineered to create the necessary deformations in the plasticine clay. Each of these tools have individual jobs and are all modeled around the design of the UTC. The tools created were: The press part, the small face press part, the wedge, and the hole punch. Each of these tools fit onto specially designed racks, allowing easy and efficient interchangement.

The press part has two main functions: to squeeze the clay and to push down on the clay. These two tasks are completed using the flat faces built into the side and bottom of the tool. Having both functions on the same tool eliminates the tool changing process of switching from one tool to another.

When designing the first iteration (Figure 34) of the press part, the design needed to reflect the dimensions of the UTC so that it would easily slide onto and connect to the tool. The interior of the tool features small tabs that interlock with the small depressions located on the sides of the UTC. These small tabs have two different slopes on both the top and bottom to allow easy connection and disconnection while keeping a strong hold to the UTC. The lower half of the press part features a much larger section to add extra support when pressing against the clay. In theory, this also allows more surface area to be used when the tool is pushing down on the clay. However, When the part was printed, and the UTC was tested to fit into the tool, it was found the tabs did not flex as intended, did not slide across the surface of the UTC and therefore did not connect at all. In fact, when enough force was applied to try to force the UTC into the tool, the tabs broke which ultimately rendered the tool useless. Additionally, the tabs even broke when the supports were taken off the tool after the printing process which further hindered this design.

The press part's design was then changed to reflect the redesign of the UTC so that the cylindrical profile of the UTC would slide directly into the tool. The new press part iteration also reflected the addition of the magnet system which consisted of an extruded cut into the bottom interior of the new tool that reflected the washer shape of the magnet. (Figure 35) The triangular support system on the lower half of the tool was also changed to strengthen the tool during the deformation process as the lower half is coming in contact the most with the clay. Lastly, two rectangular cutouts are featured on both sides of the tool to reflect the shape of the tool rack that was created to house the tools. These cutouts serve as slide bars that fit directly with the forks of the tool rack to allow easy connection and disconnection between the UTC and the press part. This tool rack will be discussed later in the paper. Once the UTC's design was changed again to reflect the key bars, the design of the press part also needed to be remodeled. The new iteration consisted of having two cut outs on the sides of the connection point to the UTC to prevent the press part from rotating during the deformation process as it was found that when a force was applied to either side of the part on the front face, whether it be the left or the right, the magnetic locking system did not hold the tool in one orientation. (Figure 36)

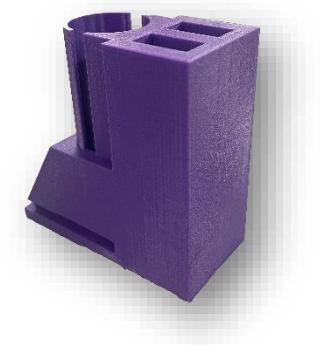


Figure 17: Press Tool

After each additional iteration and change to the press parts design, the final iteration seen in Figure 17 consisted of a reinforced cylindrical profile to prevent bending stresses in the tool when force was applied to the bottom of the front face. When the press part was first tested on the clay, it was found that the end effector was unable to fully close the parts together so that both the front faces were touching, and thus an additional 35 millimeters of "face" was added to the design so that the total distance between both parts when fully closed was zero. The design did not change the washer profile for the magnet and the slide bar cutouts for the tool rack. This press part was also redesigned to have a smaller front face and a triangular profile (Figure 39) to allow smaller deformation processes on the clay, as during testing it was found that the large face of the press part was unable to make smaller changes to the surface.

When looking for a way to create curves and slopes on a block of material, one of the first tools that was conceptualized was a wedge tool. The main function of the wedge tool is to serve to create deformations such as slopes, valleys, and depressions in the molding material.

In designing the first printed iteration of the wedge tool, like the press part, the wedge needed to be designed to be compatible with the UTC. To do this the top part of the wedge used the exact same attachment part as the final press part so that it would fit with the UTC. The lower portion of the part included a rail so that it would be able to slot into the tooling racks. Below this was where the wedge was placed. Initially the wedge was designed to be a triangular prism, which extruded from the bottom surface of the part. In testing the wedge was able to create deformations in the plasticine that it was tested on. However, the deformations being made were much sharper than desired, and the size of the slopes were much too large due to the size of the wedge.

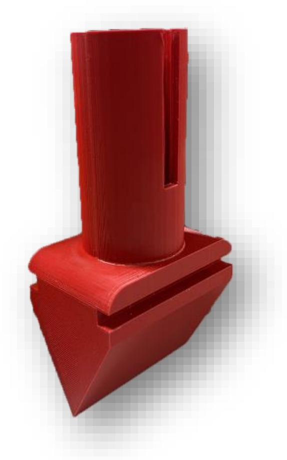


Figure 18: Wedge Tool

The wedge tool's design was reconsidered after the first print was tested. In order to remedy the problems with the first version several aspects of the wedge were changed. Firstly, the size of the wedge was reduced by more than half the size that it was previously. The reasoning for this was that the first wedge during testing had proven to be too large to control effectively for the details that were being attempted. This size reduction in theory would allow for better control while maneuvering the tool around the material. As for fixing the issue of having sharp deformations, the edge of the wedge was made to be round so that the deformations made would be smooth. The rails on the side of the wedge were also widened in order to make inserting the tool onto the tool rack much easier and not as tight of a fit.

The final design of the wedge at present follows the above changes made while iterating on the previous design. The new wedge requires testing in order to determine whether it can succeed in its role in the formation of a horseshoe shape. Based on discussion surrounding the general use of the wedge tool it is possible that multiple wedge tools of various shapes and sizes could be the next step for the tool.

When determining other tools that would be needed in the process of molding a horseshoe, a hole punch was one of the first tools considered. The hole punch serves as a general use tool that can be applied to the molding process of many parts outside of the horseshoe. Its function as it relates to making the horseshoe is to make identical holes on either side of it.

In designing the first printed iteration of the hole punch much like the other tools it needed to be compatible with the UTC. Like the wedge tool the top portion of the hole punch is the end that attaches to the UTC while the tool itself is below that. The hole punch also has the same rails on either side of it that the wedge tool has, which allow it to slide into the tool racks. The hole punch itself is a cylindrical extrusion on the bottom of the tool that then has a conical spike on the end of the cylinder. During testing the hole punch was able to push through the plasticine to create a hole in it, however the hole created was not a uniform circle as intended due to the thickness of the plasticine block. Additionally, much like the wedge tool, the hole punch did not slide into the tool rack easily and was a tight fit. The hole punch did not have any issues being attached to and detaching from the UTC.

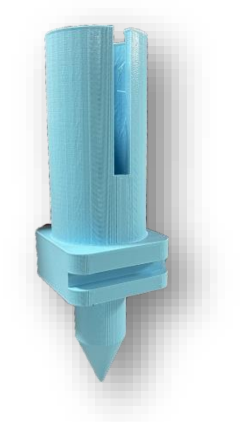


Figure 19: Hole Punch

When thinking of ways to redesign the hole punch the main aspect that was looked at was the ability of the tool to create uniform holes. The proposed redesign was to have the end be one singular conical spike instead of being a cylinder with a spike on the end. The reasoning for this was that with one singular and steeper spike the hole punch could go through one side of the material and then go through the other end to produce a more uniform hole. Upon further consideration this design iteration was thought to only work in specific cases and may not be the best way to create uniform holes. Additionally, the wedge to the rails on the side of the tool were widened to fit the tool racks more easily.

At present the hole punch design is still being iterated on, with the current proposal being reconsidered. It is likely that multiple hole punch variations could be considered for future development depending on the needs of the object being molded.

In order to complete specific tasks for shaping the clay into the horseshoe shape, other tooling was modeled to reflect the shape desired. The “big banana tool” and “Orange slice tool” (See Figure 47 and Figure 48) were modeled to directly influence the bending shape of the horseshoe. The banana would serve as the outer rim of the horseshoe and the orange slice would serve as the inner rim. When combining these tools together and applying force from both sides. These tools in theory would push and form the clay into the rounded horseshoe shape. These tools would also feature the magnetic locking mechanism and would be able to easily slide onto the universal tool changer. Note that in the models in Appendix B do not reflect the design changes suited for the UTC. Additionally, these tools were not fully modeled as they did not reflect the requirement of having a general tool that can perform multiple tasks rather than one specific task of deformation.

3.4.4 Tool rack

To create a way for the EOAT to swap out the needed tools at a given moment tool racks were designed. The purpose of these racks is to provide an efficient way to change out tools during an operation. This is accomplished by providing a space to house the tooling when it is not in use and easy access to swapping out tools.

The first tool rack design considered the existence of the two slots on the end-effector for attaching a tool. Since there are two UTCs on the end effector the tool rack first designed was a rectangular platform with a rectangular wall extrusion from the middle of the platform which had two forks on either side. These forks were designed in tandem with the rails on the tools to be compatible with each other allowing for the tools to slide onto the forks. The concept was that once the tools had been slotted on to the forks, they would be unable to be pulled upward and still be attached to the UTC so the magnets would disconnect. While this would be happening the tool rack would be attached to the operating surface so that it would be locked in place. During tests it was found that the double-sided tool rack was inefficient for swapping out or attaching two tools at once due to its orientation. There was also the issue of taller tools using the racks, since they would be too large for the short size of the first tool rack.

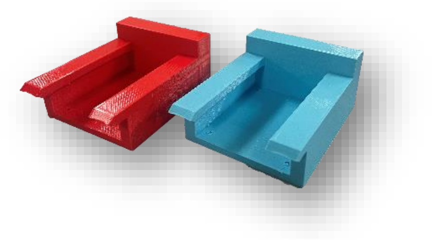


Figure 20: Tool Racks

When iterating on the design of the tool rack two aspects were looked at. The first being the size of the tool rack and the second being the orientation of how the racks were set up. To satisfy both aspects, which the previous issues were derived from the racks were redesigned with these in mind. Firstly, two separate variations of the tool rack were created to accommodate the different lengths of the tools. The initial design was now considered the small tool rack, while a taller variation created for tools like the wedge and hole punch would be the large tool rack. Additionally, the original idea for the racks to be double sided was scrapped in favor of having one sided separate tool racks so that they could be oriented to make tool changing more efficient.

The final tool rack designs remained unchanged after the decision to make two separate variations that were one sided. The only tweaks made to the designs were in the distance between the forks on the racks since there were problems during tool changing tests caused by the tolerance being too tight on the tools and tool racks, and a lack of wiggle room. With these design tweaks the changing of tools should be much smoother than it had been previously during testing.

4 Results

4.1 Clay Deformation

After several iterations of tools, we were able run several full system tests verifying that all components were functional. After verifying the system functionality, we began developing the G-Code (see Section 4.3 Software for why we developed G-Code instead of closed loop control) for manufacturing a horseshoe). We chose a horseshoe (see Figure 21) because it was one of the objects specified by the LIFT challenge (See Section 2.1.3 Metamorphic Manufacturing). We started with a blank of plasticine in the shape of a rectangular prism 40x40x70mm. We determined that the rather than forming the curved horseshoe shape throughout the process, we would elongate the blank into a longer prism in a process called “drawing out” by repeatedly pressing the sides and top of the blank (See Figure 22, Figure 23, and Figure 24). Holes and valleys would then be formed in the bar by the wedge and hole punch tools, prior to bending the whole bar into the final shape.

We found that using the smaller press tools was faster and more effective at moving material during the drawing out process as the angled sides push the material outwards more effectively than the wider press. One downside of using the thinner press was the excess amount of tooling marks compared to the wider press (See Figure 25), however using the larger press caused extra stress to the EOAT due to the larger surface area (See 4.2 End of Arm Tooling).

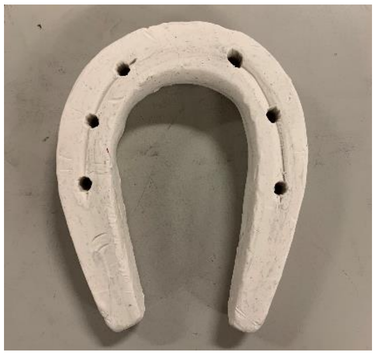


Figure 21: Example Horseshoe

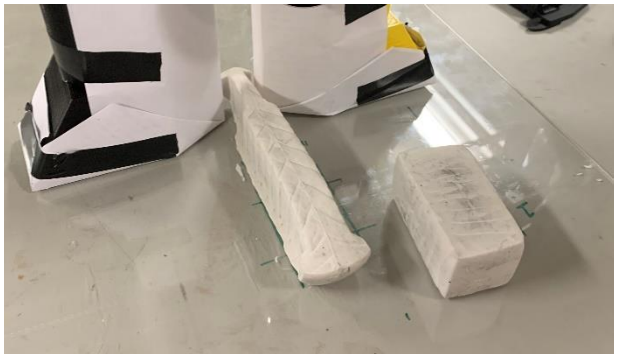


Figure 22: Drawing Out Process 1

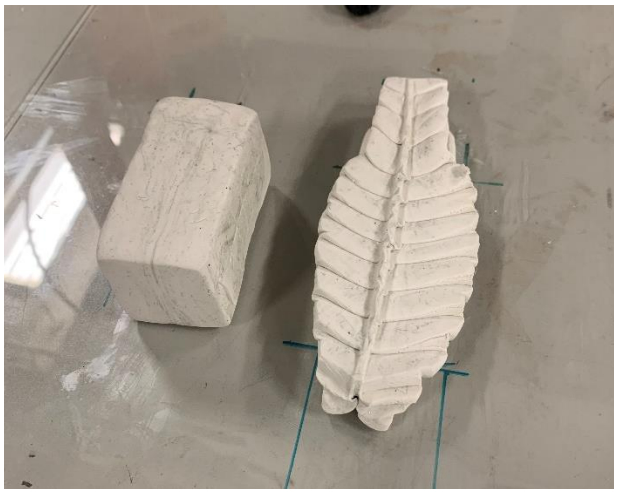


Figure 23: Drawing Out Process 2

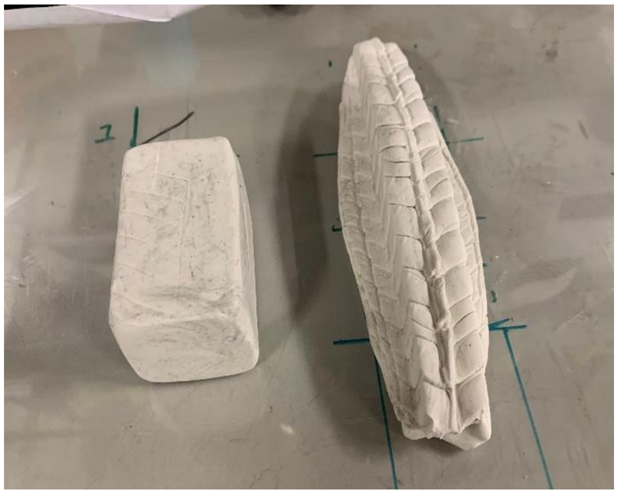


Figure 24: Drawing Out Process 3

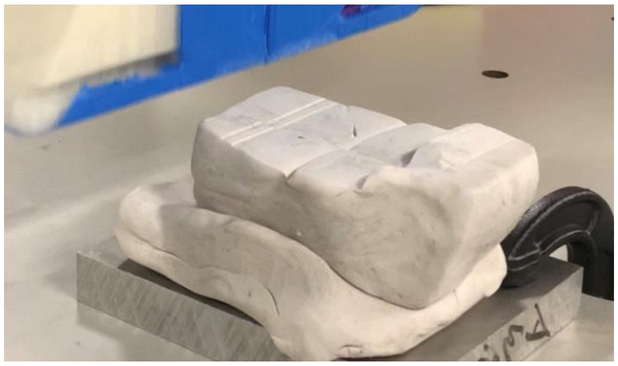


Figure 25: Deformation under Wide Press

4.2 End of Arm Tooling

Despite several persistent intermittent bugs, by the end of the project, we had achieved all our goals for the EOAT, rewiring and neatening up the connections, migrating from an external PLC to a Raspberry Pi, and developing the protocol between the EOAT and main nodes.

During the wiring process, all the existing hookup wire and floating breadboards were replaced with shielded wires and bundled neatly and securely to avoid accidental catching and tangling as the tooling was used. These wires then connect into the Raspberry Pi, which issues stepper motor control signals, interprets sensors data and communicates with the central program, avoiding much of the wiring previously necessary as seen in Figure 4. The updated wiring can be seen in Figure 26.

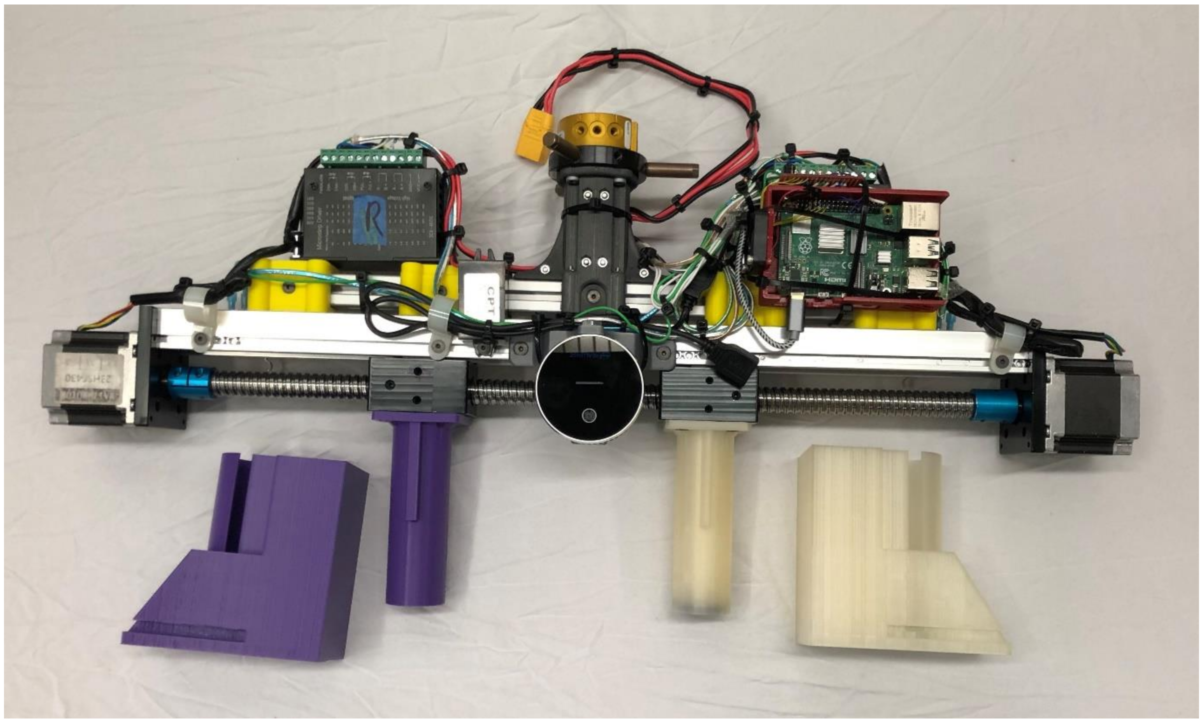


Figure 26: Updated EOAT Wiring

The Raspberry Pi itself runs a State Machine with a built in ROS Melodic Node as described in Section 3.3.4, wherein after conducting the startup sequence, the EOAT repeatedly steps through a loop of waiting for new instructions and executing them. The limit switches at each end of the EOAT drive interrupts to immediately override the current commands if the manipulators attempt to overrun their mechanical limits. A status is published to the central code every second to enable the user to understand what the EOAT is currently doing.

An unexpected discovery during our clay deformation tests was the extreme flexibility of the EAOT frame, and the tendency for it to separate or flex when large forces were applied (see Figure 27). While we were able to mitigate these issues by reducing the deformation area, and frequent recalibration, the issues persist, and we were unable to resolve them over the course of our work and would recommend extensive structural redesign to generate higher deformation forces and tolerate the accompanying reaction forces.

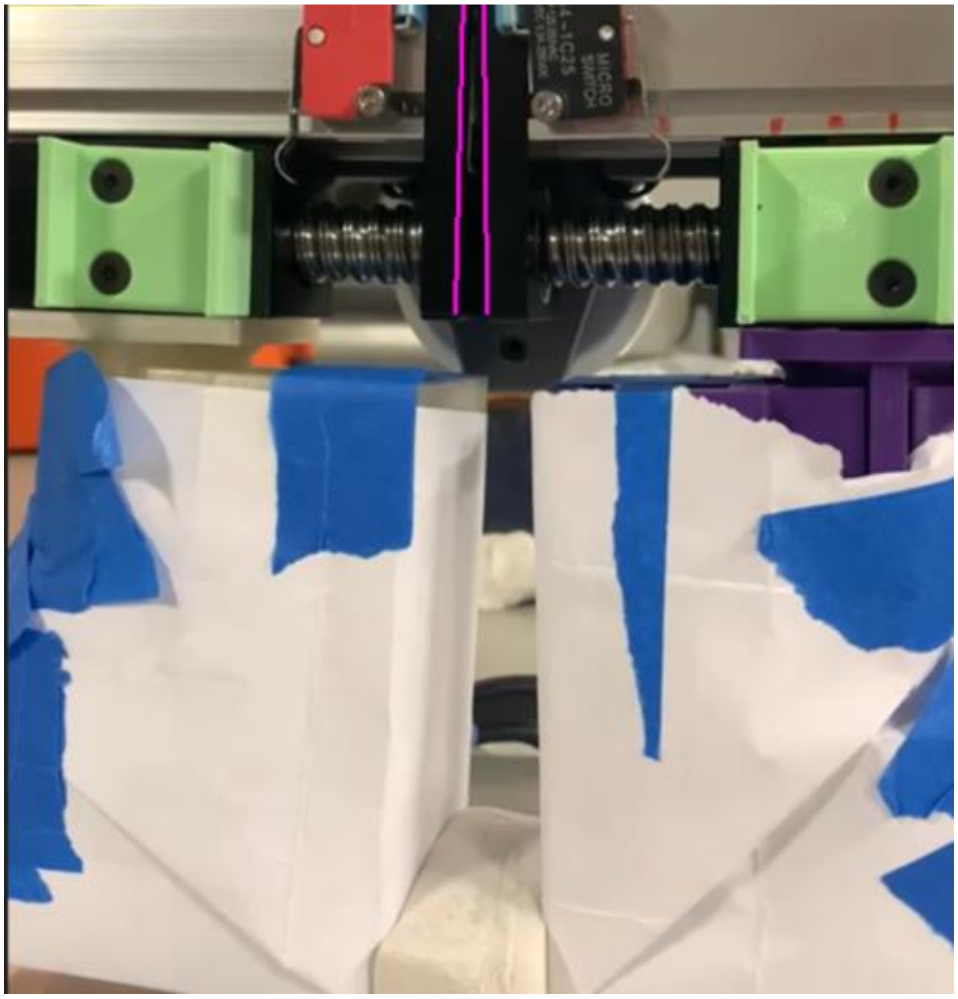
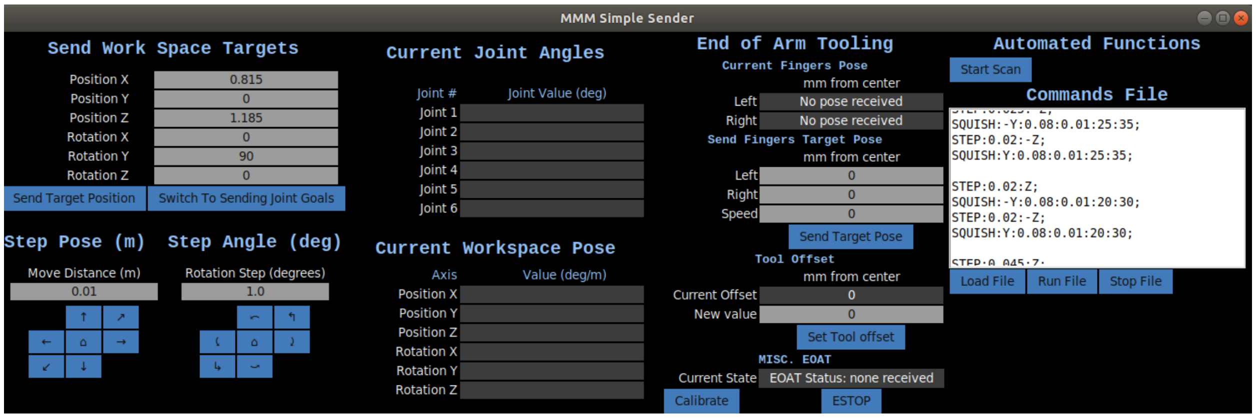


Figure 27: EOAT Flex

4.3 Software

Due to time constraints the closed loop control system was not fully implemented. The outlier detection code was written and the code to control and communicate with the EOAT and ABB robotic arm was Implemented. A unified graphical user interface (GUI) for both the robotic arm and EOAT was made along with a G-code like language referred to as ‘MMM Files’ and is depicted in Figure 28



Position X 0.815
Position Y 0
Position Z 1.185
Rotation X 0
Rotation Y 90
Rotation Z 0

Move Distance (m) 0.01
Rotation Step (degrees) 1.0

Current Fingers Pose, mm from center: Left: No pose received; Right: No pose received

Send Fingers Target Pose, mm from center: Left 0; Right 0; Speed 0

Tool Offset, mm from center: Current Offset 0; New value 0

Current State: EOAT Status: none received

```
SQUISH:-Y:0.08:0.01:25:35;
STEP:0.02:-Z;
SQUISH:Y:0.08:0.01:25:35;

STEP:0.02:Z;
SQUISH:-Y:0.08:0.01:20:30;
STEP:0.02:-Z;
SQUISH:Y:0.08:0.01:20:30;

STEP:0.045:Z;
```

Figure 28: GUI for Full System Control

The GUI is comprised of four main sections that each have their own functionality. The first section responsible for controlling the pose and movement of the ABB robotic arm manually. The second section of the GUI is a readout for the current pose of the arm as well as all the joint angles. The third section of the GUI is used for controlling the EOAT its various functions as well as having a readout for the EOAT. The fourth section of the GUI is used for automated functions such as the MMM file execution.

The first section which is responsible for manual control of the ABB’s position consists of two sections. The first is for entering either target poses of the end of the arm or target joint angles for all the joints. The entries in this section pass from the GUI to the underlying ROS nodes, and covert degrees to radians, the ROS node then commands the ABB to move. The second part of the manual control portion of the GUI is used to step either the position or the rotation of the end of the ABB arm, which works by reading the current position and adding the step size and then commanding the ABB to move to that pose.

The readout section of the GUI works by reading the robot’s current pose in the world space as well as the robots current pose in the joint space, all the units in radians are converted to degrees and then displays that in the corresponding box. In order to speed up performance this process is multi-threaded. The ROS message queue was limited to a single message in order to reduce the readout latency.

As outlined in 3.3.3 End of Arm Tooling Code EOAT commands are all variations of the same ROS message and in order to simplify controlling the EOAT the GUI provides a layer of abstraction for these commands as well as a user-friendly interface. The GUI allows for normal use of the EOAT to be done through buttons and value entries, as well as providing information on the current position current tool offset and current status of the EOAT. The feedback from the EOAT is received over a ROS message and is multithreaded and queue limited just as the read out for the ABB arm is.

The final section of the GUI holds the automated function. Currently there are only two sub sections: the first contains a button which runs a scanning routine, the other section runs the MMM command files. The scanning routine button works by sending a service request to the mmm_Scaning_Service node telling it to start. The scanning proceeds as described in section 3.5. The MMM file execution works by loading in the file to be executed and clicking run file, the file execution can be terminated gracefully at the end of the current command using the stop file button. There are 11 unique commands as well as support for comments. The commands are outlined in Table 3: MMM Commands.

Table 3: MMM Commands

MMM Command	Arguments	Units	Function
POSE	X:Y:Z:rX:rY:rZ	Meters	Move the end of the robot to the pose (X,Y,Z) with rotation (rX,rY,rZ)
JOINT	j1:j2:j3:j4:j5:j6	Degrees	Moves the robot so that each joint is at the angle specified (Much faster than POSE)
EOAT	Left Right Speed	Millimeters	Moves the EOAT fingers to distance (Left, Right) from center for each finger at speed (Speed). Speed of -911 is for Estop. Speed of -732 is for Calibrate. Speed of -111 for setting offsets (might make diff command in the future)
EOATWAIT	Left Right Speed	Millimeters	Same functionality as EOAT but is blocking
SQUISH	Direction Distance Step Size Finger Closed location Finger Open Location	(X, -X, Y, -Y, Z, -Z) Meters: Meters Millimeters Millimeters	Closes the EOAT to the close location, then Opens to the open location, then steps in step direction, and then repeats until the distance squished is distance,

FLATTEN	Direction Distance Step Size Down Step Size Up Step Size	(X, -X, Y, -Y, Z, -Z) Meters Meters Millimeters Millimeters	Moves the EOAT down by Down Step Size, then moves the EOAT up by Up Step Size, then steps in step direction, and then repeats until the distance squished is distance, MOVES RELITIVE TO START LOCATION AND ROTATION
STEP	Distance Direction	Meters	Moves the (X,Y,Z) Pose of the robot by Distance in the Direction. Direction can be (X, -X, Y, -Y, Z, -Z). Negative distances are also supported.
ROTATE	Distance Direction	Degrees	Moves the (Roll,Pitch,Yaw) Pose of the robot by Distance in the Direction. Direction can be (ROLL, -ROLL, PITCH, -PITCH, YAW, -YAW). Negative distances are also supported.
WAIT	Time	Seconds	Have the code wait for (Time) seconds
HOME	None	None	Homes the robot by calling JOINT:0:0:0:0:0:0
SCAN	None	None	Calls the scan service which starts a scan routine
COMMENT	None	None	Allows you to leave a comment in the file to help with readability

4.4 Tools

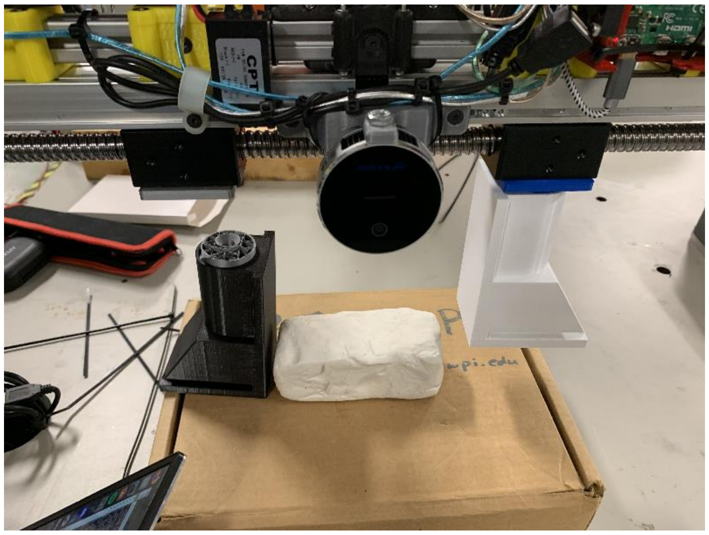


Figure 29: Broken UTC

In the beginning stages of testing, the UTC and press part were put to work and began deforming the clay. However, it was found that upon applying stress on the lower end of each of the press parts, the UTC could not withstand the stresses and ultimately failed (See Figure 29). This failure, seen in Figure 29, depicts the lower half of the UTC detached from the base on the end effector. After analyzing the damage, it was clear that the UTC needed to have additional support surrounding the connection between the shaft and base connected to the end effector. Additionally, there was no nut and screw extending down the center of the UTC which, later on, proved to be a defining factor to the prevention of failure. It can also be noted that the design of the UTC did not have the correct hole placement for the base to connect to the end effector, thus not having a strong anchor point. All of these design flaws ultimately led to setbacks of testing and needed to be addressed accordingly.

To combat the failure, a fillet was added between the base and shaft of the UTC. During the printing process, the wall thickness was increased by 300%, bringing it to 4mm thick. Additionally, the redesign consisted of adding two key bars to the sides of the UTC to prevent the rotation of the tooling during the deformation process, as when tested, the tools were unable to stay straight and aligned. Implementing these changes to the final design proved to be effective in supporting the stresses of deformation when tested.

The press part underwent many changes that effected both the overall appearance and functionality. Early testing showed that the design of the press part was effective in deforming the surface of clay but was limited by the design of the UTC and could not be fully tested at its current state. Additionally, it was found that the press parts were unable to be fully closed due to the limitations of the end effector's design and needed to be extended by at least 35 mm on the side surface. The press part also needed to reflect the changes of the UTC's design and have cutouts for the key bars. Testing also showed stress concentrations centered around the outer back wall that formed the shape of the UTC when forces were applied to the lower base of the press part. This stress is partially due to the tolerance between the cutouts on the press part and the key bars of the UTC, which unintentionally allows too much wiggle room when pressure is applied (See Figure 27) In an effort to reduce this stress, this back wall was increased in thickness, which, when tested, proved to be a good solution.

A secondary sub-part of the full press part was created and dubbed the "small face press part" due to larger pronounced rolling in the surface when deforming the clay using the standard press parts. The smaller face on the front side of the press part allowed more precision and the smoothing of these rolls. One important note about this small face press part was that it surprisingly reduced the amount of strain on the end effector and overall reduced the total flex. The forces exerted by the clay on the small face press part (See Figure 30) was more concentrated, which resulted in less force on the overall structure of the end effector. This small face press part did not have any additional changes to the design other than a smaller surface area on the front face to squish the plasticine.

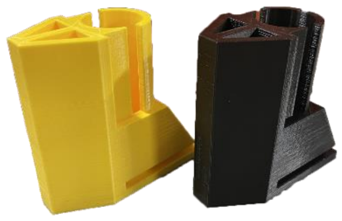


Figure 30: Small Face Press Part

To change out tooling on the end effector, an easy and quick mechanism needed to be implemented into the design to allow minimal or no user interference with the tooling. The first idea to be implemented was a slide and locking system using tabs built into the siding of the tooling. The UTC had sloped cutouts to, in theory, allow the tabs to slide in and out when changing tools. However, it was found during initial testing of the tooling that the tabs were very ineffective and did not flex at all. In fact, when forcing the UTC into the press part specifically, the tabs snapped off. This resulted in the both the press part and the UTC being completely unusable. As a result of this complication, a new system was created as an efficient tool interchangement system. This system, formally named the magnet interlocking mechanism utilizes washer shaped magnets fixed on opposing ends of both the UTC and each of the tooling. The benefit of having magnets as the locking mechanism is that only one force is holding the UTC and the tooling together, and the direction at which the force exists does not exist during the deformation process, meaning the tooling and UTC will not separate unexpectedly, and the act of changing tools must simply overcome the magnetic force. Over the course of the remainder of testing, the magnet interlocking system proved to be a highly efficient method for the interchangement of tools. All tooling was fitted to have a magnet with opposing sides to the UTC.

Additionally, while the rails on the part worked with the tool rack, the tolerances on them were much too tight during testing, which resulted in a UTC being broken. Aside from these results the wedge tool fit well into the UTC and there were no difficulties attaching and detaching it from the UTC.

5 Conclusion and Future Recommendations

5.1 Conclusion

Over the course of the project the team was able to improve on the previous work. By reworking the electrical systems on the EOAT to use a raspberry pi, that sub-system was able to be programed and made functional. With a working EOAT the tools were able to be tested and iterated on leading to improvements in function and design. Having a functional EOAT, allowed the ABB and EOAT to be tied together into a single system using ROS and a GUI to control it all was able to be implemented. The completed system allowed for further testing and improving of the sub systems and tools. On top of the complete system's code a G-Code like language was written to control the robot and EOAT. All the progress allows for future teams to continue to develop and improve upon the metamorphic manufacturing dream.

5.2 End of Arm Tooling

During this project, we discovered several major flaws in the EOAT design, both mechanically and in software. On the software side, we noticed that the lead screws tended to drift as the stepper motors skipped steps, especially when the EOAT was applying a lot of force, leading to inaccuracies in deformation over time. For many of these situations, we were able to sidestep the issue by frequently recalibrating the tooling, however frequently recalibrating consumes valuable time, and is cumbersome. Our recommendation for dealing with lead screw drift is to transition away from dead reckoning and counting steps on the motors, to an absolute position system, anchored to the extrusion connecting the two leadscrews. We envisioned this change to either involve a linear resistor, or a hybrid system using a high precision encoder to record the movement relative to the calibration positions as defined by the limit switches. This would also have the benefit of eliminating the limit switches as a frequently used device and transition them to a purely safety-related sensor. Some refinement of the software limitations for manipulator positions can also be integrated into these sensors to prevent unsafe motion where manipulators collide with each other due to their size.

On the hardware side, related to the drift of the lead screws and stepper motors, we discovered that when the manipulators drift, the tools tended to drift close to the center, and when closing the manipulators to perform a vertical press, had the potential to tear the EOAT apart as the tools exerted their full pressure against each other. Fortunately, the leadscrew pairs are not physically attached to each other, and merely separate, preventing catastrophic damage to the system, this situation does require the operator to manually readjust the leadscrews to close the gap created. A similar issue is presented when using the wide presses to achieve a better surface finish, where the reaction force upon the tools flexes the whole tooling. Although the first situation would be resolved with the update to the sensors and software as described in the previous paragraph, the second issue persists, and we would recommend a redesign of the EOAT to be stronger and more rigid, as well as attaching the lead screws to each other to prevent separation.

5.3 Code

Regarding the GitHub for ease of use and to leave the GitHub in state that represents this team's work, it is suggested to fork the repository and work from the forked repository. There is also the option of forming a GitHub organization so each section of the code can go into a different repository instead of a branch for each section.

For scanning it is not recommended to continue using the dumb point cloud assembler and move to more intelligent techniques such as registration and using the Point Cloud Library, there is some work in the point_cloud_converter package. There is also the matter of the camera and its current mounting, it needs some work and would benefit from being removed from the EOAT or placed more purposely.

For the outlier detection program, there are a few things to be done in the future. It has been tested with the files produced by the camera, but it has not been implemented to directly receive the files. Since it's not implemented, it does not send a point to the robot either. Once these are implemented with ROS, the robot, and the raspberry pi, the system should be one step closer to completion. The last thing is figuring out the bounds for the shape that is being calculated. The square is a simple example to demonstrate how the program works but it will need more technical bounds to accurately calculate shapes with angles and curves.

5.4 Tools

To improve the future tooling for this project, a few changes need to be implemented into the design. When using the current tooling, it can be noted that only the bottom half of each of the tools is used during the deformation process, while to upper portion remains unused. This not only adds an unnecessary amount of stress on the end effector, seen by the flexing, but drastically increase the print time for new tools. Therefore, it is recommended that the tooling be reduced in size by half of the current height, which would theoretically reduce the stresses on the end effector when applying pressure to the plasticine.

All the printing processes for the tooling were done using 3D printers and PLA plastic, which is a simple path to creating a viable solution. The print quality isn’t necessarily lacking regarding accuracy and strength but printing solely in PLA is not the only solution. There are multiple ways to produce these tools and all options can be explored. Recommendations for improving the quality of the parts by researching other printers on campus, such as the Mark Forge printers located in Higgins Laboratories.

Using the materials at hand, printer paper proved to be a very efficient anti-stick material against the plasticine. This is a quick simple and cheap solution to prevent the tooling from sticking to the plasticine during the formation processes. The paper can be easily attached to the tooling and molding to the correct measurements with ease. However, the paper wears out the more it is used, thus becoming less nonstick and the tooling starts to stick more to the clay. Therefore, it is recommended to find another permanent solution such as using Teflon spray, which would coat the exterior of the tooling surfaces that would come in contact with the clay.

In order to deform the clay into the necessary rounded shape of the horseshoe, they clay would have to be either picked up and bent around a rounded object, or formed into the shape using tooling. Though not tested, the banana and orange slice tools could have been a simple solution to the problem, but it cannot be concluded. Therefore, it is recommended that new tools be created to satisfy this task. One potential idea to explore would be using a single and double prong tool, each on a different UTC respectively. By applying pressure from one side using the single prong tool and pushing the clay against the siding of the two prong tool, this in theory, would act as a bending tool and deform the clay into the crude shape of a horseshoe. These tooling ideas are based off real world bending tools that add deformities into hot or cold metal in order to bend the metal into the required shape. A prime example of this is a hydraulic press bending tool, which uses two stationary rollers on the base and a single roller attached to the press. The metal rod would be positioned on the base rollers, then pressed down upon by the hydraulic roller, forming a bend in the rod. This same concept can potentially be replicated, but using a horizontal pressing action.

Appendices

Appendix A: Glossary of Terms

CAD – Computer Aided Design

CNC – Computer Numeric Control

EMD – Earth Movers Distance

EOAT – End of Arm Tooling

GPIO – General Purpose Input Output

PLC – Programmable Logic Controller

PWM – Pulse Width Modulation

ROS – Robot Operating System

UTC – Universal Tool Changer

Appendix B: CAD Iterations

Universal Tool Changer

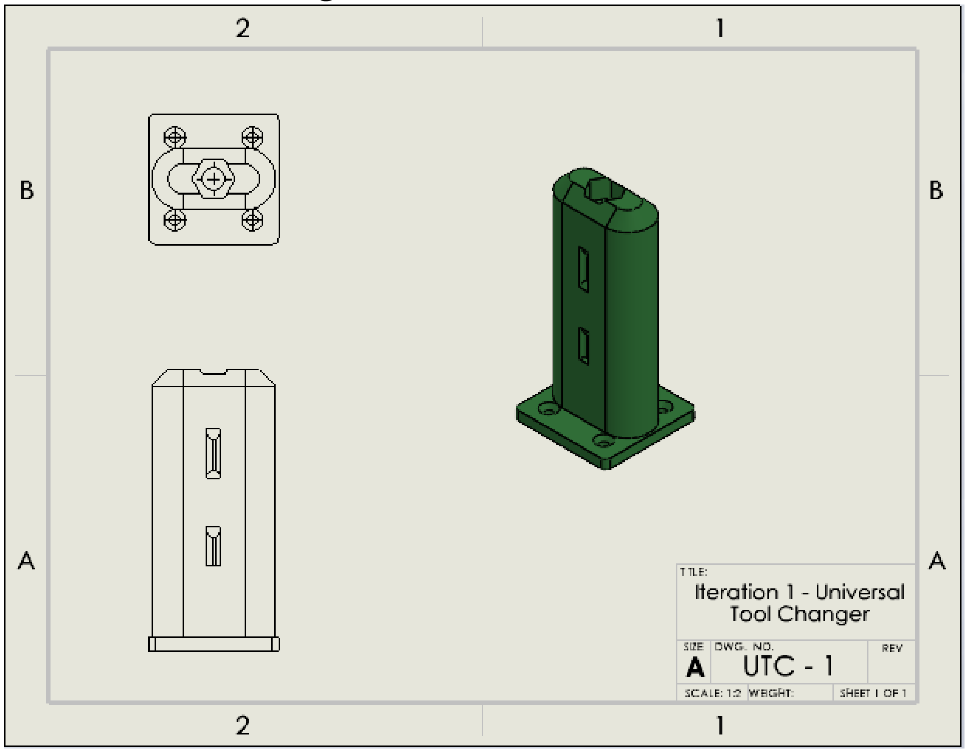


Figure 31: UTC Iteration 1

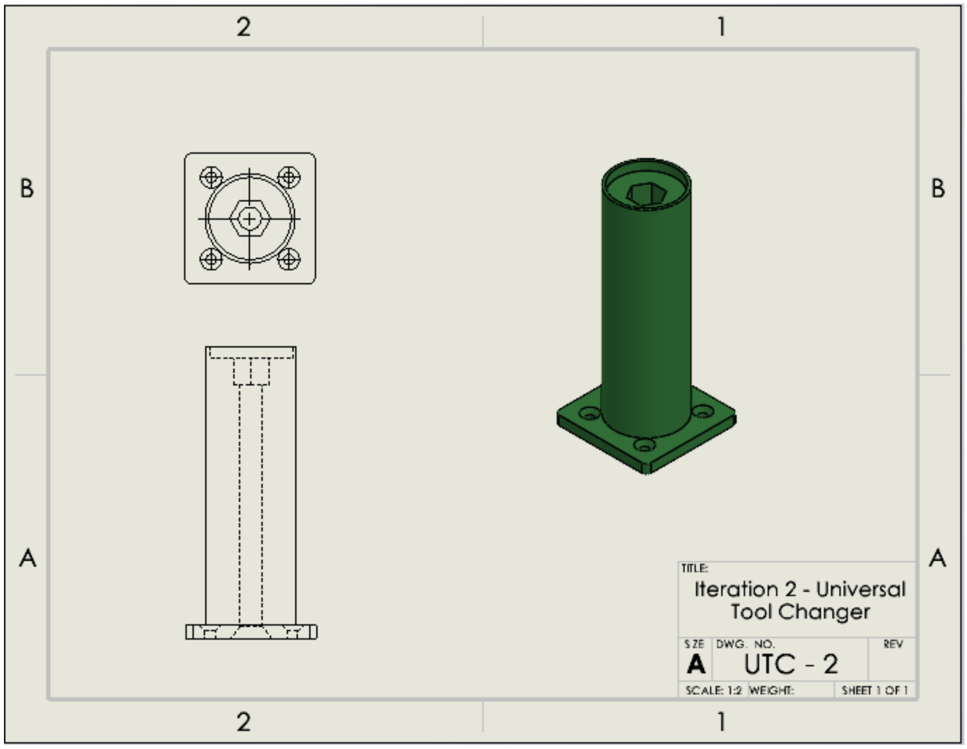


Figure 32: UTC Iteration 2

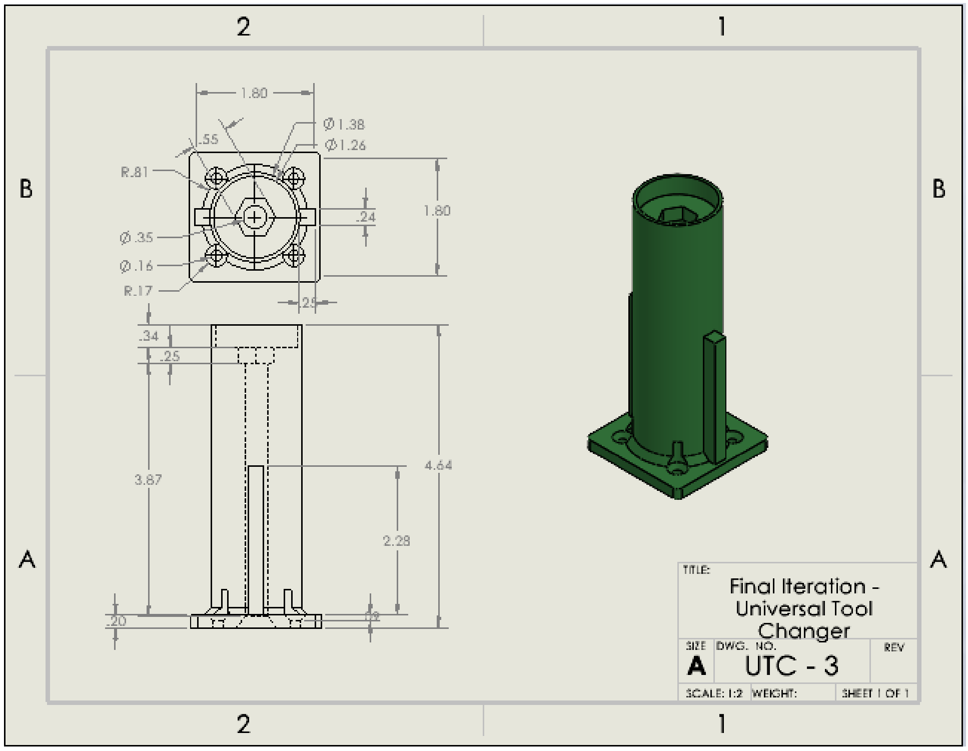


Figure 33: Final Iteration UTC

Press Part

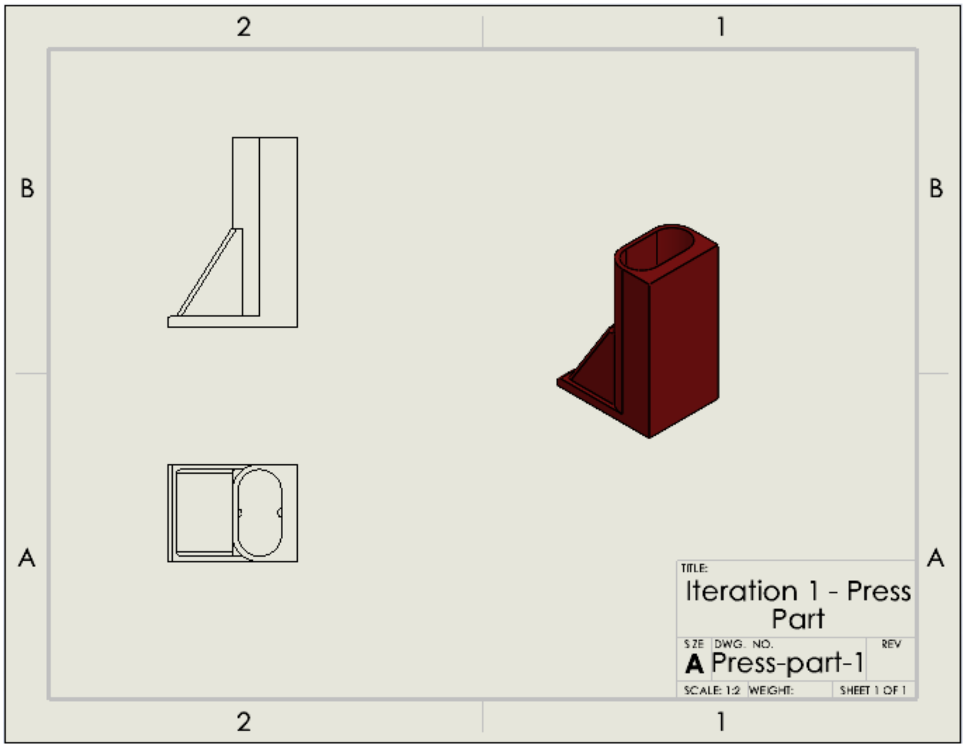


Figure 34: Press Part Iteration 1

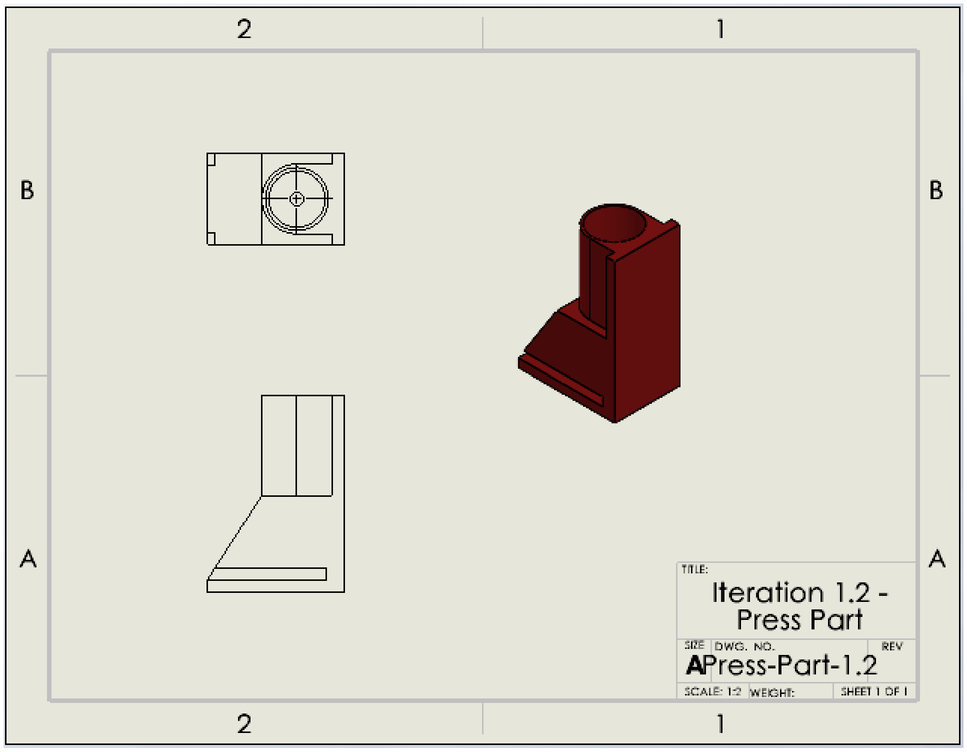


Figure 35: Press Part Iteration 1.2

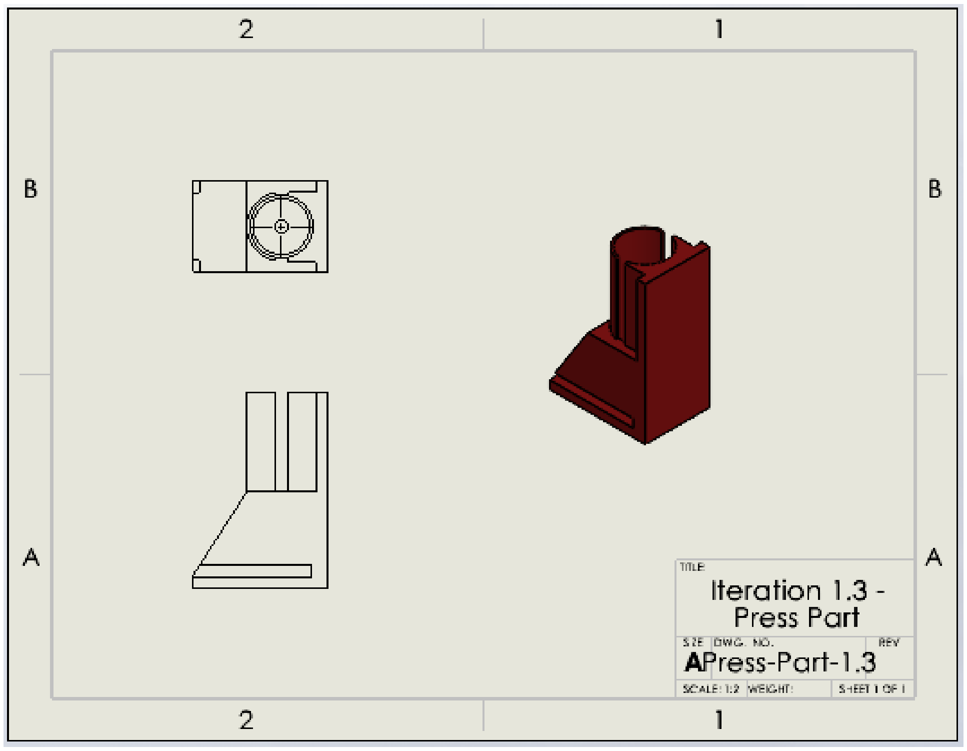


Figure 36: Press Part Iteration 1.3

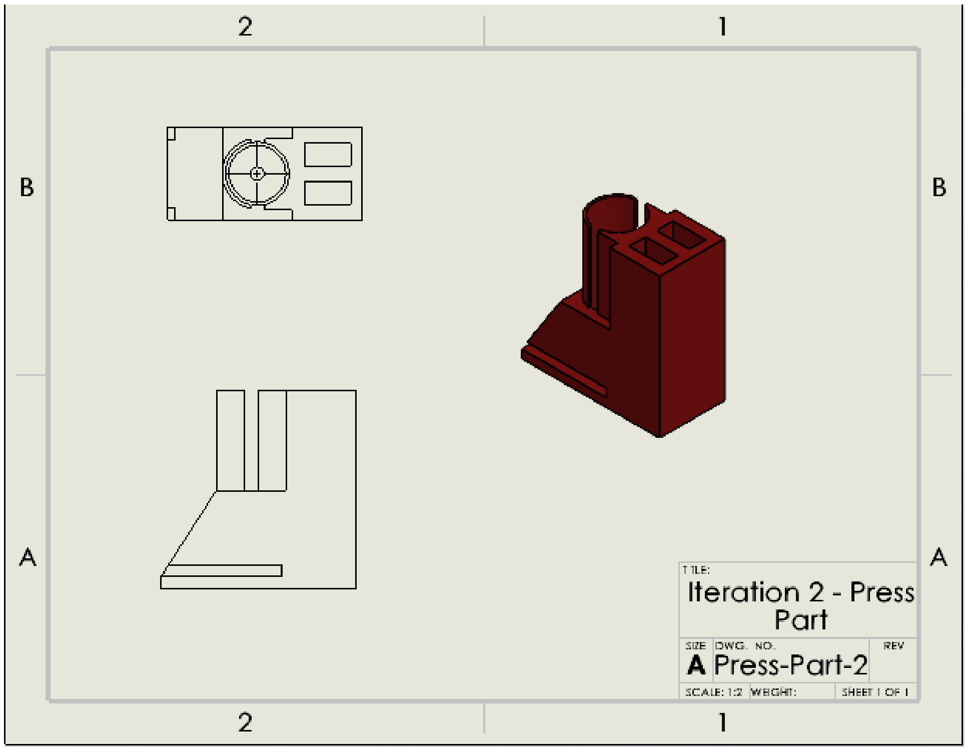


Figure 37:Press Part Iteration 2

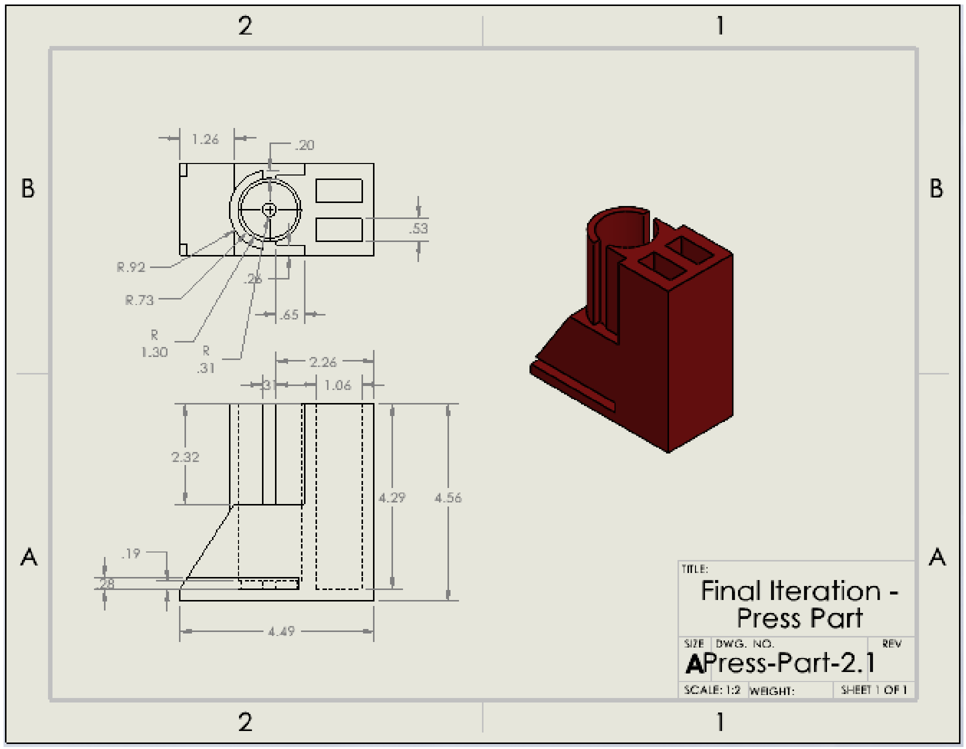


Figure 38: Final Iteration Press Part

Small Press Part

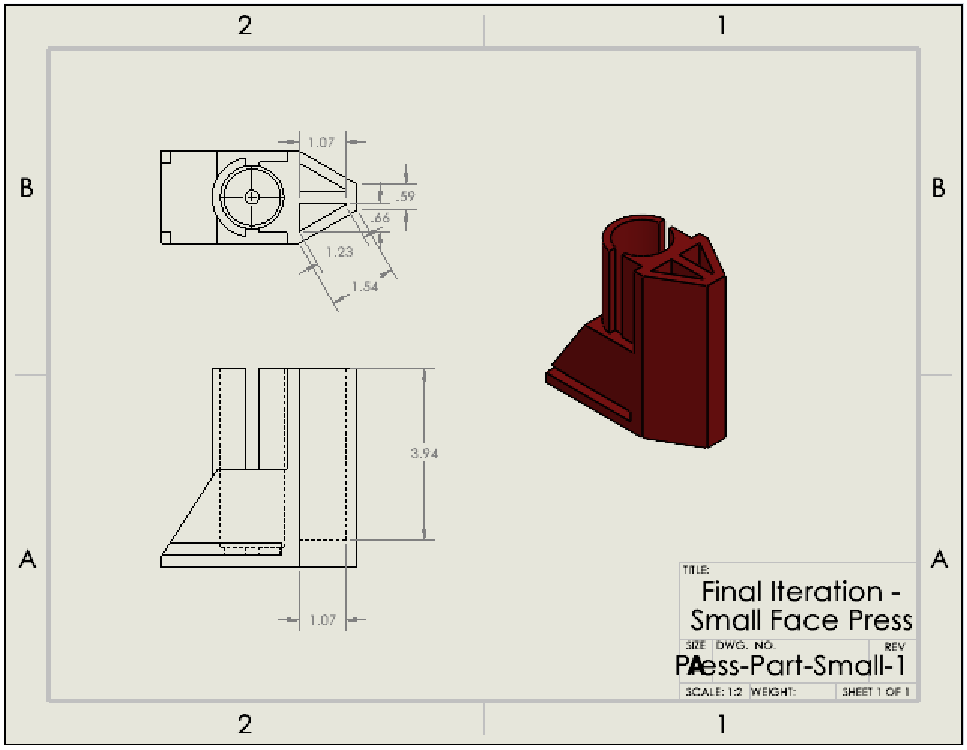


Figure 39: Final Iteration of Small Press Part

The Wedge

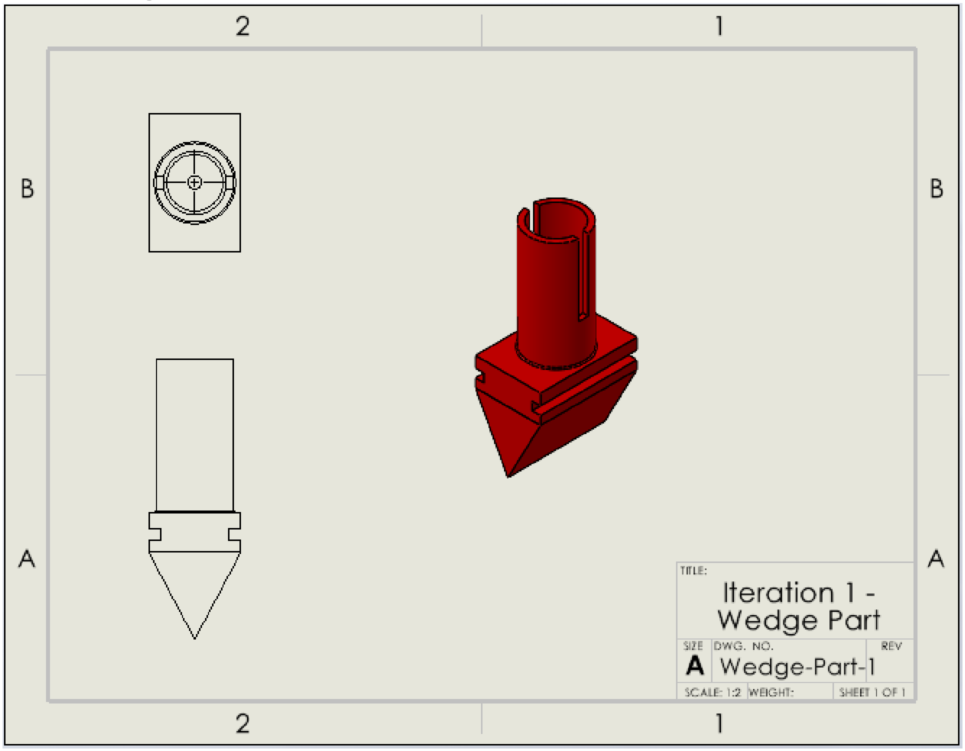


Figure 40: Wedge Iteration 1

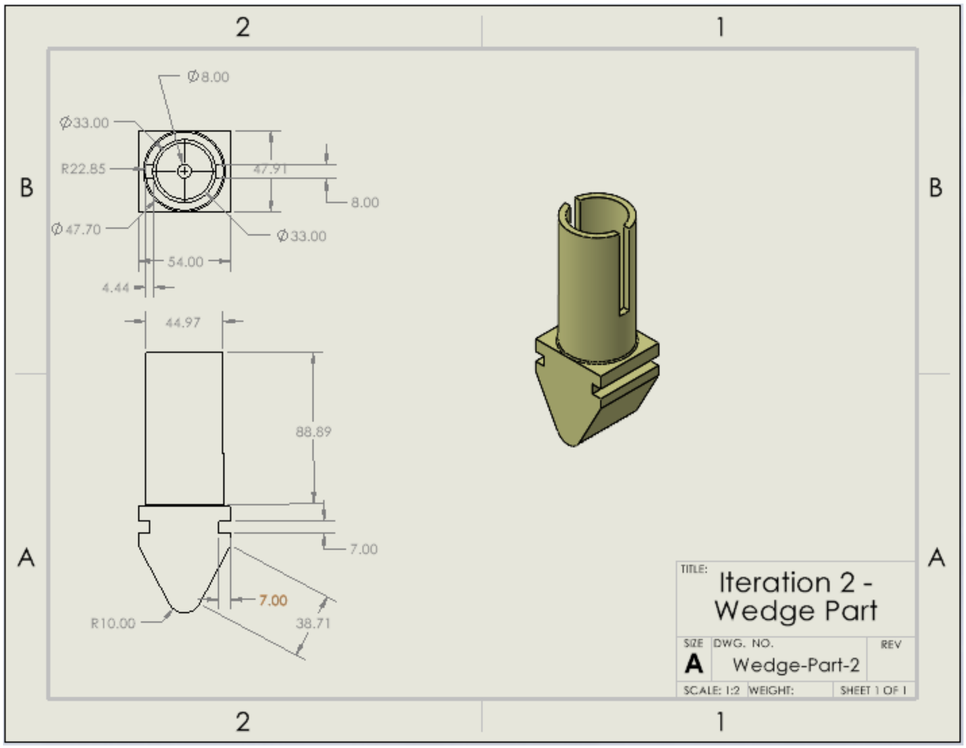


Figure 41: Final Iteration of the Wedge

Hole Punch

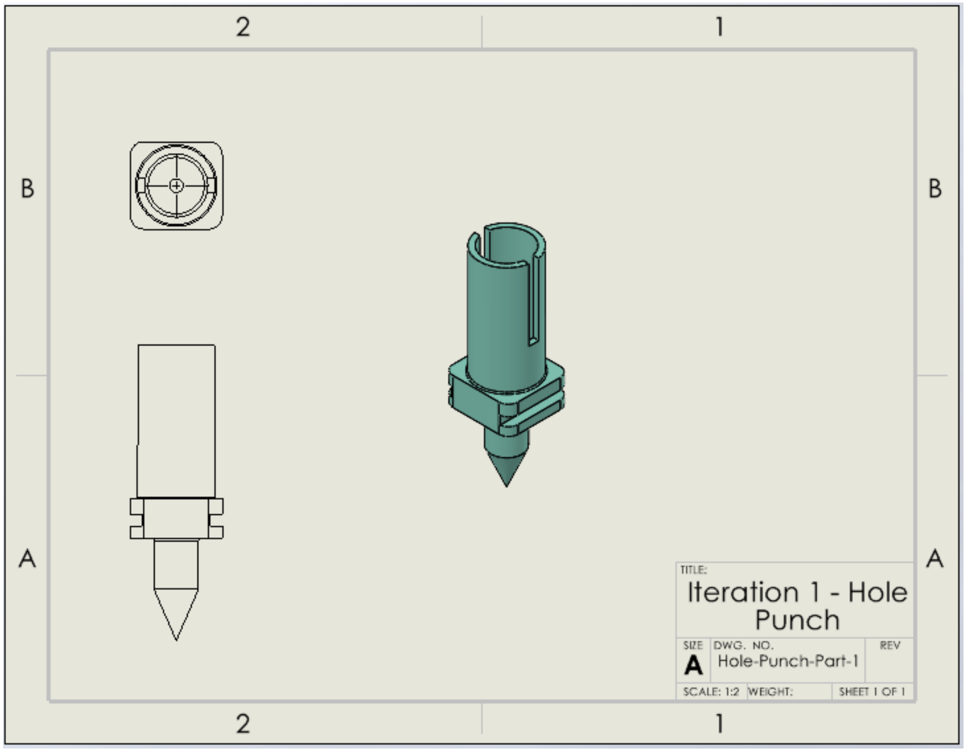


Figure 42: Hole Punch Iteration 1

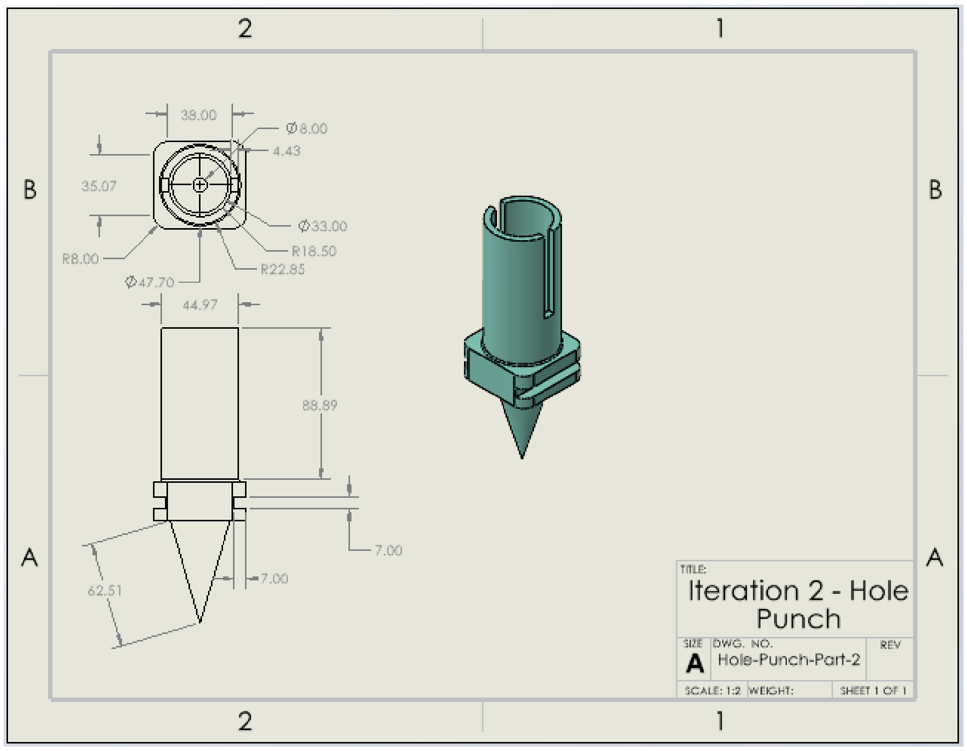


Figure 43: Final Iteration Hole Punch

Tool Racks

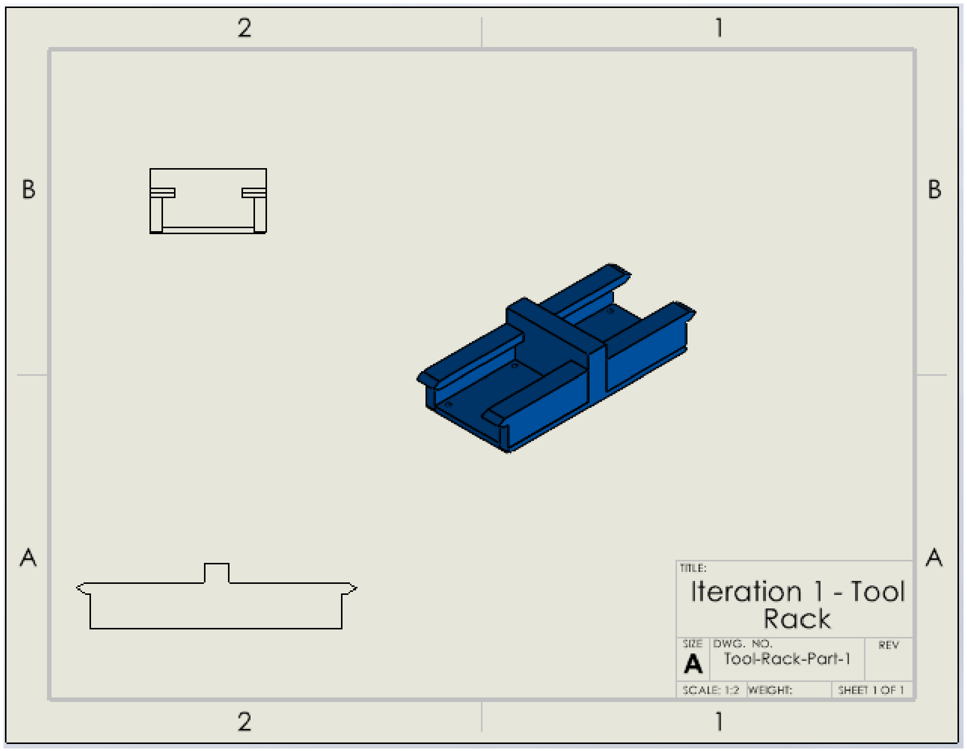


Figure 44: Iteration 1 Tool Rack

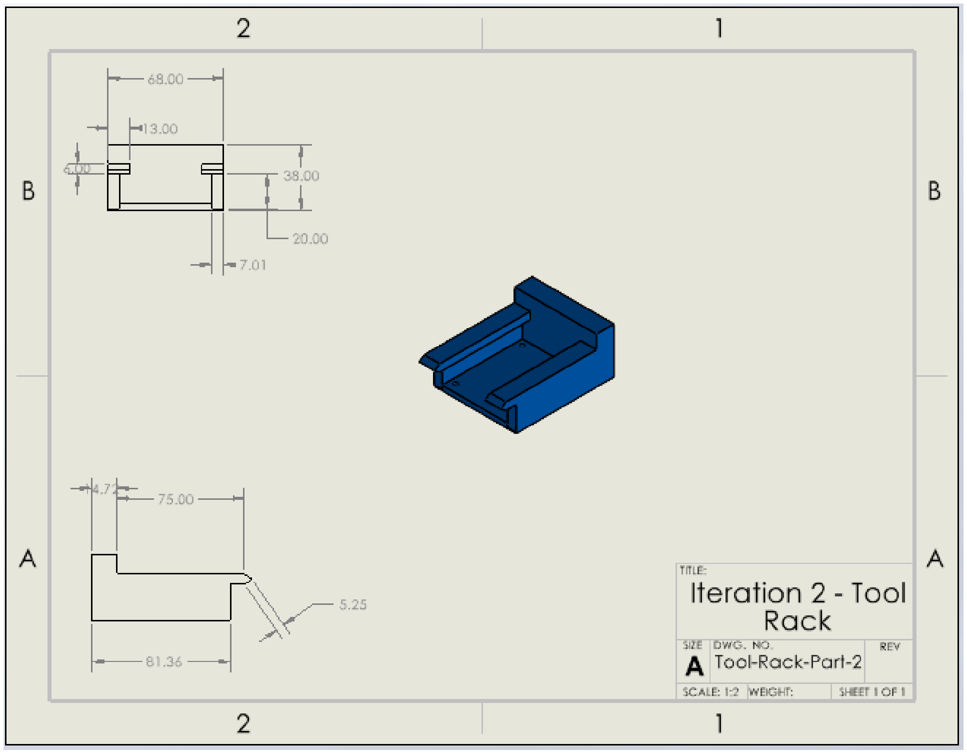


Figure 45: Final Iteration Tool Rack A

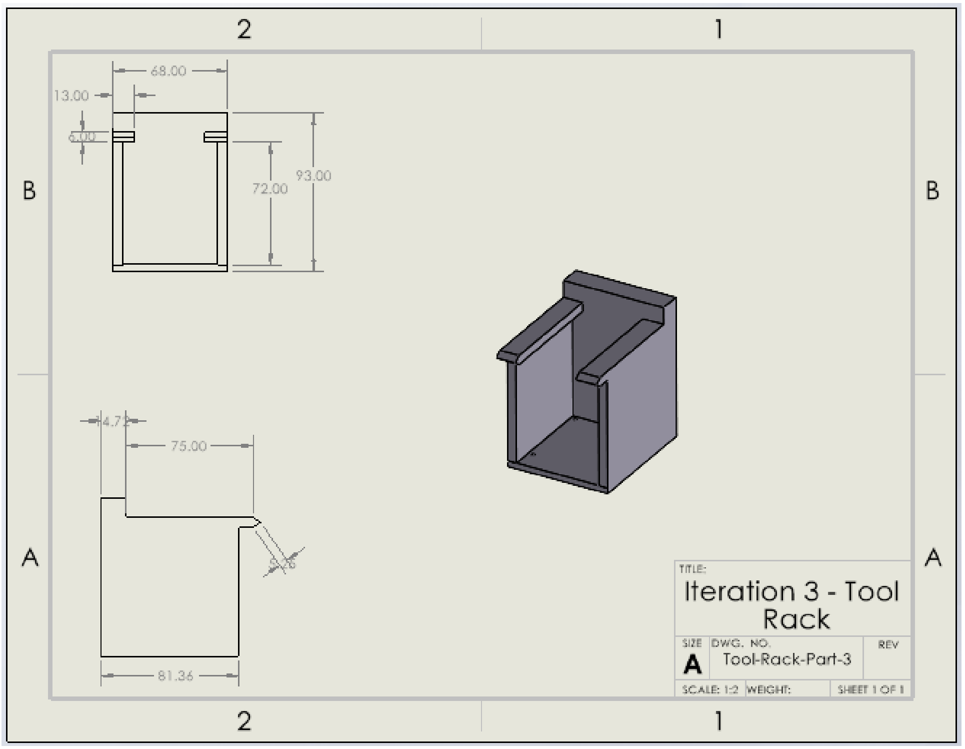


Figure 46: Final Iteration Tool Rack B

Conceptual Tools

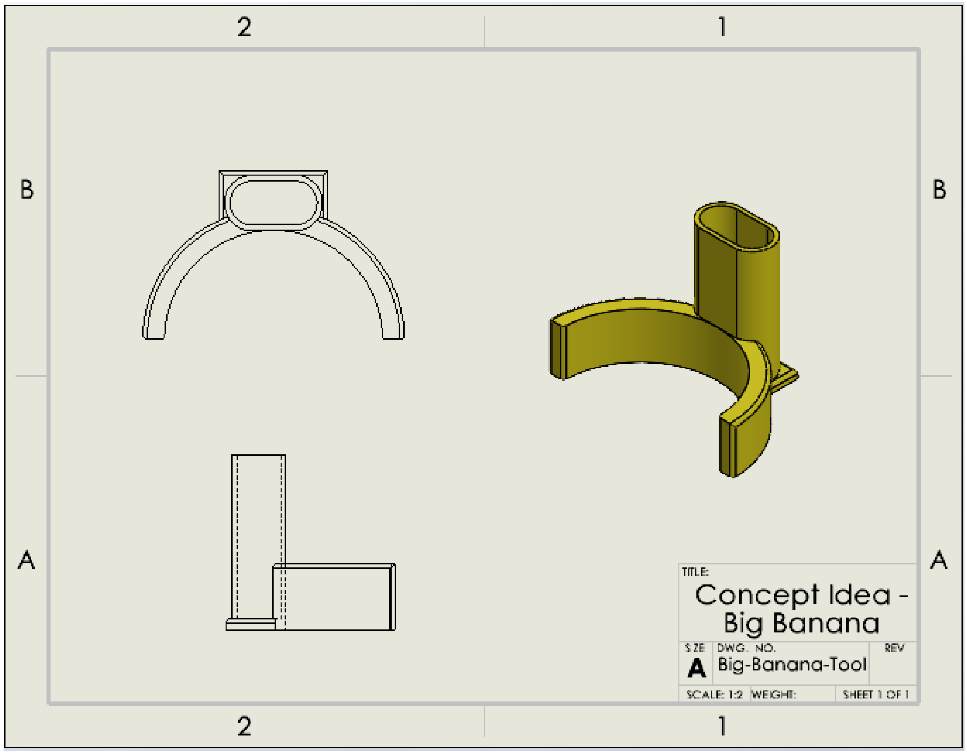


Figure 47: Banana Tool

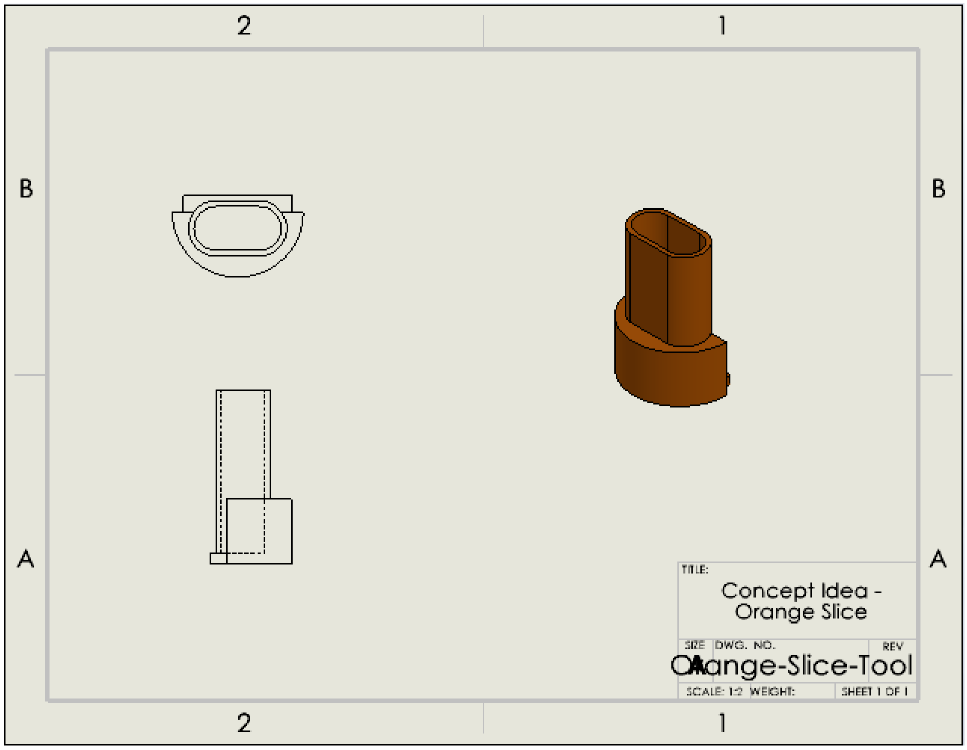


Figure 48: Orange Slice Tool

Appendix C: Digital Media

The software, code and documentation developed for this project, including the EOAT Control Code, Outlier Detection, RealSense Camera, and User Interface can be found at https://github.com/cvkittler/Metamorphic-Manufacturing

The CAD Files for the tools developed can be found at https://tinyurl.com/MMMQP-cad

Bibliography

[1] T. Ngo, “Additive manufacturing (3D printing): A review of materials, methods, applications and challenges - ScienceDirect.” https://www.sciencedirect.com/science/article/pii/S1359836817342944 (accessed Oct. 11, 2021).

[2] A. H, “The main problem in metal 3D printing - 3Dnatives.” https://www.3dnatives.com/en/problem-3d-printing-metal-070120215/ (accessed Apr. 21, 2022).

[3] I. Hutchings and P. Shipway, “9.4.1 Metal-cutting,” in *Tribology (Second Edition)*, I. Hutchings and P. Shipway, Eds. Butterworth-Heinemann, 2017, pp. 303–352. doi: 10.1016/B978-0-08-100910-9.00009-X.

[4] J. Dahmus and T. Gutowski, “An Environmental Analysis of Machining,” Jan. 2004, vol. 15. doi: 10.1115/IMECE2004-62600.

[5] A. Kumar, “What are Different Types of Forging Tools and their Uses? [Notes & PDF],” *THEMECHANICALENGINEERING.COM*, Sep. 23, 2021. https://themechanicalengineering.com/forging-tools/ (accessed Oct. 11, 2021).

[6] K. R. McClay, “The rheology of plasticine,” *Tectonophysics*, vol. 33, no. 1, pp. T7–T15, Jul. 1976, doi: 10.1016/0040-1951(76)90047-0.

[7] G. S. Daehn, “Metamorphic Manufacturing, or Robotic Blacksmithing – A Vision for a New Technology,” p. 3.

[8] “Team Honey Badger.” https://honeybadgerosu.wordpress.com/ (accessed Oct. 11, 2021).

[9] A. P. Green, “XLII. The use of plasticine models to simulate the plastic flow of metals.,” *Lond. Edinb. Dublin Philos. Mag. J. Sci.*, vol. 42, no. 327, pp. 365–373, Apr. 1951, doi: 10.1080/14786445108561061.

[10] R. N. Beaver, E. Campbell, A. G. Mavrotheris, N. J. Gere, and A. W. Muralt, “Metamorphic Manufacturing,” May 03, 2021.

[11] “Product specification - IRB 1600/1660.” ABB.

[12] “AMCI : Advanced Micro Controls Inc :: What is a PLC?,” *Advanced Micro Controls Inc*. https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/ (accessed Oct. 10, 2021).

[13] “EMD.” https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/RUBNER/emd.htm (accessed Apr. 25, 2022).