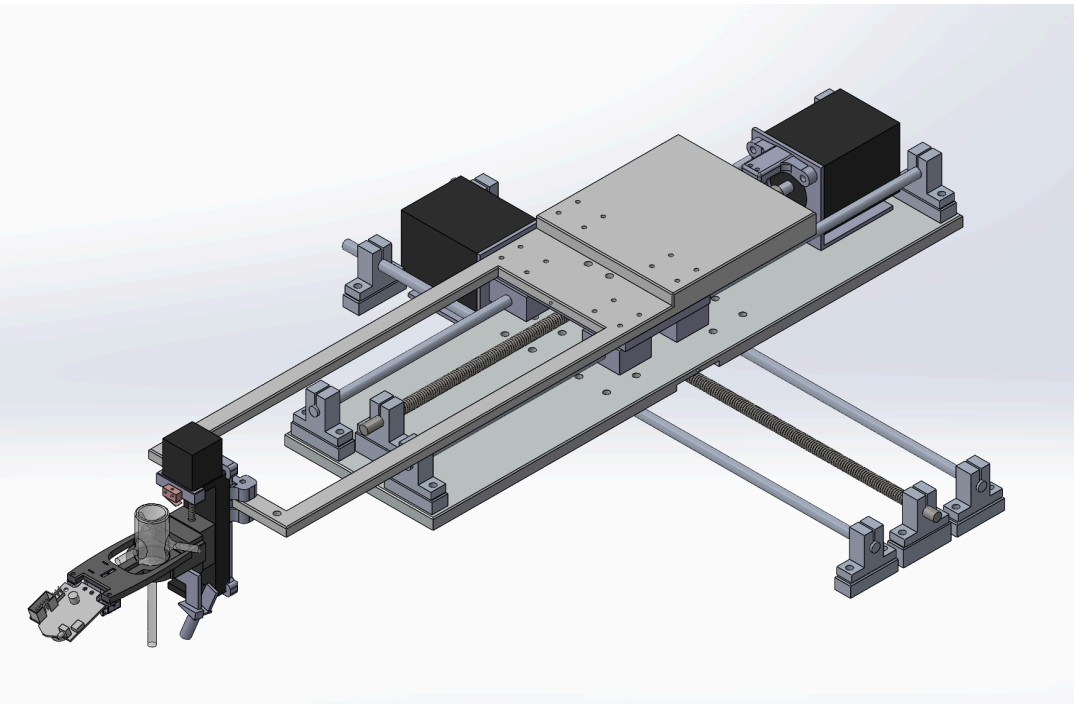
SAN JOSÉ STATE UNIVERSITY

CHARLES W. DAVIDSON
COLLEGE OF ENGINEERING

3-Axis Machine for Wound Image Processing and Plasma Medicine



Unurbayar Bayarsaikhan, Manuel Espindola, Eric Montoya,

Jorge Quintero, Nicholas Sandberg

Senior Design Project II, ME195B - 06

Dr. Zaidi

May 15, 2024

San Jose State University

Mechanical Engineering Department

Abstract

We developed a 3-axis machine for wound image processing and plasma medicine. The system features an end effector with a glass torch and a Pixy2 camera on a tilting Z-axis mechanism. The Pixy2 camera uses a Color Connected Components API to detect open wounds in real-time and convert boundaries from pixel to machine coordinates for precise movement. Controlled by an Arduino Mega 2560, the machine operates through TB6600 micro stepper drivers and Nema 23 and Nema 11 motors, with positional calibration via limit switches and safety features like an emergency stop and infrared temperature sensor. Results show reliable control by the Arduino, dependent on stable USB serial communication, with precise movements meeting plasma exposure time requirements. Finite element analysis and iterative design improvements ensure mechanical robustness. The control logic ensures safe operation, with safety features preventing harm from errors.

Acknowledgements

Special gratitudes to Dr. Syed Zaidi, IntelliScience Training Institute, and the Mechanical Engineering Department at San Jose State University for providing the funds, support, and materials to make this project possible. Special gratitudes are also extended to Akhil Agarwal and Aahan Patel for their contributions to the initial design of the project.

Table of Contents

Executive Summary **2**
Acknowledgements **2**
Table of Contents **3**
List of Figures **4**
List of Tables **5**
Chapter 1: Introduction **6**
I. Problem Definition 6
II. Objectives 6
III. Specifications 7
IV. Literature Review 8
V. Teamwork 10
VI. Gantt Chart/Timeline 11
Chapter 2: Theoretical Background **12**
Chapter 3: Design Concepts **13**
Chapter 4: Analysis and Design **17**
I. Analysis 17
II. Design Documentation 22
Chapter 5: Fabrication and Assembly **28**
I. Bill of Materials and Cost Analysis 28
II. Assembly 28
III. Fabrication 29
Chapter 6: Testing, Results, and Analysis **32**
Chapter 7: Social Impacts **34**
Chapter 8: Conclusions and Future Work **34**
References **36**
Appendices **38**
Appendix A: FEA Analysis: X limit Switch Mount 38
Appendix B: CAD Drawings 40
Appendix C: Component Datasheets 45
Appendix D: Motor Sizing and Specification Calculations 56
Appendix E: Main Arduino Script 57

List of Figures

Cold Atmospheric Plasma 7
Controlled Vs CAP healing process 10
PlasmaDerm Flex Device 10
Nema 23 Stepper Motor 13
Z-axis stepper motor with lead screw 13
Z-Axis Assembly Rev. 1 14
Z-Axis Assembly Rev. 2 14
Z-Axis Assembly Rev. 3 15
Z-Axis Assembly Rev. 4 15
Arducam Mega 5MP . 2024. Arducam 16
Pixy 2 Camera (LEft), Shown on Z axis assembly (Right) 17
Visual representation of the workspace as visible by the Pixy2 Camera 19
Section of the Arduino script responsible for image processing 19
Raw wound image (left) with plotted toolpath visual representation overlaid (right) 20
Emergency stop button as seen in final design 21
Z axis assembly with infrared temperature sensor to monitor plasma temperature 21
Limit Switch Mount Total Deformation Test 22
Limit Switch Mount Von Mises Stress Test 22
X-axis limit switch mount 23
Y-axis limit switch mount 24
Z-axis limit switch mount 24
Dashboard Assembly 25
Final assembly with components labeled (1/2) 26
Final assembly with components labeled (2/2) 26
Z axis assembly with all components labeled 26
Machine control logic diagram for Arduino Script 27
Wiring Diagram for electronic components on final prototype 28
Ultimaker Cura sliced models showing filament weight and print time 30
Expansion modifiers to ensure dimensional accuracy 31
Sliced models of mounts carrying various components 31
3 axis gantry robot with live plasma in operation 33

List of Tables

Team Roles **11**
Project Gantt Chart **12**
Project BOM **29**
3-Axis Machine Commands **32**

Chapter 1: Introduction

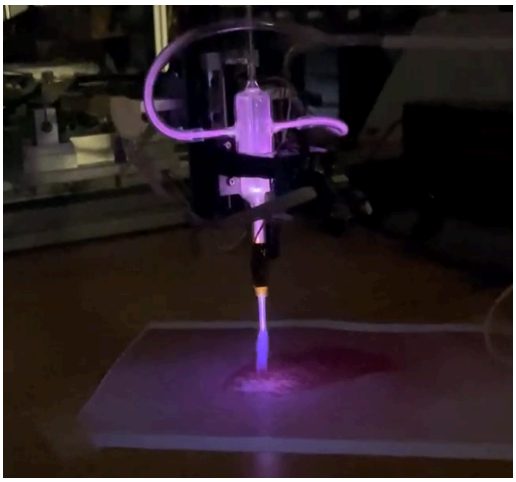


Figure 1: Cold Atmospheric Plasma

I. Problem Definition

Insufficient and improper patient wound sanitization is one of the leading causes of improper healing. The lack of proper sanitization allows for bacterial and microorganisms to survive on a wound leading to an increased chance of infection. This also leads to a prolonged recovery time for the patient's wound to fully heal. Not only are antiseptics painful, but an excess amount of product can actually damage the surrounding area as well. These inconsistencies of product application are typically caused by human error. Additionally post covid conditions have caused logistical issues with the availability of common antiseptics. The 3 Axis Machine for Wound Image Processing and Plasma Medicine guarantees a quality and proper wound sanitization to the patient utilizing Cold Atmospheric Plasma (CAP) . Incorporating image process technology to the 3-axis gantry robot will ensure that the entire wound receives even coverage. The 3-axis gantry mechanism provides the patient with a consistent, precise, and accurate application of plasma to the wound to maximize the results. Once a wound is recognized by the image processing, a tool path is automatically created to guarantee an even amount of plasma application to the entire wound. This service provides the patient with a painless experience while properly sanitizing their wounds and delaying their recovery times.

II. Project Objectives

The main objective of this project is to create a gantry robot that allows for movement in 3 directions and processes images automatically to detect a wound. Along with creating a pathway for the machine to apply consistent Cold Plasma treatment over the detected wound. The machine must also perform toolpaths that can be repeated within a reasonable margin of

accuracy over multiple consecutive operations to maintain the overall integrity of the treatment procedure. The design of the machine must be automated to the fullest extent available with minimal intervention from the machine operator. The defined problem was tackled differently with this project than previous ones due to the image processing components and the manually controlled 3rd axis. The initial design of the machine by the previous group consisted of 2 axes control and camera component which were operated by the machine operator. The machine was improved into 3 axes control with a fully automatic camera module. The Z axis is intended to be manual which will allow for greater control over the plasma intensity when applied to the wound surface. The image processing component allows for the mechanism to administer plasma treatment with minimal operator intervention and more accuracy. The development and implementation of this project will require the use of various different principles and technologies to be a success. The use of cold atmospheric plasma will necessitate a working knowledge of plasma as it pertains to medical treatment. The physical movement aspects of the project will require an application of mechatronic principles. The 3 axes movements will be performed with a stepper motor, lead screw, and carriage assembly for each axis. The stepper motors will be controlled by an Arduino microcontroller and powered by an external power supply unit to ensure proper electrical voltage and current for the signal and high voltage circuits. All circuits will be embedded and routed via a PCB designed specifically for this application. The wound detection will be handled by image processing which requires the use of a camera interfaced with a microcontroller, Arduino in our case, and then the data from the camera must be interpreted to gain useful information about the wound in order to perform wound treatment.

III. Specifications

There are various design specifications which our machine must meet in order to satisfy the objectives of the project. First, the mechanism must have mobility in all 3 axes with automated movement in the X and Y directions. As well as manual control in the Z axis. The machine should have a minimal workspace of 218mm x 167mm x 85mm movement. This will allow our mechanism to treat a wide variety of wounds and allow for sufficient control of the plasma intensity by the machine operator. The X & Y axis will operate at 24 mm/s and provide an even coverage of plasma in about 10 seconds. Second, the machine must have precise and accurate Z-axis movement of .5 mm per rotary encoder position. The machine must retain accuracy with multiple consecutive operations. Third, the mechanism must be semi automated with minimal intervention from the operator for streamlined operation. This will be accomplished by an implementation of a homing and self calibrating operation controlled by the Arduino with assist of limit switches. Lastly, the mechanism must be able to process images and successfully locate a wound with repeatable accuracy and consistency with the camera module.

IV. Literature Review

When it comes to wound recovery, sterilization is one of the key components besides time. One of the most commonly used methods of wound sterilization is the use of antiseptics. The utilization of antiseptics is very common due to its availability and easy-of-use products that do not require any professional training. Most commonly used antiseptics include iodine solution, rubbing alcohol, and hydroperoxide. These items are extremely popular due to it being over-the-counter products and its wide usage in medical facilities. Antiseptics are known to kill microorganisms and bacteria, but if used incorrectly these products can cause damage. When sanitizing a wound it is extremely important to be cautious with the quantity of product being applied to the damaged area, in order to not over saturate the wound. According to *Effects of the most wound antiseptics on human skin fibroblast,* antiseptics are known to impact the recovery of open wounds by killing necessary fibroblast required to heal. While antiseptics do in fact kill microorganisms and bacteria, it has been found that these products kill fibroblasts and skin cells, when an excess amount of product is applied.

Cold Atmospheric Plasma (CAP) is a new up-and-coming technology being utilized for wound sterilization. Initially, plasma was only produced in extreme levels of energy and density through vacuum, making it impossible to be used on humans. Fortunately, technological breakthroughs have allowed plasma to be produced at atmospheric pressure. Plasma is a high concentration of energy shaped like a lightning bolt. It is recognized as the fourth state of matter following solids, liquids, and gasses. CAP is a mixture of reactive gasses such as helium, argon, nitrogen, and oxygen. The gas mixtures have an important role in the efficiency and quality of the plasma. As seen in **figure 2**, a two week study was conducted on rats under *Bioeng. Biotechnology* study. The most effective gas mixture found was Helium and Argon as the subject dramatically healed during the trial compared to its control counterpart. According to *Cold Atmospheric Pressure Plasma in Wound Healing and Cancer Treatment,* unlike antiseptics, CAP has been proven to kill and reduce microorganisms/bacteria around a wound while promoting tissue repair. The study has found when Plasma makes contact with the skin, it creates a hierarchy group of reactive oxygen and nitrogen species called RONS. These reactive oxygen and nitrogen groups cause an increase in skin tissue microcirculation and monocyte stimulation. One of the most significant disadvantages of using Cold Atmospheric Plasma is the meticulous requirements to have peak performance. When applied to the skin, the temperature must not exceed 40 degrees celsius or else the plasma may cause permanent damage to the wound. Lastly, the key factor to ensuring an optimal recovery is providing a consistent and even CAP to the wound. A consistent coverage ensures that all areas receive an even amount of plasma, by doing so it would allow the wound to heal in unison.

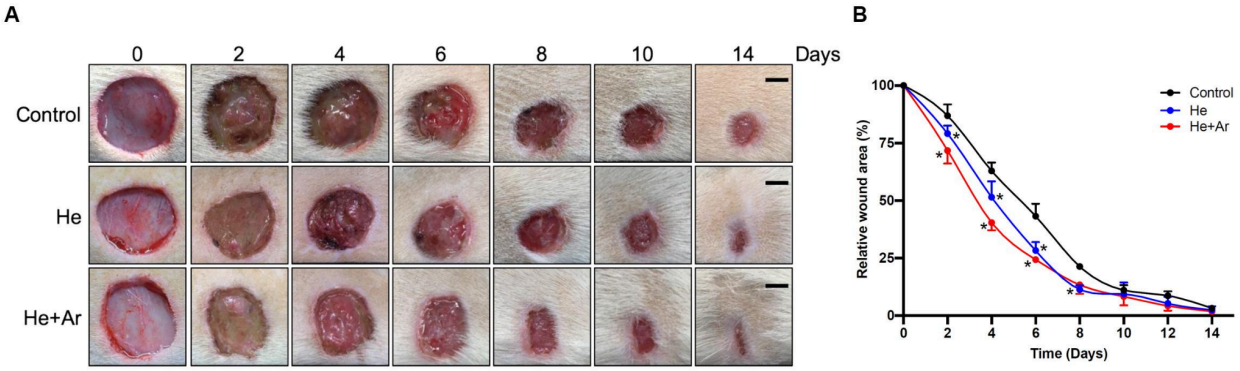


Figure 2: Controlled Vs CAP healing process

Figure 3: PlasmaDerm Flex Device

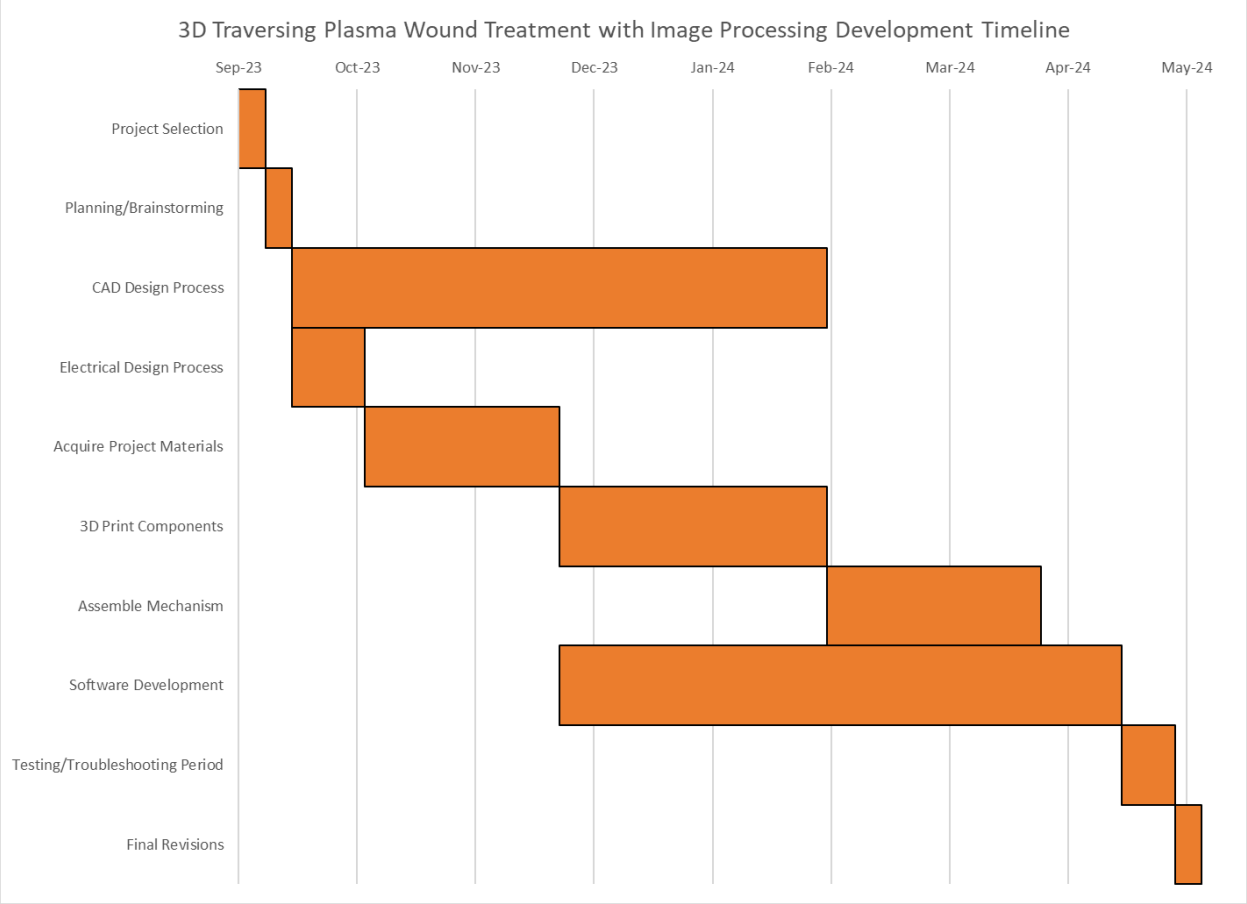
Being that Cold Atmospheric Plasma (CAP) is a cutting edge technology, there isn't an established market or high demand for medical equipment capable of applying CAP. The currently available medical devices suited to administer CAP are handheld mechanisms from a company called *PlasmaDerm Flex*. While a handheld device allows us to sanitize a wound, there is a lot of room for operator error which may lead to reinfection. In order to guarantee patients receive proper care, an automated sterilization mechanism is necessary to ensure that CAP is applied evenly with each application. A cartesian robot otherwise known as Gantry robot is a mechanical device that uses linear axes movements in the 2D or 3D space. Generally the linear motion is caused by a stepper motor spinning a leadscrew with a plate attached to it that moves back and forth. Gantry robots are known for their high level of accuracy and precision and can have tolerances as low as micrometers. Gantry robots also have a relatively low cost to build and have a huge online support system that makes them easy to program. Some popular examples of these types of robots are CNC milling machines, 3D printers, Soda Machine systems etc. Typically Gantry robots are “blind” and have a pre-set code that commands what the system will do. One huge advantage of these mechanisms is the ability to add any kind of vision device such as cameras and sensors in order to help automate. In this project we utilized a gantry robot

for its accuracy and precision in order to ensure each wound gets properly treated. We also added image processing to help locate the wound. In order for the Gantry robot to be ISO certified we are following ISO standard 13850:2015 and ISO 14971:2019. These standards are reviewed with great detail in Chapter 4. In order to avoid any miscommunication with engineering drawings ASME Y14.5:GD&T Dimensioning and Tolerancing Standards.

V. Teamwork

Name	**Role**
Unurbayar Bayarsaikhan	Group Leader, Technical Lead: Image processing
Jorge Quintero	Technical Lead: Electrical Design
Eric Montoya	Technical Lead: Assembly, Manufacturing, CAD Modeling
Nicholas Sandberg	Communications Director, Technical Lead: Software, Writing
Manuel Espindola	Technical Lead: CAD Modeling, Programming, Assembly

VI. Gantt Chart/Timeline



Chapter 2: Design Concepts

Figure 4: Nema 23 Stepper Motor

The X-axis lead screw on our 3-axis gantry robot experiences the most downward forces. The total amount of weight that sits on the X-axis lead screw is about 4 Kg. We utilized an online lead screw calculator and found that $.722N/m$ is required to spin the lead screw. Through trial and error we found a stepper motor velocity of 480 RPM to be ideal. Using the stepper motor torque graphs, we determined that the Nema 23 motor is best for our application. Our team decided to utilize the Nema 23 motor on both our X and Y axis. The motor specs and calculations that were used to determine our motor sizing can be found in Appendix D.

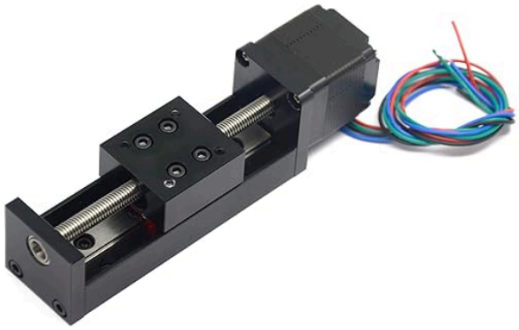


Figure 5: Z-axis stepper motor with lead screw

The Z-axis stepper motor was chosen off the shelf from amazon. Our design revolved around this stepper motor as it has the linear motion already incorporated which allows for the Z-axis controllability. This motor is also rated at 1 kg capacity for vertical movement which is more than what is required for our application.

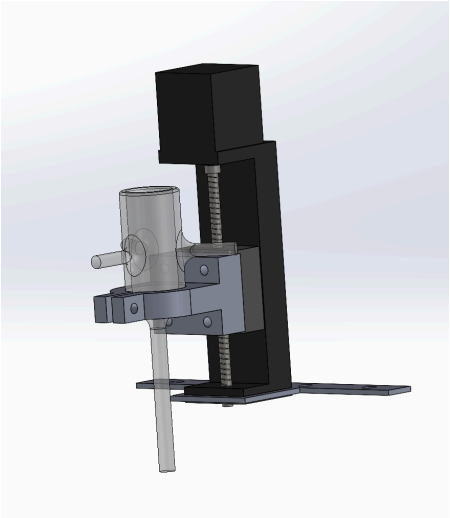


Figure 6: Z-Axis Assembly Rev. 1

Our initial design concept of the Z-Axis/Vertical Assembly was solely assigned with the task of holding our glass plasma torch without slippage, and mounting the linear actuator on top of our mechanism.

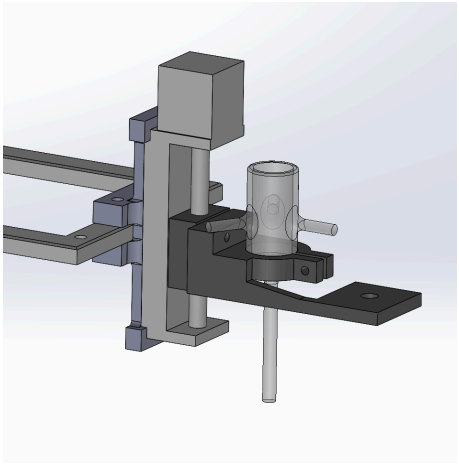


Figure 7: Z-Axis Assembly Rev. 2

We quickly made changes from the first iteration upon the realization that we needed our Z-Axis Assembly to travel below the mechanism's Y-Axis platform. We implemented a mount that will hold on to the Z-Axis assembly entirely. We additionally designed a slot where the mechanism's Y-Axis platform will fit and be bolted together. We also incorporated the mounting platform with the former Arducam 5MP.

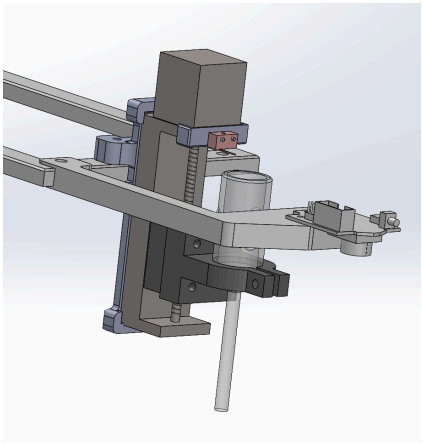


Figure 8: Z-Axis Assembly Rev. 3

Our third revision came to fruition when we made the decision to switch camera components from the Arducam to the PixyCam2 and incorporated the use of limit switches for mechanism homing. This revision we also experimented with camera position and managed to build a camera mount with a fixed position on the mechanism so the camera's field of view FOV would be at its maximum and we would not encounter issues with torch position relative to the mechanism and wound.

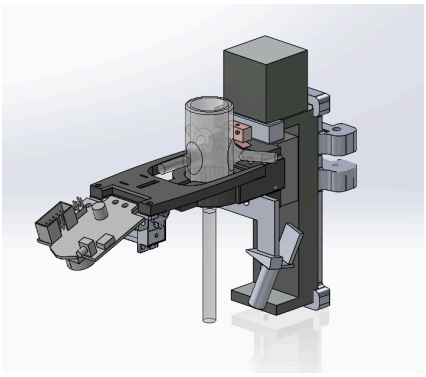


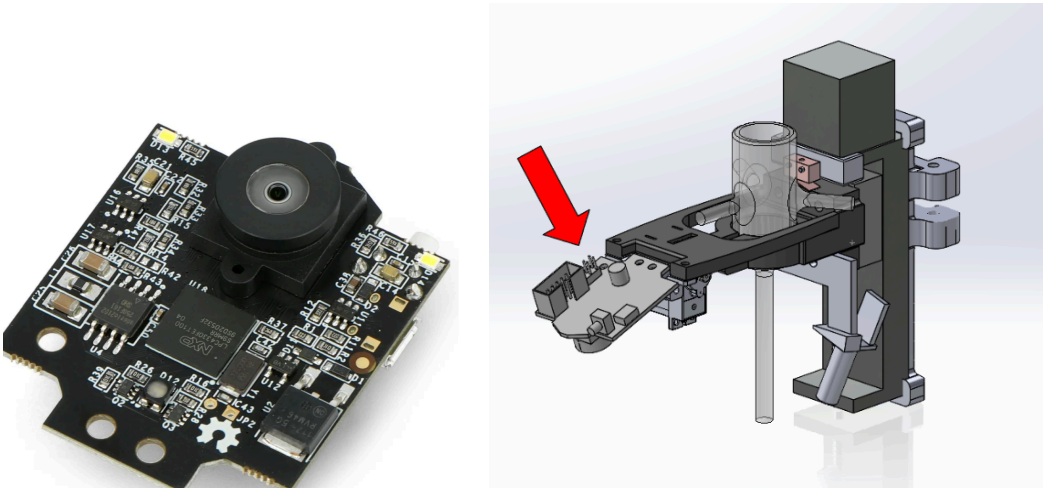
Figure 9: Z-Axis Assembly Rev. 4

Lastly we arrive at the final revision of our Z-Axis assembly. For our final iteration we continued to utilize Revision 2's limit switch mount, and linear actuator mount. The camera mount once again was integrated into the torch mount that was held onto a stage driven by the linear actuator subassembly. The main difference between this camera mount was that it also held a servo motor which was used to adjust the camera tilt onto the workspace to allow better FOV should the wound be directly below the torch preventing us from seeing where the plasma was being applied. There was also an incorporation of an Infrared Sensor in the bottom of the linear actuator stage that was set to a fixed angle whose line of sight would coincide with the active site of the plasma and wound. This component and its mount allowed us to monitor skin temperature complying with ISO 14971:2019.

Figure 10: Arducam

Arducam Mega 5MP , 2024. Arducam

The initial design of the project utilized the use of ArduCam Mega camera with 5MP and Matlab for image processing. The manufacturer of the ArduCam Mega camera, ArduCam, developed the ArduCam Mega application to configure the camera settings and take images with it via Arduino Mega 2560. The image would have to be taken by the user on the ArduCam Mega application and saved onto the computer to be later called in Matlab Script. The Matlab Script we developed would process the image by taking the RGB values of each pixel in the image, and matching it with the desired value, which in our project would be a range of red colors that are similar to the color properties of the wound. The unmatched pixels were converted to black pixels, in doing so the matlab script will easily draw out a boundary box around the wound and relay its properties to the microcontroller. These properties consist of X, Y coordinates of the top left corner, width and height of the boundary box. In the microcontroller, the properties of the boundary box are scaled to the physical domain by converting the pixel distances to mm. The microcontroller is also able to locate the wound with respect to gantry stages by calculating the distance between the center of the camera and the top left corner of the boundary box.



Pixy 2 Camera, 2024, Amazon

Figure 11: Pixy 2 Camera (Left), Shown on Z axis assembly (Right)

The final design of the project made use of Pixy2 camera for the image processing. The Pixy2 camera has a refresh rate of 60 frames per second and capability to process images live. The Pixy2 utilizes the built-in Color Connected Components algorithm, also referred to as CCC algorithm. This algorithm applies a color base filter method to detect objects, for this project it would be a wound. Unlike basic RGB color models like the one used in the Matlab Script, the algorithm focuses on analyzing the color and saturation of each RGB pixel from the image. The developer of the algorithm states "the algorithm ensures color of the object remains consistent with changes in lighting and exposure". Since the Pixy2 camera has a built-in image processing algorithm on the module, the need to transfer images from the camera module to the computer was eliminated. The Pixy2 utilizes the PixyMon V2 application to configure the camera and the saturation of the sensors.

Chapter 3: Theoretical Background

The use of Cold Atmospheric Plasma (CAP) for our application requires an understanding of its fundamental physical properties. Plasma is the 4th state of matter following solid, liquid, and gas. According to an article from *Thin Solid Films,* plasma can be characterized as a partially ionized gas containing neutral particles as well as an equivalent number of negative electrons and positive ions (Bárdos 6705). Plasma can be classified as hot or cold plasma, dependent on the temperature. Since plasma is being applied to living tissue, the plasma gas must be of low temperature in order to avoid tissue damage. In this application, the ideal plasma qualities are low temperature and non equilibrium with a gas temperature of less than 50 degrees Celsius. (Bárdos 6710). According to an article in the *Journal of Personalized Medicine,* the working method of the cold atmospheric plasma on the skin occurs via the deposition of a film of reactive oxygen and nitrogen ions that promote increased skin tissue microcirculation, increased

monocyte stimulation, increased cell migration, and stimulation of the keratinocytes and fibroblasts primarily involved in wound healing (Bolgeo 736). With this in mind, the key design factors for the plasma component of our mechanism includes gas composition to be released through the plasma torch, the distance between the anode and cathode supplying electrical charge to the gas mixture, and the amount of electrical charge supplied to the gas mixture to ionize it.

The next aspect of the mechanism that requires engineering theory is the image processing. One of the key design objectives for this project is defined as using a camera to successfully capture an image of a wound and then use a microcontroller to interpret the image data and use this information to automatically size and locate the wound to deliver the plasma treatment as desired. To understand image processing, the fundamental properties of images must be considered. According to an article published by *Simplilearn*, an image is defined by its height and width based on the number of pixels, or the resolution of the image. Each of these pixels contains information including the color, shade, and opacity of the image, which can be interpreted by a computer program to get useful information from the image. There are various types of image processing, the type used for the scope of this project is recognition. Recognition uses image data to distinguish and detect objects in the image. A complete image processing apparatus includes a computer, microcontroller, camera, and the software needed for it all to run. The design of the final product had to incorporate all of these components and then they were to be set up to detect wounds automatically and direct the stepper motors to move the plasma torch apparatus in a tool pathway to administer treatment to the wound. Image processing as applied in the final mechanism is described in greater detail in **Chapter 4: Analysis and Design**.

Chapter 4: Analysis and Design

I. Analysis

The image process of the wound was executed by the Pixy2 camera module with its Color Connected Components algorithm. By using the PixyMon V2 application, the camera was trained to search for wounds by placing images of wounds in front of the camera. When the image is under the camera, the CCC algorithm marks and stores the color signature of the wound. The stored color signatures are set as a prime filter to search for wounds. The Pixy2 camera creates a 315 x 207 pixel view as shown in **Figure 12** , which is equivalent to 216 x 167mm visible workspace. Within this view, the Pixy2 camera is able to detect wounds, draw a boundary box and relay the properties of the boundary box to Arduino Mega 2560 instantaneously. The relayed properties consist of midpoint (X and Y coordinates), width and height of the boundary box. In the script, these properties are stored in the array and can be accessed by the command pixy.ccc.block as shown in **Figure 13** .

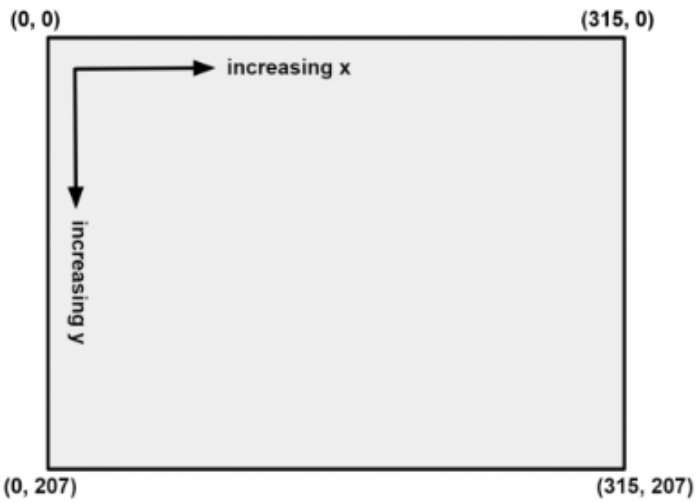


Figure 12: Visual representation of the workspace as visible by the Pixy2 Camera

```
// Grab the latest blocks from Pixy2. Color Connected Components
pixy.ccc.getBlocks();
if (pixy.ccc.numBlocks) {
  for (int i = 0; i < pixy.ccc.numBlocks; i++) {
    // Convert Pixy2 coordinates to machine coordinates
    if (pixy.ccc.blocks[i].m_signature == signatureNumber) {
      float cameraX = (pixy.ccc.blocks[i].m_x - 157.5) / 157.5 * 150;
      float cameraY = (pixy.ccc.blocks[i].m_y - 103.5) / 103.5 * 85 + 50;
      float boundaryWidthMM = pixy.ccc.blocks[i].m_width * widthScaleFactor;
      float boundaryHeightMM = pixy.ccc.blocks[i].m_height * heightScaleFactor;

      Serial.print("Wound Detected - X: ");
      Serial.print(cameraX);
      Serial.print(" mm, Y: ");
      Serial.print(cameraY);
      Serial.print(" mm, Boundary Dimensions - w: ");
      Serial.print(boundaryWidthMM);
      Serial.print(" mm, h: ");
      Serial.print(boundaryHeightMM);
      Serial.println(" mm");

    }
  }
}
```

Figure 13: Section of the Arduino script responsible for image processing

Once the boundary box has been created around the wound, the Arduino script will generate a toolpath for the motion of the plasma torch via the x and y axis stepper motors to follow. See below for a visual representation of the toolpath.

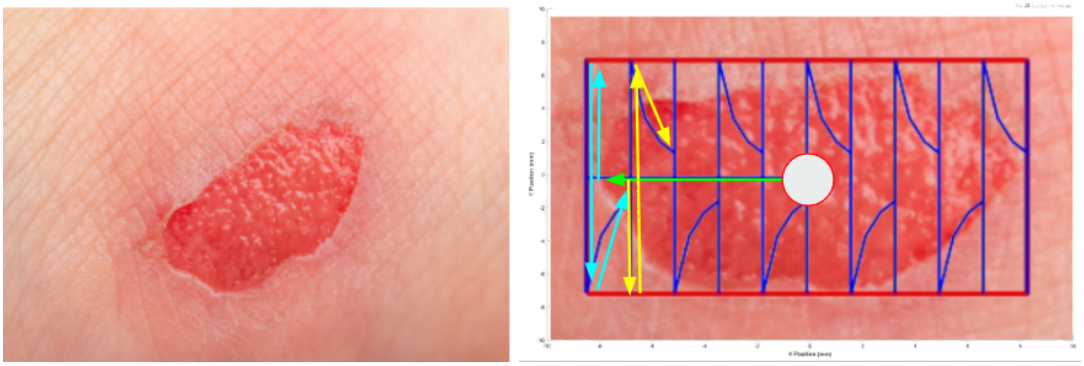


Figure 14: Raw wound image (left) with plotted toolpath visual representation overlaid (right)

This pattern for the toolpath will start in the center of the boundary box drawn around the wound by the PixyMon software and then will move to the top left corner before iterating over the pattern shown with arrows in the visual representation above. The cross section of the plasma torch is represented by the white dot in the image, equal to 1 mm in size. This toolpath allows for maximum coverage over the wound surface while minimizing the amount of healthy tissue contained in the main treatment area.

The toolpath was calibrated for accuracy through the application of experimentally determined scale factors. To accomplish this, a red object of a known size was placed in the camera's field of view and various scale factors were applied until the motion of the plasma torch over the surface matched the surface area of the test object. For the final setup of the working mechanism, the optimal scaling factor in the x direction was determined to be **0.5957** and **0.625** in the y direction.

Some design decisions were made in order to ensure compliance with the various codes and standards present in the industry. The first of which is ISO 13850:2015. This standard concerns the safety standards of equipment field machinery. As outlined in section 3.1, machines must have an emergency stop function in order to avert or reduce existing hazards to persons, damage to machinery, or to work in progress, and be initiated by a single human action. In order to establish compliance with this standard, an emergency stop button was incorporated into the design of the final prototype. If the end user engages this button, all operations of the machine will be stopped immediately, a red LED will illuminate, and the machine will require a reset by the end user prior to restarting operations. This will ensure the safety of the end user, the patient, and the machine in the event of a malfunction or an error.

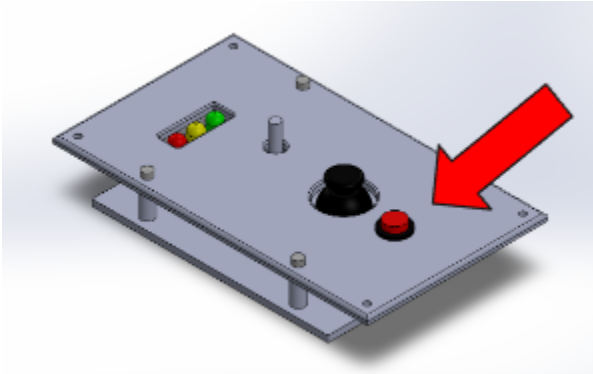


Figure 15: Emergency stop button as seen in final design

The next standard that was considered in the development of this mechanism was ISO 14971:2019. This standard regards risk management requirements for medical devices. Since this mechanism involves the use of plasma gas, there is an inherent risk for causing injury to the patient through standard operation. To comply with this standard, the final prototype will have two measures designed to ensure safety to the patient. Firstly, the manually controlled z axis assembly that holds the plasma torch will be manually adjustable with a wide range of motion to allow for the end user to control the plasma acting on the wound surface. The second mechanism to be incorporated into the final prototype to mitigate risk of injury to the patient is an infrared temperature sensor mounted on the z axis assembly as shown below:

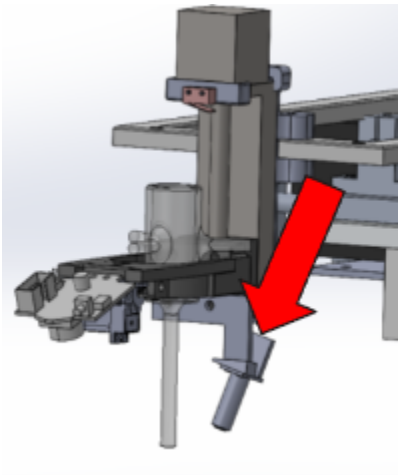


Figure 16: Z axis assembly with infrared temperature sensor to monitor plasma temperature

The infrared temperature sensor we used is a DCI non-contact infrared temperature sensor module with high precision. It has a high precision reading within a 50 cm range. The sensor is installed on the mount at an angle pointing at the area where the plasma and the skin will contact. During the plasma treatment, the temperature sensor has a safety threshold of 40

degree celsius. Any reading over the safety threshold will terminate the operation and pull the plasma torch away from the patient's wound to mitigate the risk of damaging the tissue of the patient.

II. Design Documentation

Our vertical assembly is composed of multiple subassemblies. First subassembly is our linear actuator system that drives the stage for our plasma torch, Pixy2 camera and Infrared sensor. Our pixy2 camera has its own sub assembly consisting of a micro servo motor that drives its viewing angle of the wound with a servo arm. A metal hook connected to a specific location on the Pixy camera module, allowing us to control the tilt of the camera for better a point of view when running our mechanism. Lastly, the IR sensor is mounted at an angle where its line of sight is coincident to the plasma's active sight where we can monitor the temperature of the skin should we need to relieve the plasma applied by raising the stage.

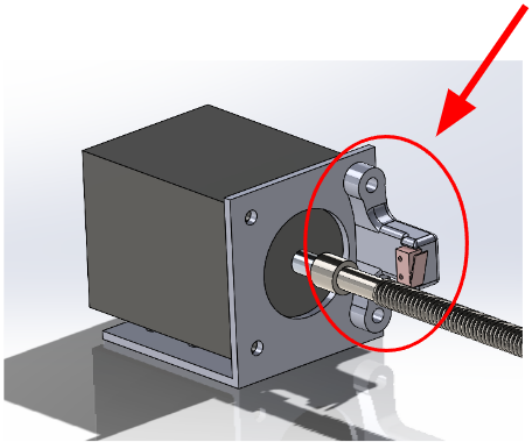


Figure 17: X-axis limit switch mount

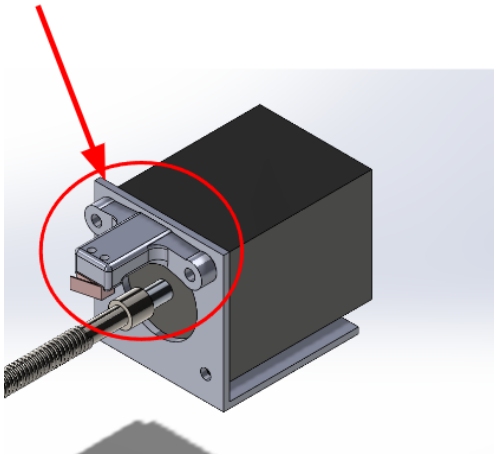


Figure 18: Y-axis limit switch mount

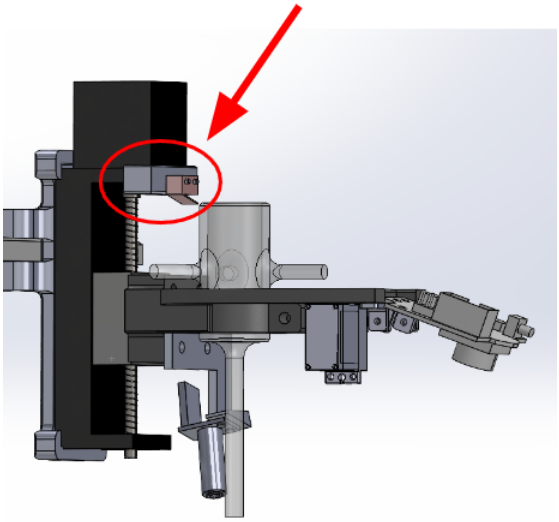


Figure 19: Z-axis limit switch mount

All of the physical controls available to the end user are contained on an external dashboard assembly as shown in **Figure 20**. The CAD detail drawing for this component is attached in **Appendix C.** From left to right, there are 4 main components contained in this assembly. Firstly, there are 3 colored LED lights that indicate the status of the machine during its operations. When the machine is in standby/ready mode, the green LED is illuminated. When the machine is executing an active operation (i.e. homing, toolpath movement), the yellow LED is illuminated. If the machine has an error or an emergency stop condition is triggered, the red LED will illuminate. To the right of the LED lights, there is a rotary encoder that allows the end user to manually adjust the height of the plasma torch via the Z-axis stepper motor assembly. Clockwise rotation of the encoder will move the plasma torch up and counter-clockwise rotation will move the plasma torch down. Next to the rotary encoder is the joystick that allows for manual control of the plasma torch's position in the x and y directions. Finally, there is the emergency stop button that if engaged, all operations of the machine will be stopped immediately. As previously mentioned, this is a safety measure that was implemented to be in compliance with ISO standard 13850:2015.

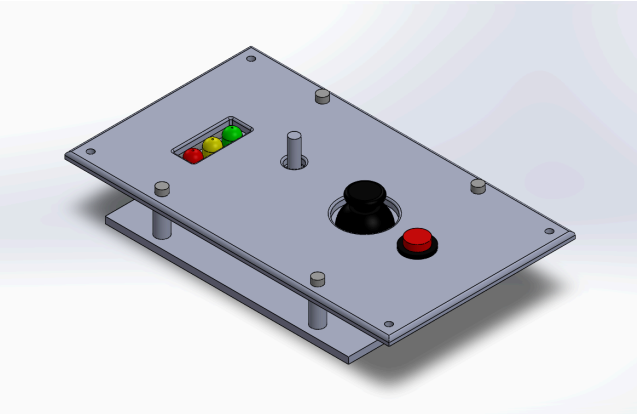


Figure 20: Dashboard Assembly, detail drawing referenced in Appendix.

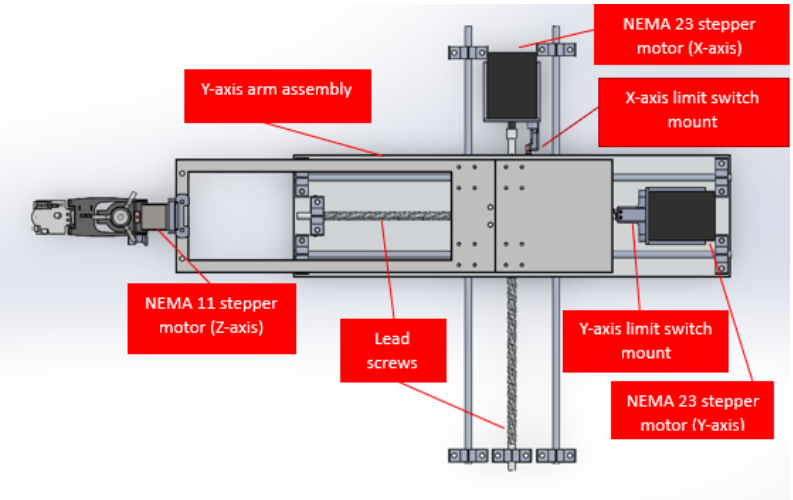


Figure 21: Final assembly with components labeled (1/2)

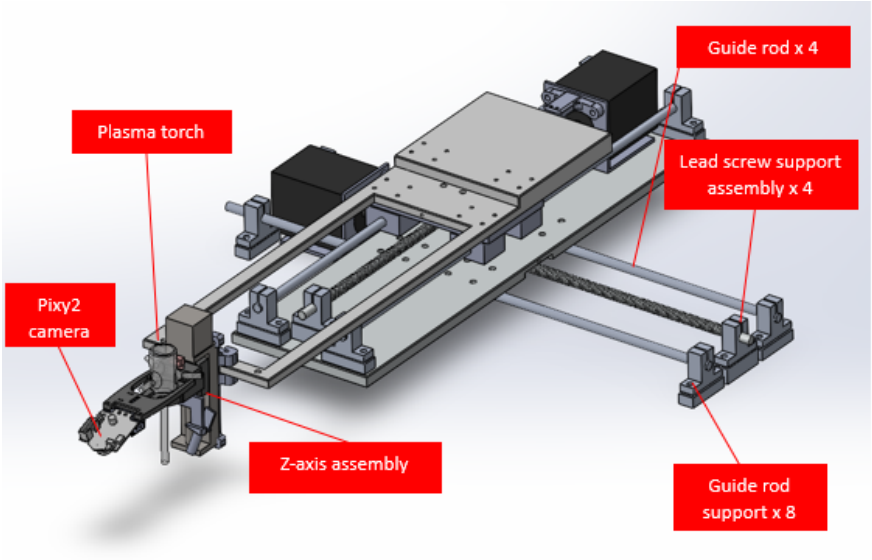


Figure 22: Final assembly with components labeled (2/2)

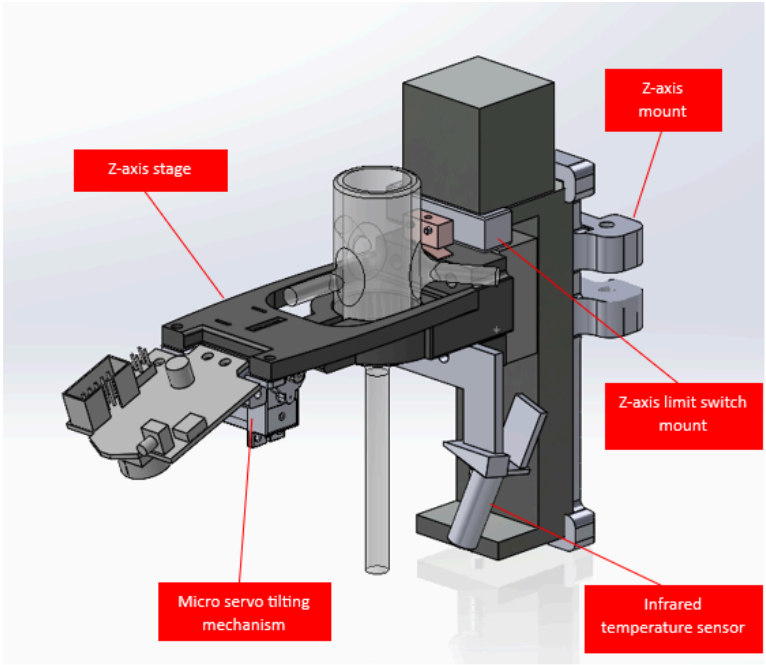


Figure 23: Z axis assembly with all components labeled

The control logic chart is meant to demonstrate that if every system functionality criterion is met, the machine is operating in a closed loop. While users have a digital readout displaying critical serial monitor information, LED indicating safety status, and a means to stop all functions immediately, the machine can be considered safe and fully operational. From left to

right on the chart depicted in **figure 24**, each subsequent component after the initial script upload to the Arduino IDE is dependent on the state of the components it is connected to. An OLED display, Arduino Mega 2560, and Pixy2 camera are the three start up components. Once the serial monitor indicates a homing status on the display, the DC power supply button needs to be triggered while the emergency button is at its highest 'reset' position. This allows 30V 1.5A power delivery to the three TB6600 micro stepper drivers sending pulses per revolution and direction signals to the Nema 23 bipolar stepper motors. User commands or manual operation can only occur after successfully reaching home position (0,0,0). This ensures the machine is properly calibrated to repeat the starting position after triggering each limit switch during startup. If the machine crashes into a limit switch either by user error during joystick or rotary encoder inputs, "Axis Crashed" is displayed with a red LED requiring user reset. Incorrect axis position coordinates that exceed the 300mm x 140mm x 85mm domain dimensions typed into the serial monitor will be prompted for a new command to be entered. Meanwhile, an IR sensor monitors the temperature at the wound surface. If triggered by an excess temperature signal, a red LED will power on to signal that the system has stopped and must be reset by the user. These functionalities are designed to prevent harm to the patient due to human or machine error. At startup, the Pixy2 camera waits for a Color Connected Components signature trigger that produces wound boundary center coordinates and rectangular dimensions displayed on the screen. On the PixyMon V2 interface, a boundary is drawn over the wound in the camera's view to provide the necessary values to type into the correct format to initiate a toolpath command.

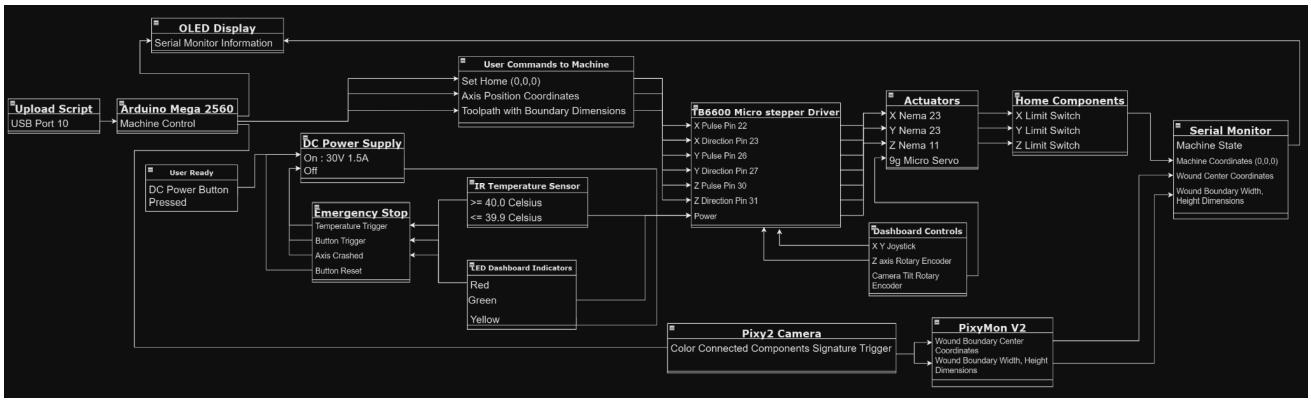


Figure 24: Machine control logic diagram for Arduino Script

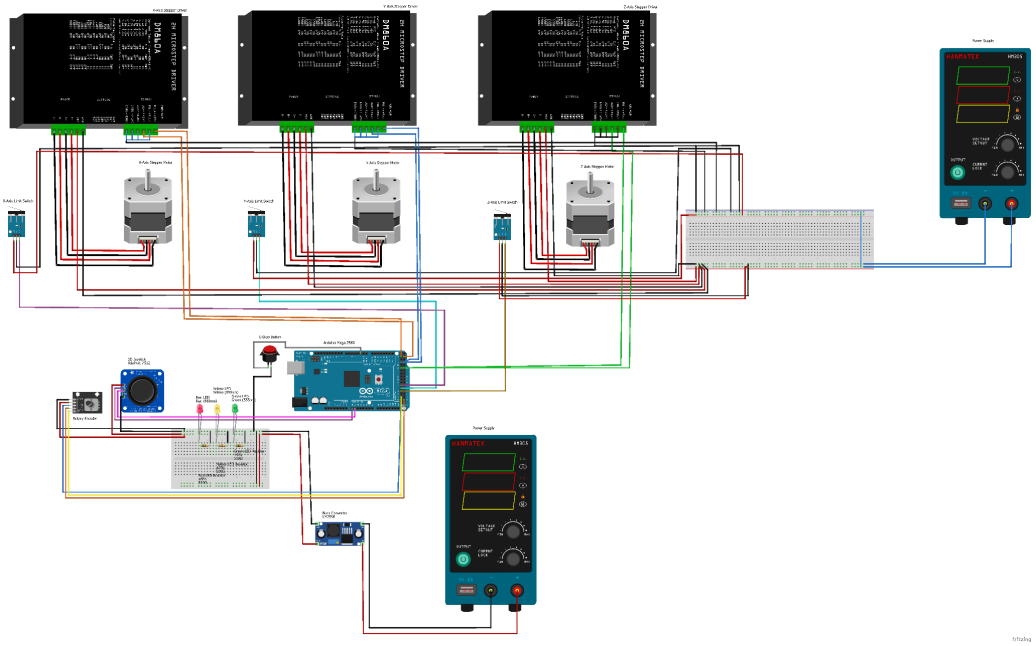


Figure 25: Wiring Diagram for electronic components on final prototype

Chapter 5: Fabrication and Assembly

I. Bill of Materials and Cost Analysis

Item/Component	Weblink	Vendor	Price	Amount	Total Price
Nema 23 Stepper motor	Befenybay 50mm Linear Stage Actuator	Amazon	$55.00	2x	$110.00
Heat Shrinks	Heat Shrink Tubing Kit	Amazon	$11.99	1x	$11.99
Wires	24 Gauge Solid Core Wire Kit	Amazon	$17.99	1x	$17.99
Arduino Mega 2560 Rev 3 Board	Arduino Mega 2560 REV3	Amazon	$48.99	2x	$97.98
USB Data Sync Cable	USB Data Sync Cable	Amazon	$8.58	2x	$17.16
Allen Key set	Hex Key Allen Wrench 26 Set	Amazon	$14.53	1x	$14.53
Micro Limit Switches	25 Pcs Limit Micro Switch	Amazon	$6.99	1x	$6.99
Temperature Sensor	Teyleten Robot IR Temperature Sensor Module	Amazon	$11.99	1x	$11.99
Threaded Inserts	Kadrick 420Pcs Threaded Inserts Assortment Kit	Amazon	$19.97	1x	$19.97
Shaft Couplers	Yeebyee 8mm to 8mm AL Shaft Coupler	Amazon	$12.99	1x	$12.99
Solderless Butt Connectors	Cionyce 100 Pcs Solderless Sleeve Heat Shrink Connectors	Amazon	$8.99	1x	$8.99
TB660 Stepper Motor Driver	EASON Stepper Motor Driver TB6600	Amazon	$12.89	1x	$12.89
DM556 Stepper Motor Driver	DM556 Stepper Motor Driver	Amazon	$16.68	2x	$33.36
Tiny Hex Socket Head Cap Screws	410 Pcs Tiny Hex Socket Head Cap Screws Bolts	Amazon	$8.99	1x	$8.99
Digital Camera Component	Charmed Labs Pixy2 Smart Vision Sensor	Amazon	$69.90	1x	$69.90
Digital Camera Component	Arducam 5MP SPI Camera	Arducam	$34.99	1x	$34.99
1 KG PLA	Polymaker PLA PRO Filament Dark Grey	Amazon	$24.99	1x	$24.99
PCB	PCB/Manufacture/Components	DigiKey	$15.44	1x	$15.44
Arduino Mega 2560 Header Pins	Treedix Header Pins	Amazon	$8.99	1x	$8.99
				Total	$540.13

Table 3: Bill of Materials for the 3 axis gantry robot

This Bill of Materials (BOM) was made possible by sourcing the most cost effective materials online. The cost could have been higher had we utilized higher quality materials, but we were inclined to develop our product with the least amount of funds required in order to gauge manufacturing costs should we decide to improve components like stepper drivers, 3D filament, etc,. Due to time constraints, the ability to source and calculate a product with higher end specs was not achievable, albeit with our newly acquired knowledge on manufacturing this product we could save further on materials if we invested more time on a Printed circuit board. A well thought out PCB would cut costs on wiring, solderless connectors, data cables and possibly a Micro Processing Unit should all be integrated into PCB's making large scale manufacturing easier with material sourcing and costs.

II. Assembly

In order for our mechanism to locate and travel to a wound, we implemented limit switches on the three stepper motors to identify the mechanism's workspace origin and maximum travel distance. With the aim of implementing these components we had to decide how to mount them, and the best locations for contact on the limit switch button faces were on the stepper motor mounts that were parallel to the X, Y, and Z stages. Anticipating a potential crash we ran various *Static Structural* simulations on different revisions of our mounts using Ansys. For every simulation we declared a 20N force impact on the faces of the stages in parallel with the faces of the limit switch buttons. Since an impact of that magnitude will most likely render the limit switches inoperable, we anticipated the expenditure of limit switches due to

being the item with the highest component count per dollar in our Bill of Materials. Our focus for the design was more so for the safety of the coupler and alignment of the leadscrew. Using NEMA 23 Stepper motors we can produce a high force to create an impact that could damage those components during our testing. With our understanding of Finite Element Analysis, our parameters for the simulation were the material being set to Polyethylene, whose material properties are most similar to PLA plastic which we printed from an Ender 3D printer. Other parameters added were the sizes and shapes of simulation elements, for the shapes of element nodes changed the amount of calculations supplementing the color gradient of the simulation making finer color translations in the Equivalent Stress, and Total Deformation plots. We made the decisions for the final design based on the simulation results with the best color gradient plots of the least amount of red color values as those represented as high amount of stress, and deformation.

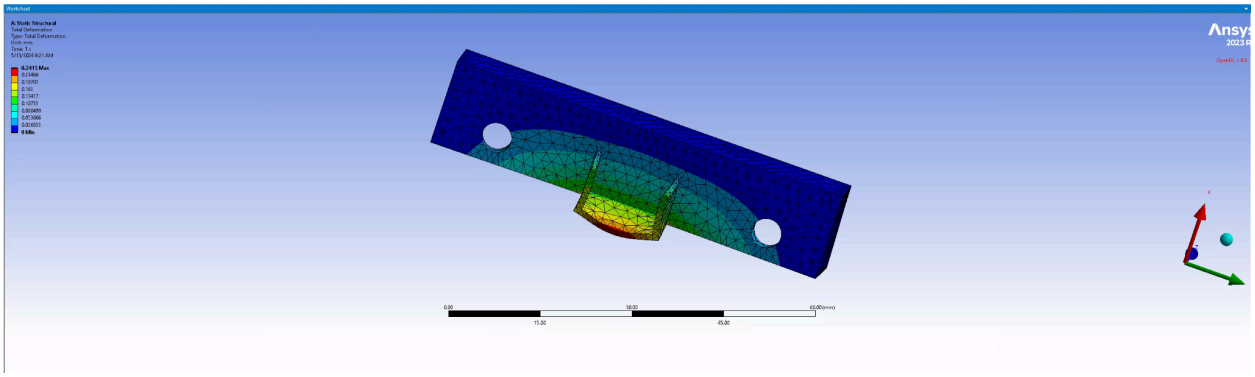


Figure 26: 0.24 mm for Total Deformation Test for Rev.1

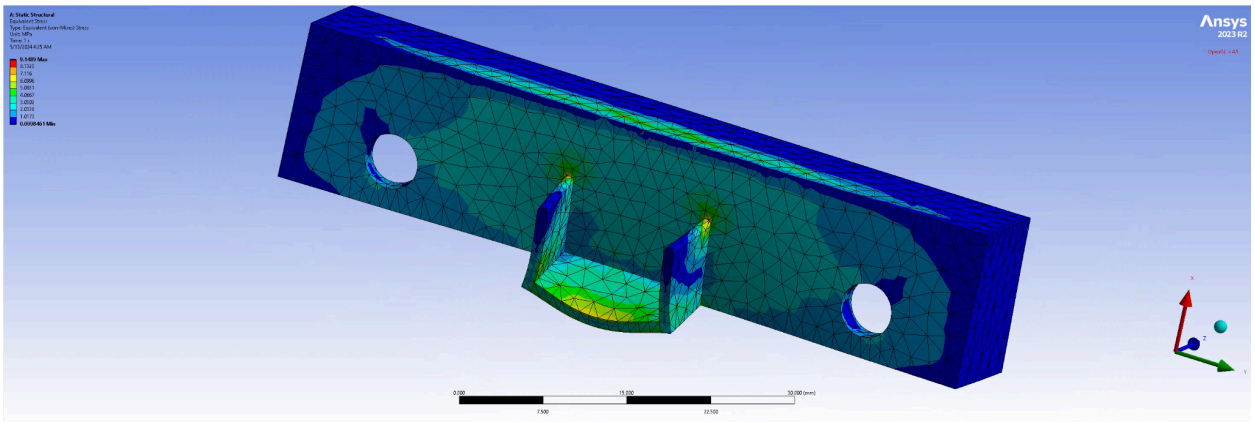


Figure 27: 9.15 Mpa forVon Mises Stress Test of Rev. 1

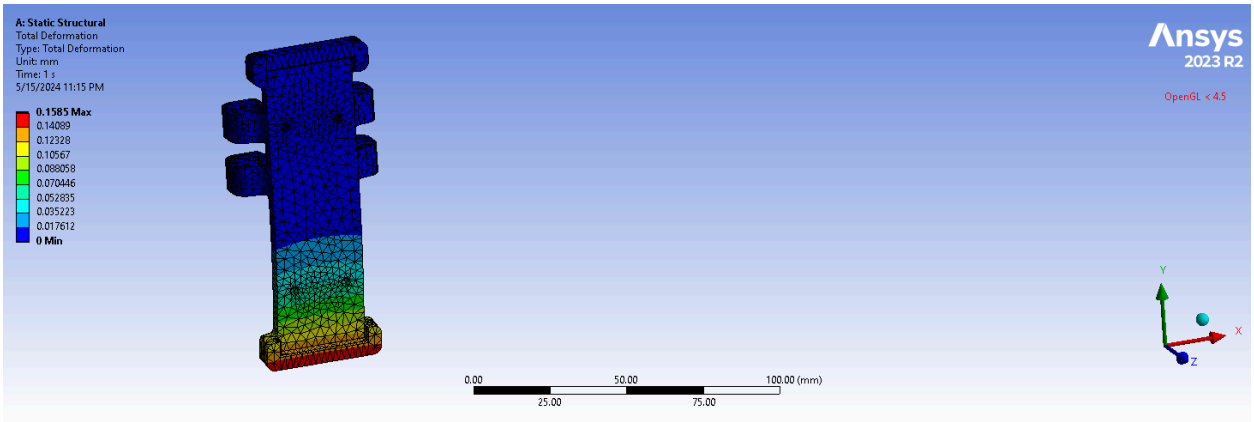


Figure 28: 0.16 mm for Total Deformation Test for Rev.4

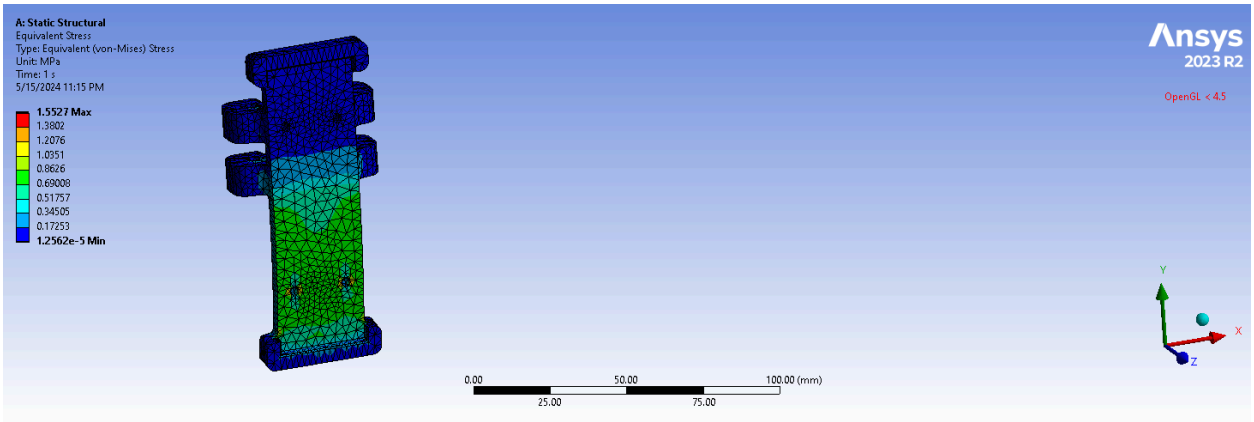


Figure 29: 1.55 Mpa forVon Mises Stress Test of Rev. 4

III. Fabrication

Using our Ender 3 Pro 3D printer, we manufactured the final prototypes with 485 grams of PLA printed at 195 degrees Celsius. SolidWorks model files were converted to gcode using the Ultimaker Cura 5.6.0 slicing software. Printing time totaled 48 hours and 19 minutes at a 0.20 mm layer height with 30% infill. Applying 0.03 mm horizontal expansion, -0.1 mm initial layer horizontal expansion, and 0.2 mm hole horizontal expansion modifiers ensured all hole features were printed within allowable specifications. Using a four layer raft and tree supports secured overhanging features to retain acceptable surface finish and structural integrity. An assortment of M2, M2.5, and M3 nuts and bolts were used to install the z axis assembly and its associated mounts. Each limit switch mount utilized two of the existing M3 nut and bolts mounting the Nema 23 motors. 15 N/m of torque was applied to each accessible mate on the machine using a ¼ inch torque wrench.

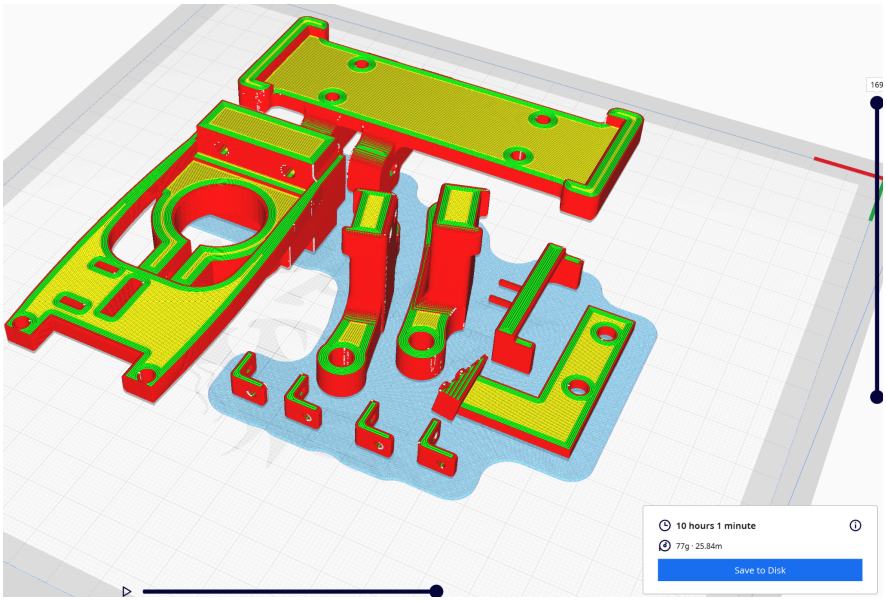


Figure 30: Ultimaker Cura sliced models showing filament weight and print time

Horizontal Expansion		0.03	mm
Initial Layer Horizontal Expansion	f_x	-0.1	mm
Hole Horizontal Expansion		0.2	mm

Figure 31: Expansion modifiers to ensure dimensional accuracy

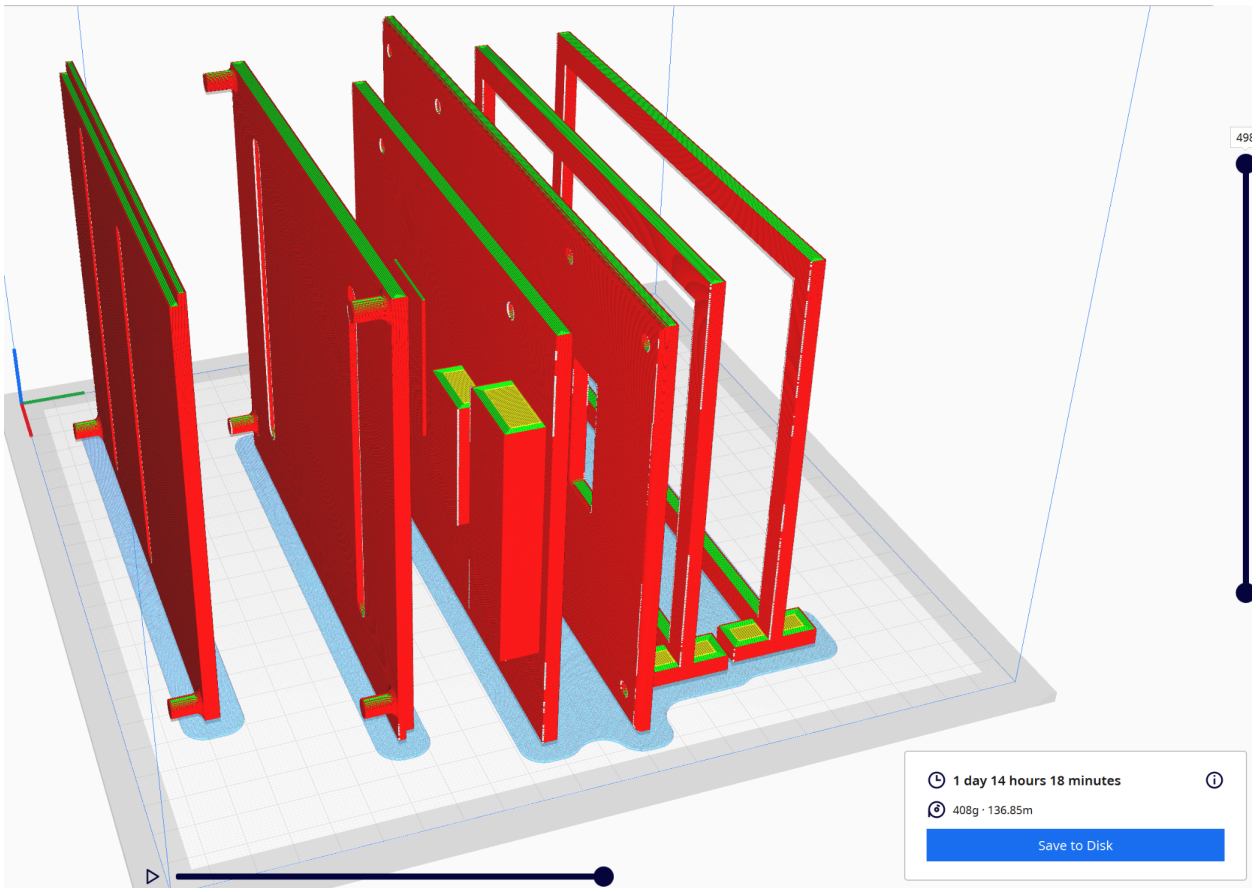


Figure 32: Sliced models of mounts carrying various components

Chapter 6: Testing, Results, and Analysis

In order to test the operation of the machine, we had to determine whether the mechanism could function as intended and meet all of its desired specifications. Part of this required the development of a set of commands that could allow the end user to control the machine in whichever way was desired. The commands are entered via the serial monitor and passed directly to the arduino to execute the action. The list of commands are shown below as follows:

Command	Function
“H”	Homing process, required to reset the machine in the event of an error or crash
“x -# y-# z-#”	Moves the plasma torch to the x,y, and z coordinates specified (replace # with desired number values)
“T w# h#”	Executes toolpath over boundary box of a user specified width (w) and height (h).

Table 4: List of Toolpath commands and their functions

The testing procedure for the mechanism involved taking a photo of a wound printed on an 8.5” x 11” sheet of paper and placing it on the table underneath the mechanism and testing the operation from start to finish to see if the mechanism functions accurately as intended. Some shortcomings of the mechanism were discovered during the testing process. As mentioned previously in chapter 3, the z axis assembly went through several iterations before taking its final form in rev 4. Additionally, the first version of the limit switch mounts were too fragile to withstand repeated machine crashes and errors, and thus had to be optimized with a stronger structure that could withstand the force of the loaded assembly weighing approximately 6 kg crashing into it at a maximum velocity of 24 mm/s.

Once consistent standard operation of the machine was accomplished and these issues were ironed out, the machine was tested using a live plasma setup as shown in **Figure 29**.

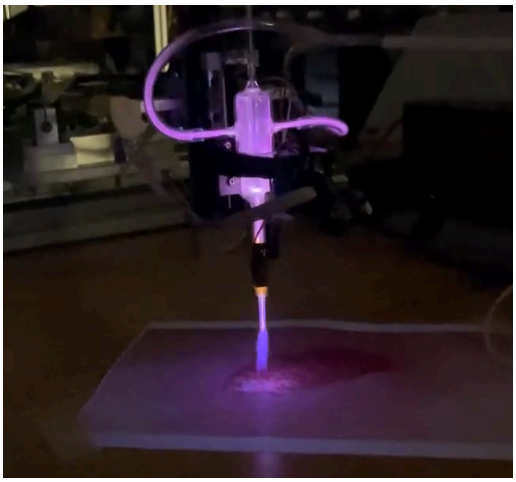


Figure 33: 3 axis gantry robot with live plasma in operation

These results solidify the success of this design and serve the purpose of providing a relatively painless means of sanitizing a wound whilst reducing recovery times. While CAP treatment already exists on a small scale, the current means of administering this treatment is imprecise as the current options are of a handheld variety. The fact that this machine is capable of automatically detecting a wound and delivering a custom treatment operation based upon the size of the wound as demonstrated during the testing phase, is groundbreaking for the medical industry and will contribute to many positive health outcomes as a result.

Chapter 7: Social Impacts

Improper wound sanitization has caused patients to avoid getting treatment due to negative experience in the past. While antiseptics are known to kill bacteria and microorganisms, they cause tremendous pain to the patient and delay the recovery time. The 3-Axis Machine for Wound Imaging and Plasma Medicine ensures patients receive a painless treatment that sanitizes wounds while promoting wound recovery. A quicker recovery time means that the patient's quality of life increases as they will be able to return to their normal life quicker. Due to the automation capabilities, the 3-axis Machine will reduce the need for professional training times and have the end goal of reducing patient treatment prices. Lastly, the goal is to improve the accessibility of Cold Atmospheric Plasma treatment for medical applicants.

Chapter 8: Conclusions and Future Work

This project successfully met all of the objectives that were declared earlier. Our team overhauled and optimized the previous mechanism design. A live camera system with image processing was also implemented as well as a manually controlled z axis assembly to vertically adjust the position of the plasma torch. Notable areas of success include the z axis assembly. The requirement of 50mm of z axis range of motion was exceeded as the final prototype has an 85mm range of motion in the z axis to allow the end user to sufficiently control the intensity of the plasma applied to the wound surface. Additionally, the image processing system is consistently capable of detecting wounds and moving the plasma torch in a toolpath over the wound with great accuracy.

While this project was overall a success, there are still a few areas that can be improved upon. Firstly, the final dimensions of the workspace area in the x and y dimensions are 218mm x 167mm. This area is sufficient for treating small wounds, however the mechanism could be improved to include a larger surface area that wounds can be treated with in a future iteration. Secondly, the software for the mechanism can be improved to be more automated and require less input from the end user to complete its operations. Lastly, the wiring setup and final assembly of the mechanism could be improved in the future as well as the implementation of a central PCB board to reduce the wiring complexity and streamline the final form of the mechanism. The project deadline for this rendition of the project did not allow for desired results in this area due to complications and setbacks that occurred during the project.

This project provided valuable experiences for all of those involved. The contributors to this project gained experience working on a long term, large scale project. They also got experience working as a team and delegating responsibilities to ensure that objectives were completed and deadlines were met. They also got experience designing and prototyping a complex mechanism with multiple independent systems and components. Lastly, all contributors got experience applying engineering principles to solve a real world problem.

This project did not come without its challenges. Firstly, coming to agreements for major design decisions was a challenge since every team member had their own perspectives and finding solutions that satisfied everyone was a challenge at times. Secondly, scheduling and time management was another challenge as there were multiple unforeseen complications and setbacks experienced with this project as well as working with the various schedules and availability of the team members. Lastly, all of the team members had to learn how to use tools that they have no prior knowledge or experience with and configure them to work successfully in the project. Our project was completed as cheaply as possible but unfortunately went $40.13 over budget.

References

Akyol, Gokcenaz. *What is image Processing* . 13 Jan. 2023. *Medium*, https://medium.com/gokcenazakyol/1-what-is-digital-image-processing-image-processing-2da13b5dfa9c.

"Ball Screws - Lead vs Pitch", *Thomson,* https://www.thomsonlinear.com/en/training/ball_screws/lead_vs_pitch, accessed 12 May 2024.

Bárdos L, Baránková H, "Cold atmospheric plasma: Sources, processes, and applications", *Thin Solid Films*, vol. 518, no. 23, 2010, pp. 6705-6713. https://doi.org/10.1016/j.tsf.2010.07.044

Boeckmann, Lars, et al. "Cold Atmospheric Pressure Plasma in Wound Healing and Cancer Treatment." *MDPI*, Multidisciplinary Digital Publishing Institute, 1 Oct. 2020, www.mdpi.com/2076-3417/10/19/6898.

Bolgeo T, Maconi A, et. al, "The Role of Cold Atmospheric Plasma in Wound Healing Processes in Critically Ill Patients", *Journal of Personalized Medicine*, vol. 13, no. 5, 2023, pp. 736. https://doi.org/10.3390/jpm13050736

Lou, Bih-Show, et al. "Helium/Argon-Generated Cold Atmospheric Plasma Facilitates Cutaneous Wound Healing." *Frontiers*, Frontiers, 2 June 2020, www.frontiersin.org/articles/10.3389/fbioe.2020.00683/full

Rueda-Fernández, Manuel, et al. "Effect of the Most Common Wound Antiseptics on Human Skin Fibroblasts." *Clinical and Experimental Dermatology*, U.S. National Library of Medicine, Aug. 2022,www.ncbi.nlm.nih.gov/pmc/articles/PMC9545306/.'

"What is Image Processing: Overview, Applications, Benefits, and More", *Simplilearn*, 2023, https://www.simplilearn.com/image-processing-article. Accessed 29 Apr. 2024.

Appendices

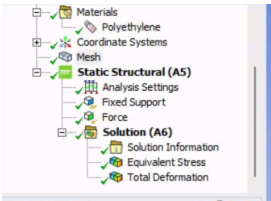
Appendix A: FEA Analysis: X limit Switch Mount

Outline of Schematic A2: Engineering Data

	A	B	C	D	E
1	Contents of Engineering Data			Source	Description
2	Material				
3	Polyethylene				
*	Click here to add a new material				

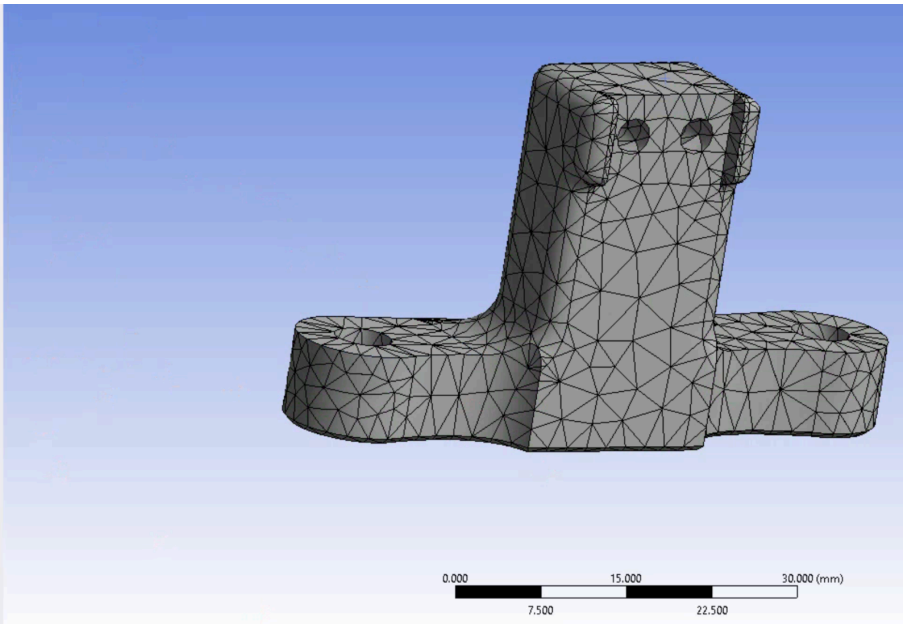
Properties of Outline Row 3: Polyethylene

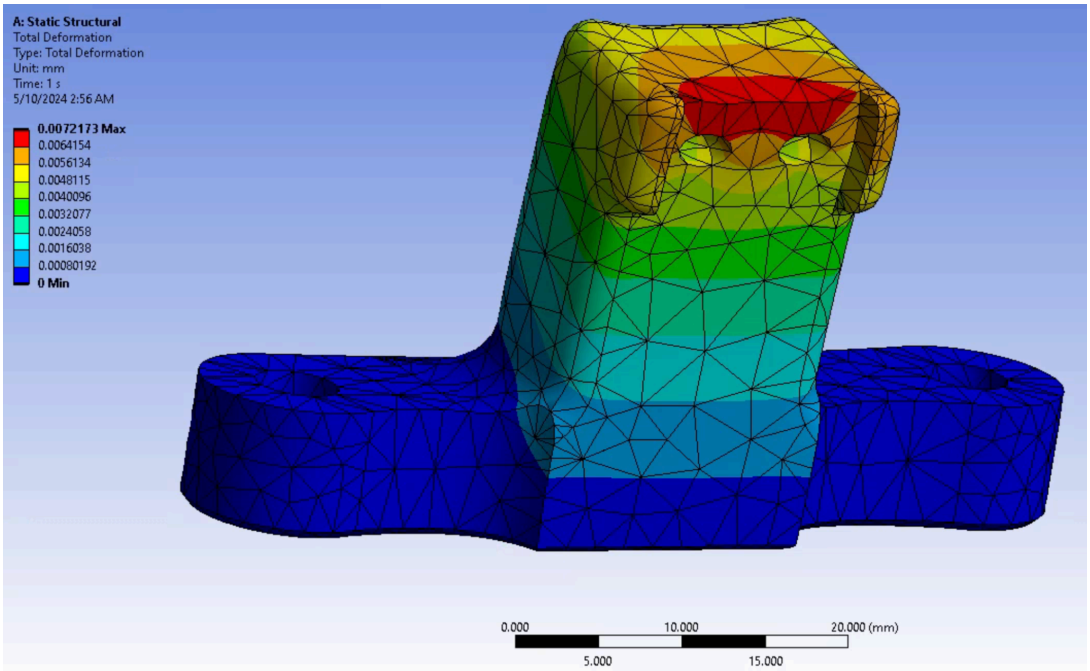
	A	B	C	D	E
1	Property	Value	Unit		
2	Material Field Variables	Table			
3	Density	1260	kg m^-3		
4	Isotropic Secant Coefficient of Thermal Expansion				
6	Isotropic Elasticity				
7	Derive from	Youn...			
8	Young's Modulus	1.1E+09	Pa		
9	Poisson's Ratio	0.42			
10	Bulk Modulus	2.2917E+09	Pa		
11	Shear Modulus	3.8732E+08	Pa		
12	Tensile Yield Strength	2.5E+07	Pa		
13	Compressive Yield Strength	0	Pa		
14	Tensile Ultimate Strength	3.3E+07	Pa		
15	Compressive Ultimate Strength	0	Pa		
16	Isotropic Thermal Conductivity	0.28	W m^...		
17	Specific Heat Constant Pressure, C_p	2300	J kg^...		

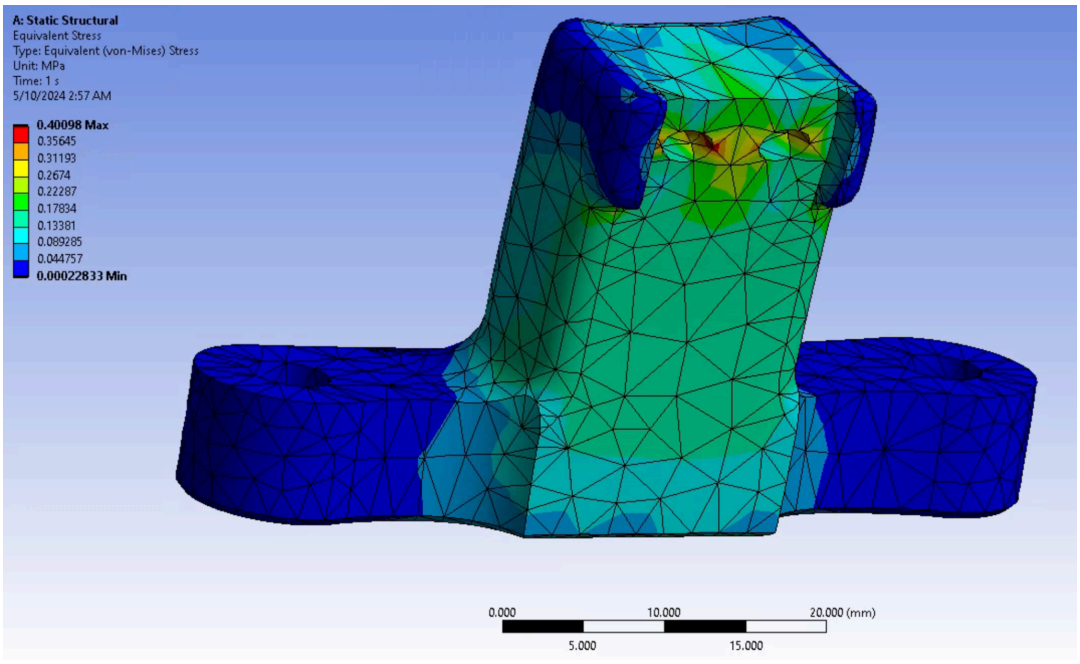


Details of "Mesh"

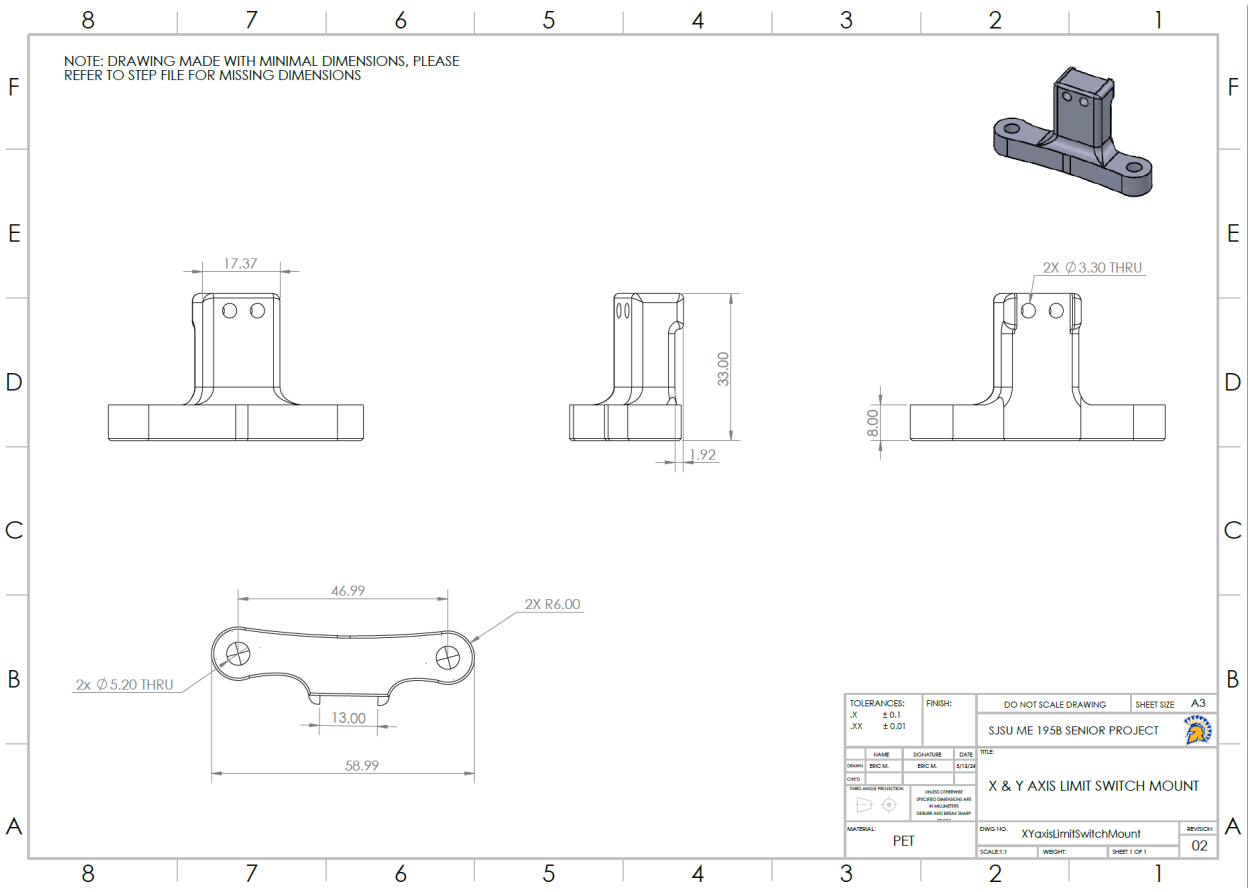
Display	
Display Style	Use Geometry Setting
Defaults	
Physics Preference	Mechanical
Element Order	Program Controlled
Element Size	2.0 mm
Sizing	
Use Adaptive Sizing	Yes
Resolution	Default (2)
Mesh Defeaturing	Yes
Defeature Size	Default
Transition	Fast
Span Angle Center	Coarse
Initial Size Seed	Assembly
Bounding Box Diagonal	70.751 mm
Average Surface Area	66.385 mm²
Minimum Edge Length	0.11077 mm
Quality	
Inflation	
Advanced	
Statistics	
Nodes	6884
Elements	3693
Show Detailed Statistics	No

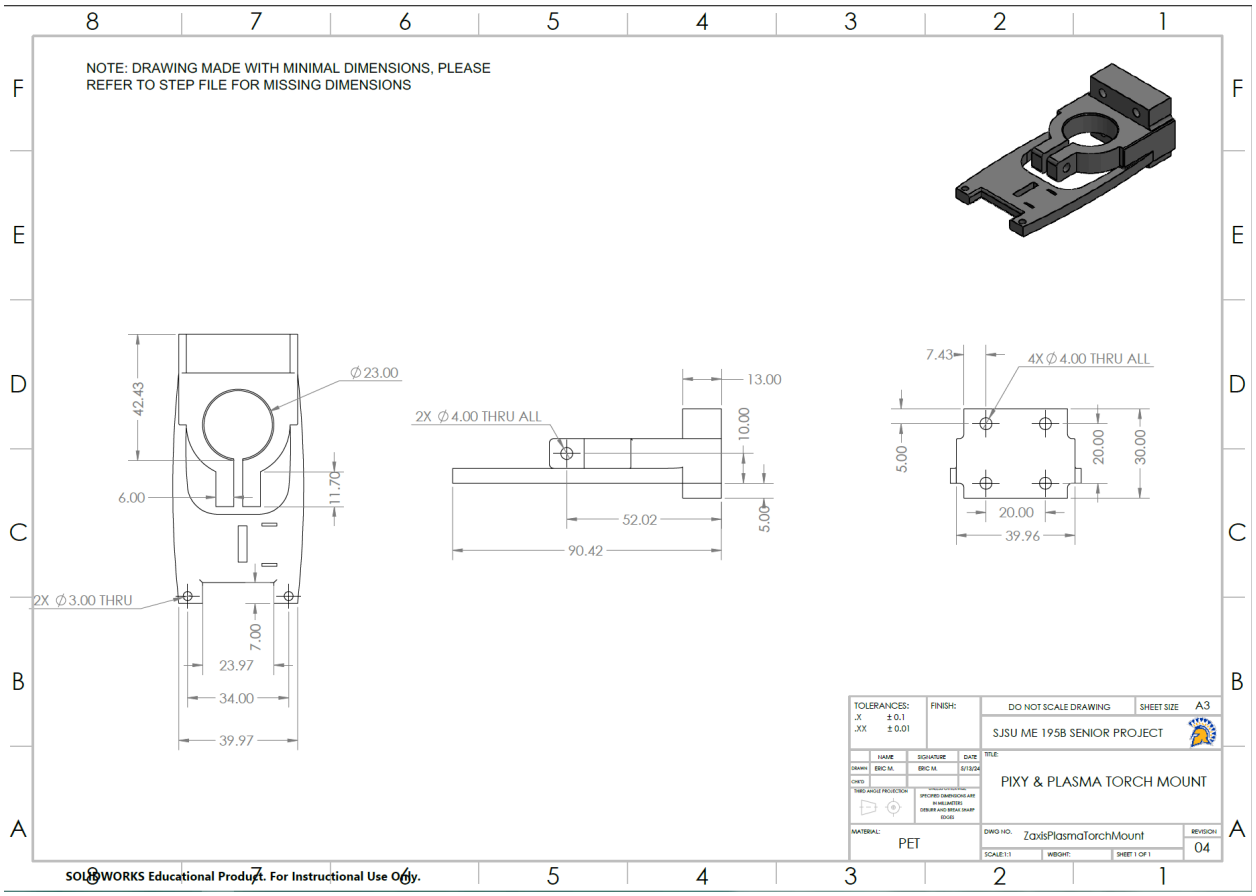


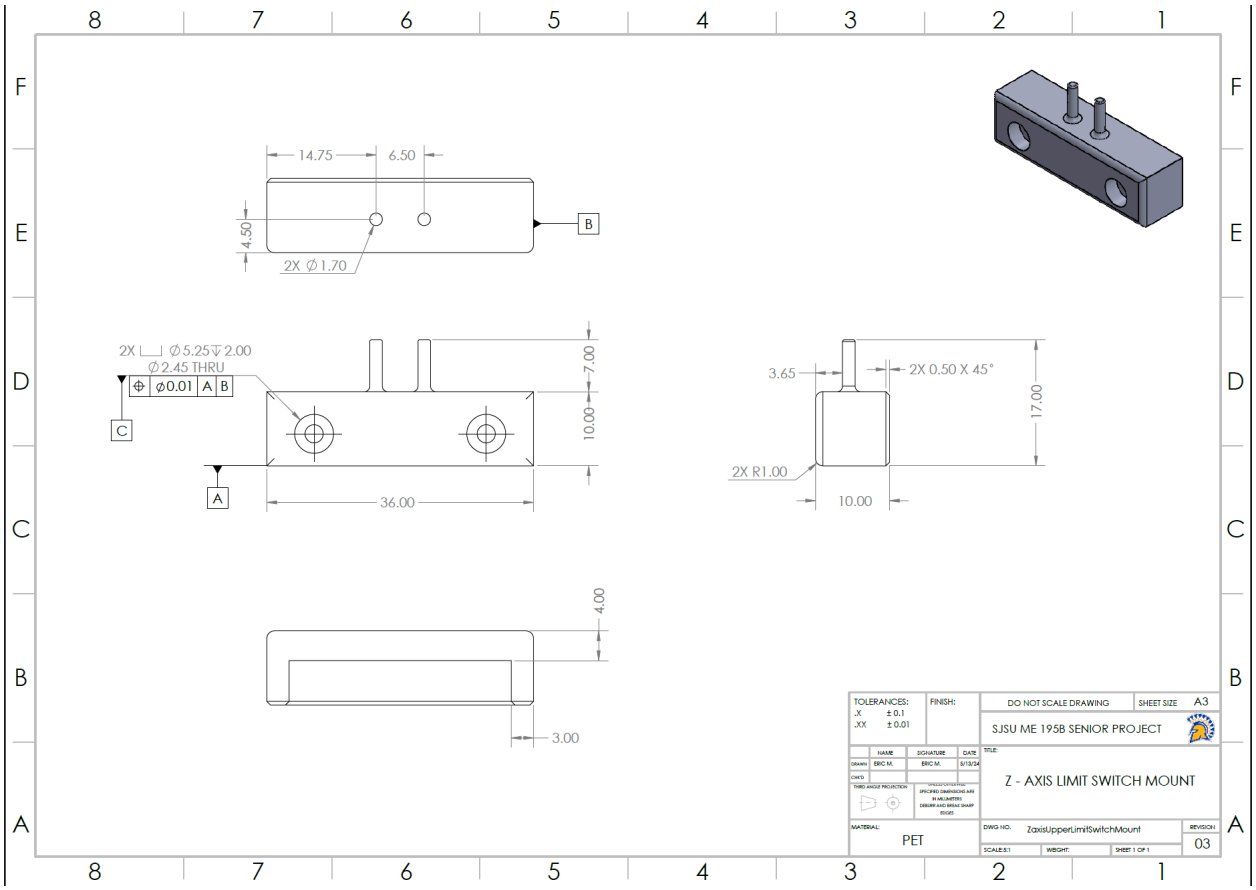

A: Static Structural
Total Deformation
Type: Total Deformation
Unit: mm
Time: 1 s
5/10/2024 2:56 AM
0.0072173 Max
0.0064154
0.0056134
0.0048115
0.0040096
0.0032077
0.0024058
0.0016038
0.00080192
0 Min
0.000
10.000
20.000 (mm)
5.000
15.000

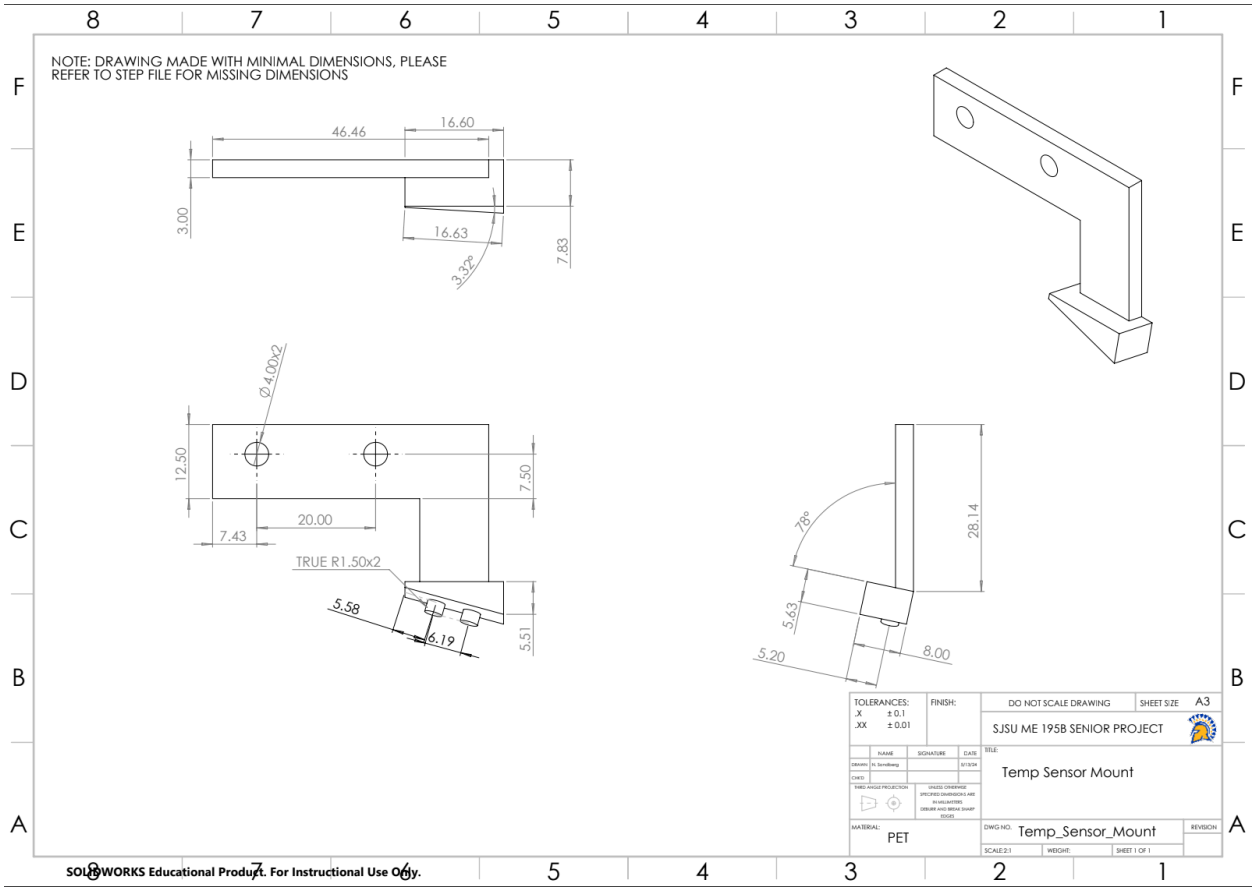

A: Static Structural
Equivalent Stress
Type: Equivalent (von-Mises) Stress
Unit: MPa
Time: 1 s
5/10/2024 2:57 AM
0.40098 Max
0.35645
0.31193
0.2674
0.22287
0.17834
0.13381
0.089285
0.044757
0.00022833 Min
0.000
10.000
20.000 (mm)
5.000
15.000

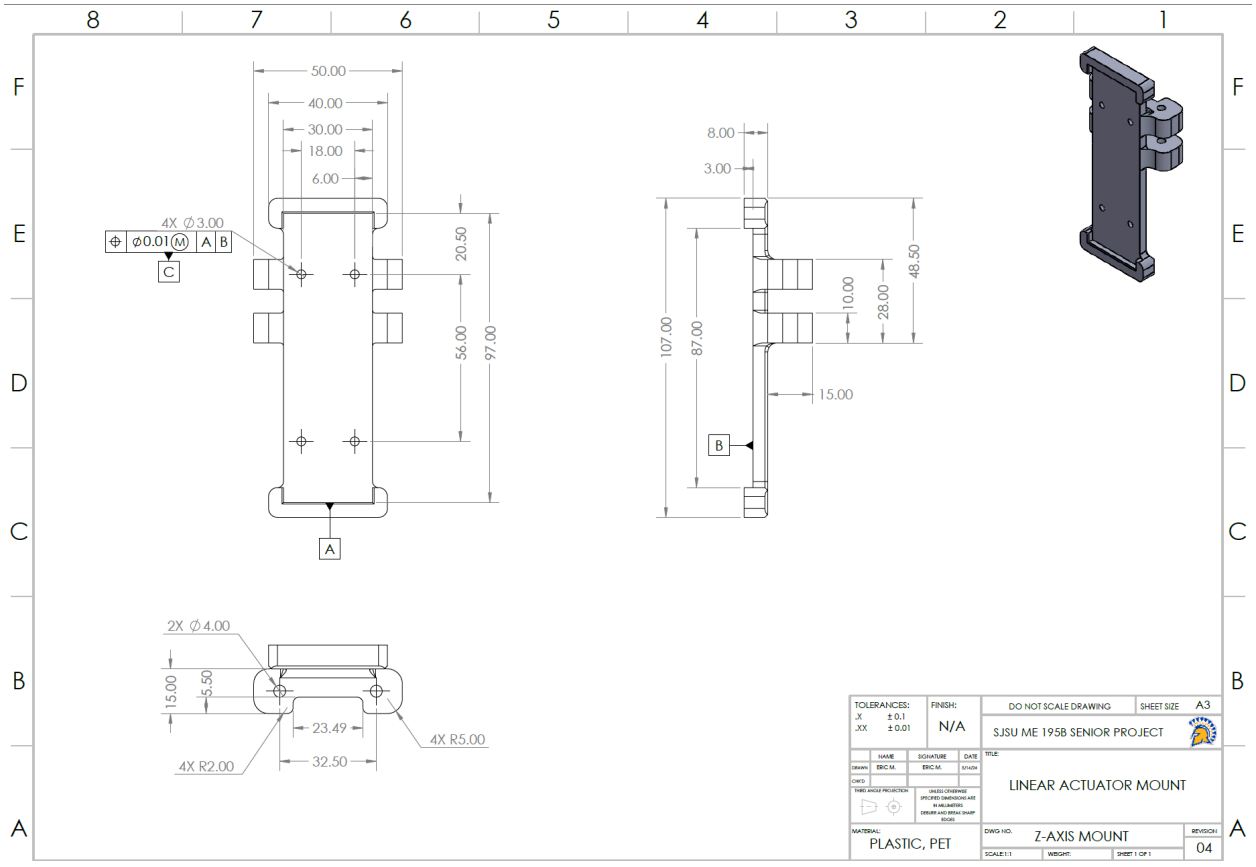
Appendix B: CAD Drawings



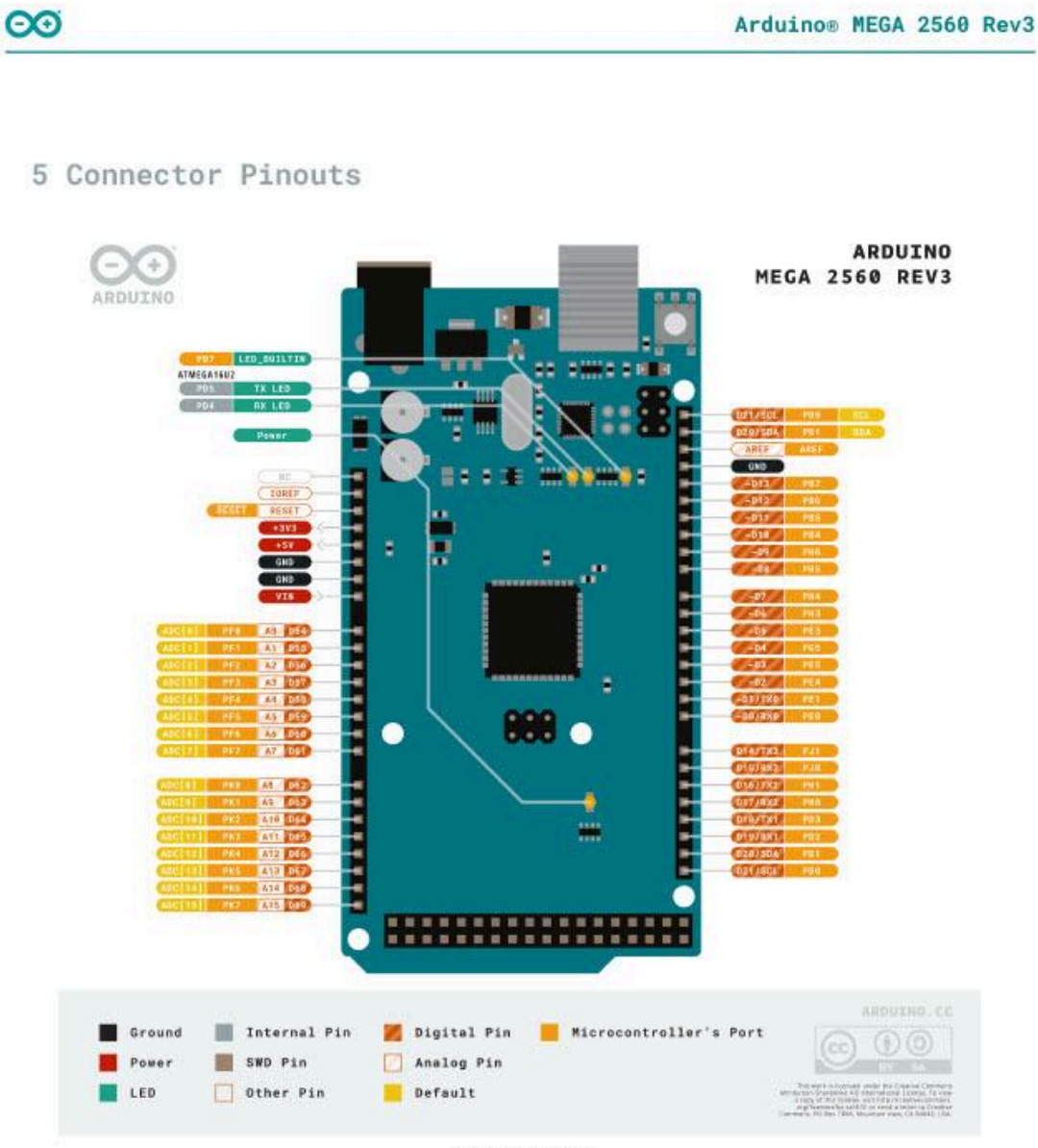

NOTE: DRAWING MADE WITH MINIMAL DIMENSIONS, PLEASE
REFER TO STEP FILE FOR MISSING DIMENSIONS
42.43
Ø23.00
6.00
11.70
2X Ø3.00 THRU
7.00
23.97
34.00
39.97
2X Ø4.00 THRU ALL
13.00
10.00
5.00
52.02
90.42
7.43
4X Ø4.00 THRU ALL
5.00
20.00
30.00
20.00
39.96
TOLERANCES:
.X ± 0.1
.XX ± 0.01
FINISH:
DO NOT SCALE DRAWING
SHEET SIZE A3
SJSU ME 195B SENIOR PROJECT
NAME SIGNATURE DATE
DRAWN ERIC M. ERIC M. 5/13/24
CHK'D
TITLE:
PIXY & PLASMA TORCH MOUNT
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: 125 µin BREAK SHARP EDGES
MATERIAL:
PET
DWG NO. ZaxisPlasmaTorchMount
REVISION 04
SCALE:1:1 WEIGHT: SHEET 1 OF 1
SOLIDWORKS Educational Product. For Instructional Use Only.


14.75
6.50
4.50
2X Ø1.70
B
2X Ø5.25↧2.00
Ø2.45 THRU
⌖ Ø0.01 A B
C
A
7.00
10.00
36.00
3.65
2X 0.50 X 45°
17.00
2X R1.00
10.00
4.00
3.00
TOLERANCES:
.X ± 0.1
.XX ± 0.01
FINISH:
DO NOT SCALE DRAWING
SHEET SIZE A3
SJSU ME 195B SENIOR PROJECT
NAME SIGNATURE DATE
DRAWN ERIC M. ERIC M. 5/13/24
CHK'D
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS
DEBURR AND BREAK SHARP EDGES
TITLE:
Z - AXIS LIMIT SWITCH MOUNT
MATERIAL:
PET
DWG NO. ZaxisUpperLimitSwitchMount
REVISION 03
SCALE:5:1
WEIGHT:
SHEET 1 OF 1


NOTE: DRAWING MADE WITH MINIMAL DIMENSIONS, PLEASE
REFER TO STEP FILE FOR MISSING DIMENSIONS
46.46
16.60
3.00
16.63
7.83
3.32°
Ø4.00x2
12.50
7.50
20.00
7.43
TRUE R1.50x2
5.58
6.19
5.51
78°
28.14
5.63
5.20
8.00
TOLERANCES:
.X ±0.1
.XX ±0.01
FINISH:
DO NOT SCALE DRAWING
SHEET SIZE A3
SJSU ME 195B SENIOR PROJECT
NAME
SIGNATURE
DATE
DRAWN N. Terrelberg 8/18/24
CHK'D
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS DEBURR AND BREAK SHARP EDGES
TITLE:
Temp Sensor Mount
MATERIAL:
PET
DWG NO. Temp_Sensor_Mount
REVISION
SCALE:2:1
WEIGHT:
SHEET 1 OF 1
SOLIDWORKS Educational Product. For Instructional Use Only.


8
7
6
5
4
3
2
1
F
E
D
C
B
A
50.00
40.00
30.00
18.00
6.00
4X ∅3.00
⌖ ∅0.01 M A B
C
20.50
56.00
97.00
A
8.00
3.00
107.00
87.00
10.00
28.00
48.50
15.00
B
2X ∅4.00
15.00
5.50
23.49
32.50
4X R5.00
4X R2.00
TOLERANCES:
.X ± 0.1
.XX ± 0.01
FINISH:
N/A
DO NOT SCALE DRAWING
SHEET SIZE A3
SJSU ME 195B SENIOR PROJECT
NAME
SIGNATURE
DATE
DRAWN
ERIC M.
ERIC M.
5/14/24
CHK'D
THIRD ANGLE PROJECTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS
DEBURR AND BREAK SHARP EDGES
TITLE:
LINEAR ACTUATOR MOUNT
MATERIAL:
PLASTIC, PET
DWG NO.
Z-AXIS MOUNT
REVISION
04
SCALE:1:1
WEIGHT:
SHEET 1 OF 1

Appendix C: Component Datasheets



Arduino Mega Pinout

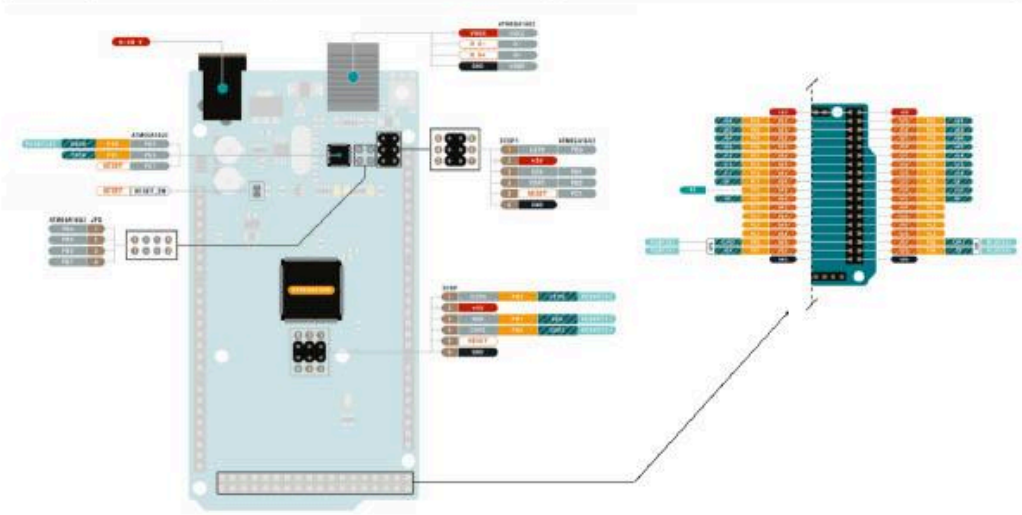
5.1 Analog

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega Pinout

5.3 ATMEGA16U2 JP5

Pin	Function	Type	Description
1	PB4	Internal	Serial Wire Debug
2	PB6	Internal	Serial Wire Debug
3	PB5	Internal	Serial Wire Debug
4	PB7	Internal	Serial Wire Debug

5.4 ATMEGA16U2 ICSP1

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

5.5 Digital Pins D22 - D53 LHS

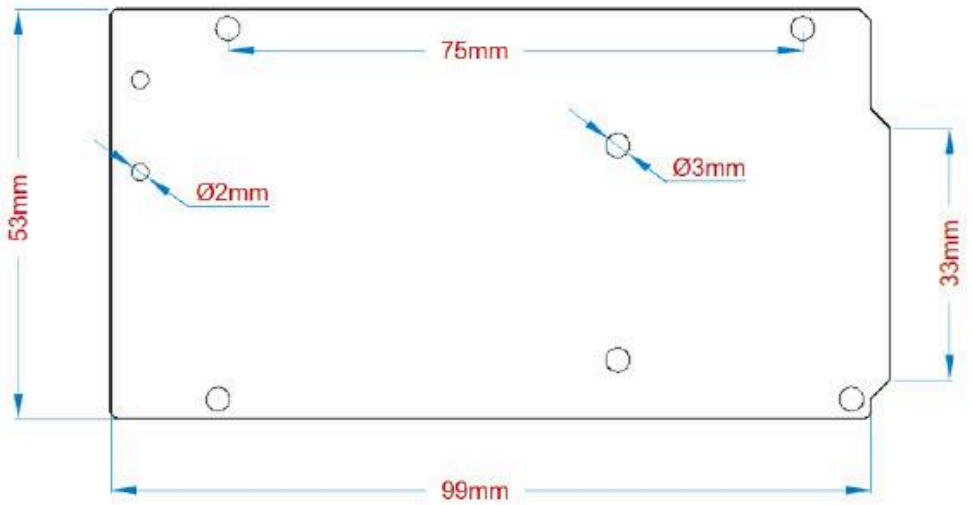
Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D22	Digital	Digital input 22/GPIO
3	D24	Digital	Digital input 24/GPIO
4	D26	Digital	Digital input 26/GPIO
5	D28	Digital	Digital input 28/GPIO
6	D30	Digital	Digital input 30/GPIO
7	D32	Digital	Digital input 32/GPIO
8	D34	Digital	Digital input 34/GPIO
9	D36	Digital	Digital input 36/GPIO
10	D38	Digital	Digital input 38/GPIO
11	D40	Digital	Digital input 40/GPIO
12	D42	Digital	Digital input 42/GPIO
13	D44	Digital	Digital input 44/GPIO
14	D46	Digital	Digital input 46/GPIO
15	D48	Digital	Digital input 48/GPIO
16	D50	Digital	Digital input 50/GPIO
17	D52	Digital	Digital input 52/GPIO
18	GND	Power	Ground

5.6 Digital Pins D22 - D53 RHS

Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D23	Digital	Digital input 23/GPIO
3	D25	Digital	Digital input 25/GPIO
4	D27	Digital	Digital input 27/GPIO
5	D29	Digital	Digital input 29/GPIO
6	D31	Digital	Digital input 31/GPIO
7	D33	Digital	Digital input 33/GPIO
8	D35	Digital	Digital input 35/GPIO
9	D37	Digital	Digital input 37/GPIO
10	D39	Digital	Digital input 39/GPIO
11	D41	Digital	Digital input 41/GPIO
12	D43	Digital	Digital input 43/GPIO
13	D45	Digital	Digital input 45/GPIO
14	D47	Digital	Digital input 47/GPIO
15	D49	Digital	Digital input 49/GPIO
16	D51	Digital	Digital input 51/GPIO
17	D53	Digital	Digital input 53/GPIO
18	GND	Power	Ground

6 Mechanical Information

6.1 Board Outline



wiki:v2:start [Documentation] (pixycam.com)

Pixy 2 CMUcam5 Smart Vision Sensor

SKU 102991074

Pixy2 is the second version of Pixy. It's faster, smaller and more capable than the original Pixy, adding line tracking/following algorithms as well as other features.

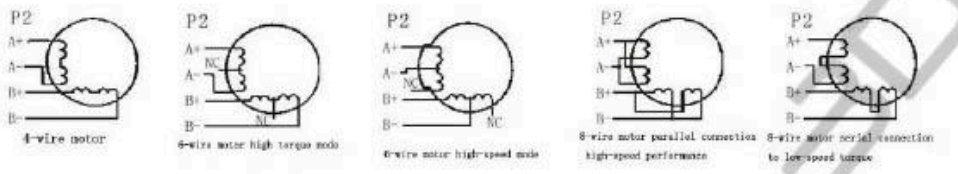
Here's what we've added to Pixy2:

- Pixy2 detects lines, intersections and small barcodes, intended for line-following robots
- Improved framerate – 60 frames-per-second
- Tracking algorithms have been added to color-based object detection
- Improved and simplified libraries for Arduino, Raspberry Pi and other controllers
- Integrated light source

TB6600 Stepper Motor Driver

Stepper Motor Wiring

Two-phase 4-wire, 6-wire, 8-wire motor wiring, as shown below:



DIP Switch Settings

Micro-Step Setting

The follow table shows the TB6600 Driver Micro step settings. The first 3 DIP switches are used to set the micro steps.

Step Angle = Motor Step Angle / Micro Step

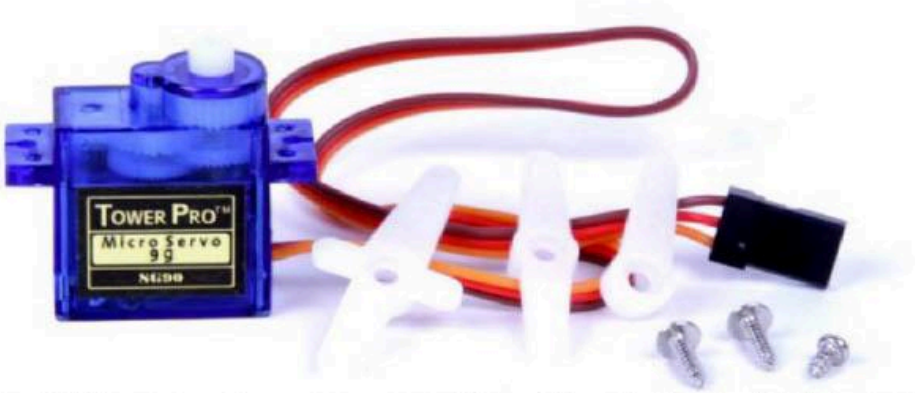
E.g. A stepper motor with a 1.8° step angle, the final step angle under "Micro step 4" will be 1.8°/4=0.45°

Micro Step	Pulse/Rev	S1	S2	S3
NC	NC	ON	ON	ON
1	200	ON	ON	OFF
2/A	400	ON	OFF	ON
2/B	400	OFF	ON	ON
4	800	ON	OFF	OFF
8	1600	OFF	ON	OFF
16	3200	OFF	OFF	ON
32	6400	OFF	OFF	OFF

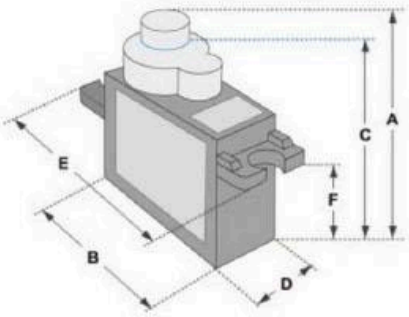
Current Control Setting

Current (A)	S4	S5	S6
0.5	ON	ON	ON
1.0	ON	OFF	ON
1.5	ON	ON	OFF
2.0	ON	OFF	OFF
2.5	OFF	ON	ON
2.8	OFF	OFF	ON
3.0	OFF	ON	OFF
3.5	OFF	OFF	OFF

SERVO MOTOR SG90 DATA SHEET

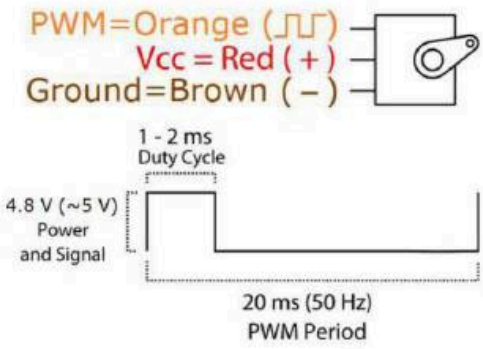


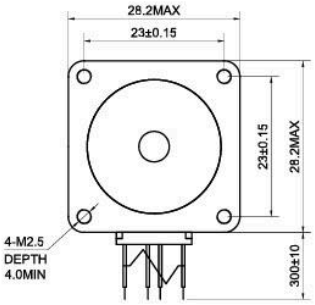
Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

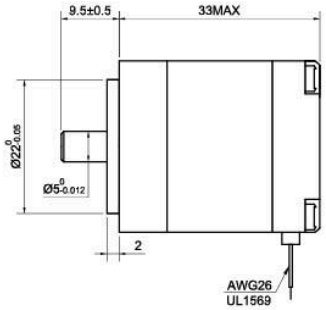


Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.





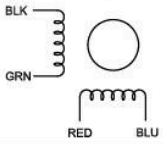


SPECIFICATION / CONNECTION	BIPOLAR
AMPS/PHASE	0.70
RESISTANCE/PHASE(Ohms)@25°C	5.30±10%
INDUCTANCE/PHASE(mH)@1KHz	3.70±20%
HOLDING TORQUE(Nm)[lb-in]	0.07[0.62]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	9.00
WEIGHT(Kg)[lb]	0.11[0.24]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL;FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm (UNDER NORMAL TEMPERATURE AND HUMIDITY)	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

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

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

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

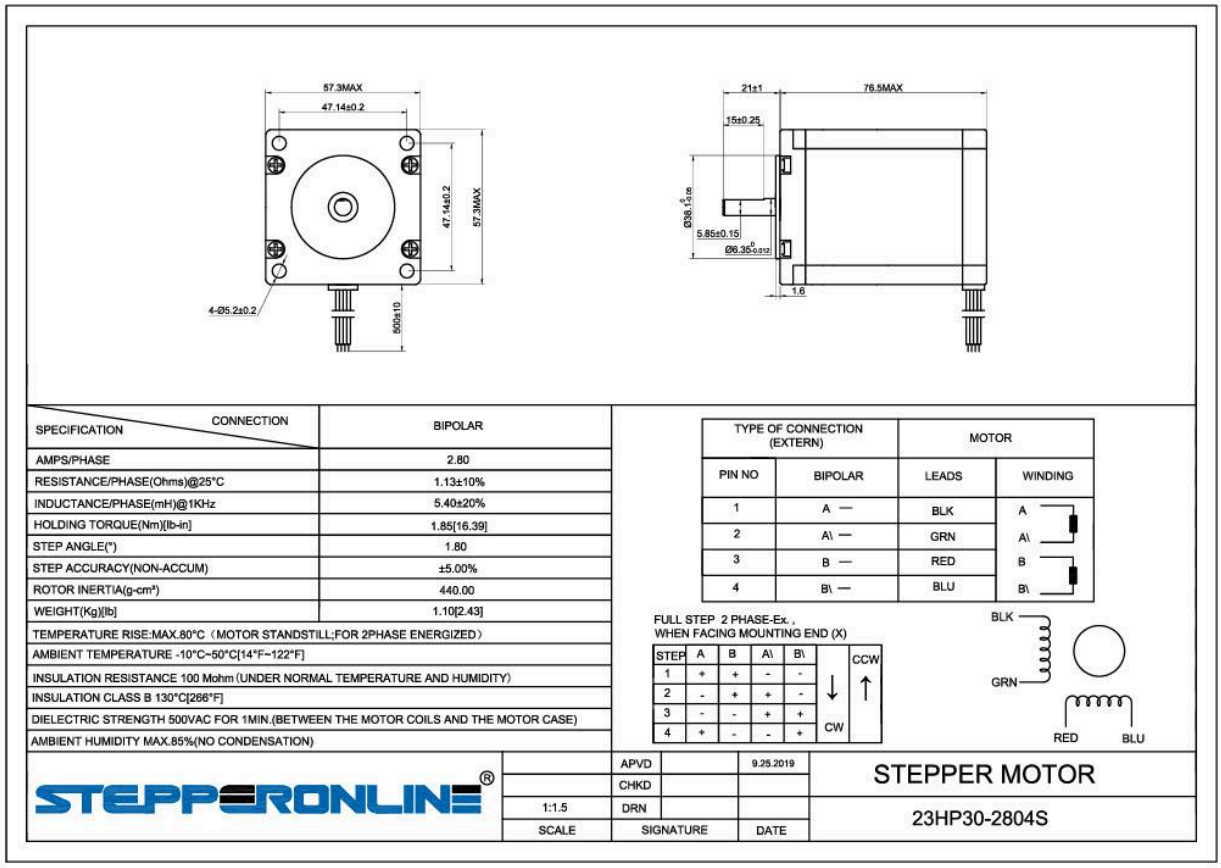


STEPPERONLINE®

	APVD		2.22.2023
	CHKD		
1.5:1	DRN		
SCALE	SIGNATURE		DATE

STEPPER MOTOR

11HS12-0704S



SPECIFICATION / CONNECTION	BIPOLAR
AMPS/PHASE	2.80
RESISTANCE/PHASE(Ohms)@25°C	1.13±10%
INDUCTANCE/PHASE(mH)@1KHz	5.40±20%
HOLDING TORQUE(Nm)[lb-in]	1.85[16.39]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	440.00
WEIGHT(Kg)[lb]	1.10[2.43]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL;FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

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

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

STEP	A	B	A\	B\
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+

CW ↓ CCW ↑

STEPPERONLINE®

	APVD		9.25.2019
	CHKD		
1:1.5	DRN		
SCALE	SIGNATURE		DATE

STEPPER MOTOR

23HP30-2804S

Appendix D: Motor Sizing and Specification Calculations

1. Pulses requires to move 25.4mm (1 Inch)

$$25.4\,mm \cdot \frac{1\,Rev}{2\,mm} \cdot \frac{1600\,Pulses}{1\,Rev} = 20,320\,Pulses$$

2. Linear Distance per pulse

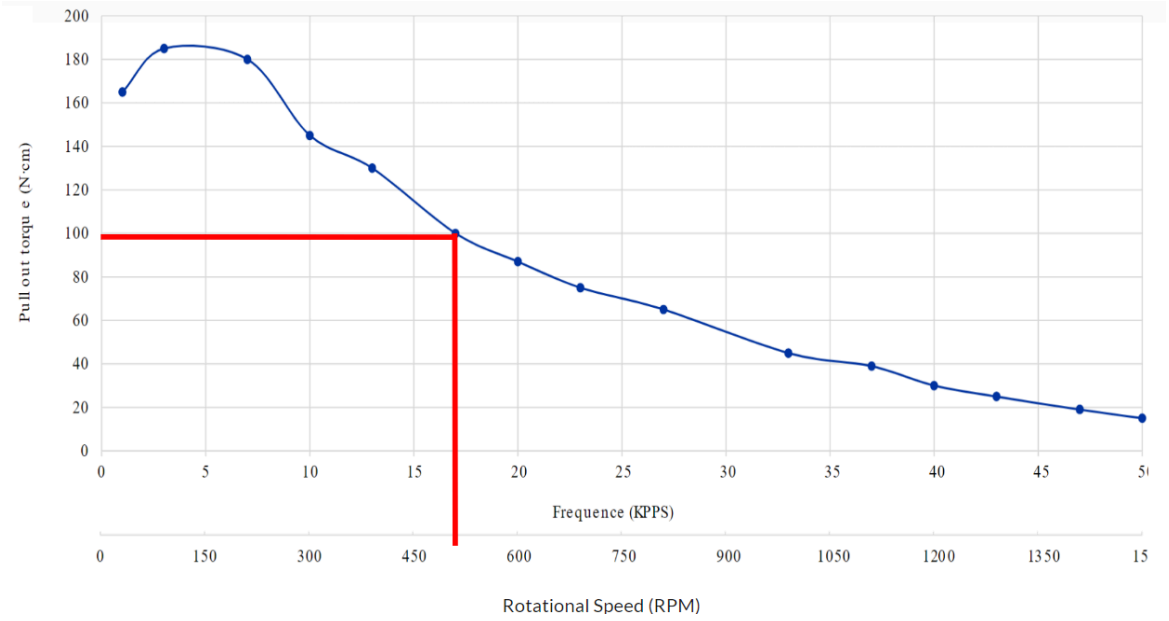
$$\frac{2\,mm}{Rev} \cdot \frac{1\,Rev}{1600\,Pulses} = \frac{1.25\,\mu m}{Pulse}$$

3. Motor Velocity

$$480\,RPM$$

4. Weight to Move: 4.053 Kg
5. Motor Torque Required

Input	
Force	4053 ○lb ○oz ◉g ○N
Pitch Diameter	8 ○in ◉mm
Thread density	.2 Threads per ○in ◉cm
Coefficient of Friction	0.25 (See table below)
Result Units	◉N*m ○N*cm ○lb*in ○oz*in
	Compute
Result	
Torque (Raise)	0.722 (Selected Units)
Torque (Lower)	0.561 (Selected Units)



Nema 23 Stepper motor and Torque Graph. 2024. OMC-Stepperonline

Appendix E: Main Arduino Script

MechanismCameraControl.ino

```
#include <AccelStepper.h>
#include <Pixy2.h>
// TB6600 stepper motor driver pins
int Stepper1Pulse = 22;       // Pulse or step pin for Stepper 1 (X-axis)
int Stepper1Direction = 23;    // Direction pin for Stepper 1 (X-axis)
int Stepper2Pulse = 26;       // Pulse or step pin for Stepper 2 (Y-axis)
int Stepper2Direction = 27;    // Direction pin for Stepper 2 (Y-axis)
int Stepper3Pulse = 30;       // Pulse or step pin for Stepper 3 (Z-axis)
int Stepper3Direction = 31;    // Direction pin for Stepper 3 (Z-axis)
// Rotary Encoder Pins
int SW_pin = 34;  // Switch pin
int DT_pin = 35;  // DT pin
int CLK_pin = 36;  // CLK pin
// Joystick Pins
int joystickX = A0; // Joystick X-axis
int joystickY = A1; // Joystick Y-axis
const int deadZone = 100; // Dead zone for joystick
const int joystickCenterValue = 512; // Center value for the joystick
// Limit Switches Pins
int limitSwitchX = 42; // Limit switch 1 for X-axis
int limitSwitchY = 44; // Limit switch 2 for Y-axis
int limitSwitchZ = 46; // Limit switch 3 for Z-axis
bool systemLocked = false;  // Automatic Locking if a limit switch is triggered after homing
// Button and LED Pins
int button = 2;
int red = 3;
int yellow = 4;
int green = 5;
const int pulsesPerRevolution = 400;   // Number of pulses per revolution for the stepper motor
```

```
// Assuming 1mm movement requires 'stepsPerMillimeter' steps for each motor.
const int stepsPerRevolution = 200;
const int stepsPerMillimeter = 50;
AccelStepper step1(1, Stepper1Pulse, Stepper1Direction);
AccelStepper step2(1, Stepper2Pulse, Stepper2Direction);
AccelStepper step3(1, Stepper3Pulse, Stepper3Direction);
// Variables to track encoder state
int lastState;
long encoderPosition = 0;
// Pixy Camera
Pixy2 pixy;
int signatureNumber = 2;
float widthScaleFactor = 0.5957;
float heightScaleFactor = 0.625;
float torchRadius = 3.0;
void setup() {
  Serial.begin(115200); // Start serial communication
  pinMode(Stepper1Pulse, OUTPUT);
  pinMode(Stepper1Direction, OUTPUT);
  pinMode(Stepper2Pulse, OUTPUT);
  pinMode(Stepper2Direction, OUTPUT);
  pinMode(Stepper3Pulse, OUTPUT);
  pinMode(Stepper3Direction, OUTPUT);
  pinMode(SW_pin, INPUT_PULLUP);
  pinMode(CLK_pin, INPUT_PULLUP);
  pinMode(DT_pin, INPUT_PULLUP);
  pinMode(joystickX, INPUT);
  pinMode(joystickY, INPUT);
  pinMode(limitSwitchX, INPUT_PULLUP);
  pinMode(limitSwitchY, INPUT_PULLUP);
  pinMode(limitSwitchZ, INPUT_PULLUP);
  pinMode(button, INPUT_PULLUP);
```

```
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
  // Turn Green LED on, red and yellow off
  digitalWrite(red, LOW);
  digitalWrite(yellow, LOW);
  digitalWrite(green, HIGH);
  // Initialize encoder variables
  lastState = digitalRead(DT_pin);
  // Pixy camera initialization, turn on 1 white led, set red led to 255 full brightness.
  pixy.init();
  pixy.setLamp(0,0);
  pixy.setLED(255,0,0);

  // Perform homing for X and Y axes
  homeStepper(step1, limitSwitchX,'X');
  homeStepper(step2, limitSwitchY,'Y');
  homeStepper(step3, limitSwitchZ,'Z');
  Serial.println("Homing Complete");
  digitalWrite(red, LOW);
  digitalWrite(yellow, LOW);
  digitalWrite(green, HIGH);
}
void homeStepper(AccelStepper &stepper, int limitSwitchPin, char axisName) {
  Serial.print("Homing ");
  Serial.print(axisName);
  Serial.println(" axis...");
  digitalWrite(red, LOW);
  digitalWrite(yellow, HIGH);
  digitalWrite(green, LOW);
  // Set a constant speed for homing
  step1.setMaxSpeed(1200);
```

```
  step1.setAcceleration(1000);
  step2.setMaxSpeed(1200);
  step2.setAcceleration(1000);
  step3.setMaxSpeed(1200);
  step3.setAcceleration(800);
  // Move towards the limit switch at a constant speed
  while (digitalRead(limitSwitchPin) == HIGH) {
    stepper.move(100); // Move 100 pulses at a time towards the limit switch
    stepper.run();
  }
  stepper.stop(); // Stop immediately once the limit switch is triggered
  // Move away from the limit switch to back off slightly
  stepper.setCurrentPosition(0); // Reset the position to 0 temporarily
  stepper.moveTo(-200); // Move steps away from the limit switch
  while(stepper.distanceToGo() != 0) {
    stepper.run();
  }
  // Reset the position to 0 to establish this as the home position
  stepper.setCurrentPosition(0);
  Serial.print(axisName);
  Serial.println(" axis homing complete.");
}
void processCommand(String command) {
    command.toLowerCase();
    // Set motor speeds and accelerations
    step1.setMaxSpeed(1200);
    step1.setAcceleration(800);
    step2.setMaxSpeed(1200);
    step2.setAcceleration(800);
    step3.setMaxSpeed(1200);
    step3.setAcceleration(800);
    if (command.startsWith("t")) {
```

```
  int wIndex = command.indexOf('w');
  int hIndex = command.indexOf('h');
  if (wIndex != -1 && hIndex != -1) {
    int width = command.substring(wIndex + 1, hIndex).toInt();
    int height = command.substring(hIndex + 1).toInt();
    performToolpath(width, height); // Call the toolpath routine with the parsed dimensions
    return; // Skip the rest of the function after handling the toolpath command
  }
}
// Define travel limits in steps
const int xMinLimit = -300 * stepsPerMillimeter;
const int yMinLimit = -140 * stepsPerMillimeter;
const int zMinLimit = -85 * stepsPerMillimeter;
const int xMaxLimit = 0;
const int yMaxLimit = 0;
const int zMaxLimit = 0;
int xMove = 0, yMove = 0, zMove = 0;
bool xMoveValid = false, yMoveValid = false, zMoveValid = false;
// Parse movements for each axis from the command
int xPos = command.indexOf('x');
if (xPos != -1) {
   xMove = step1.currentPosition() + command.substring(xPos + 2).toInt() * stepsPerMillimeter;
   xMoveValid = (xMove >= xMinLimit && xMove <= xMaxLimit);
}
int yPos = command.indexOf('y');
if (yPos != -1) {
   yMove = step2.currentPosition() + command.substring(yPos + 2).toInt() * stepsPerMillimeter;
   yMoveValid = (yMove >= yMinLimit && yMove <= yMaxLimit);
}
int zPos = command.indexOf('z');
if (zPos != -1) {
   zMove = step3.currentPosition() + command.substring(zPos + 2).toInt() * stepsPerMillimeter;
```

```
        zMoveValid = (zMove >= zMinLimit && zMove <= zMaxLimit);
    }
    // Move X Axis if valid
    if (xMoveValid) {
        step1.moveTo(xMove);
        while (step1.distanceToGo() != 0) {
            step1.run();
        }
        Serial.println("X axis movement complete.");
    }
    // Move Y Axis if valid
    if (yMoveValid) {
        step2.moveTo(yMove);
        while (step2.distanceToGo() != 0) {
            step2.run();
        }
        Serial.println("Y axis movement complete.");
    }
    // Move Z Axis if valid
    if (zMoveValid) {
        step3.moveTo(zMove);
        while (step3.distanceToGo() != 0) {
            step3.run();
        }
        Serial.println("Z axis movement complete.");
    }
    // If any movement was invalid, print an error message
    if (!xMoveValid || !yMoveValid || !zMoveValid) {
        Serial.println("Command exceeds travel limits, enter new command.");
    }
}
bool readLimitSwitchWithDebounce(int pin) {
```

```
  static unsigned long lastDebounceTime[3] = {0, 0, 0};  // Last time the pins were toggled
  static int lastState[3] = {HIGH, HIGH, HIGH};  // the last reading from the input pins
  static int currentState[3] = {HIGH, HIGH, HIGH};  // the current reading from the input pins
  const long debounceDelay = 50;  // The debounce time in milliseconds
  int index = pin - 42;  // Example to map pin number to index: adjust as per your pin numbers
  int reading = digitalRead(pin);
  if (reading != lastState[index]) {
    lastDebounceTime[index] = millis();
    lastState[index] = reading;
  }
  if ((millis() - lastDebounceTime[index]) > debounceDelay) {
    if (reading != currentState[index]) {
      currentState[index] = reading;
      if (currentState[index] == LOW) {
        return true;  // Return true immediately when the limit switch is triggered
      }
    }
  }
  return false;
}
void loop() {
 int currentState = digitalRead(DT_pin);
 int joystickXValue = joystickCenterValue - analogRead(joystickX);
 int joystickYValue = joystickCenterValue - analogRead(joystickY);
 static unsigned long lastPrintTime = 0;
 const unsigned long printInterval = 2000;
 // Check limit switches with debounce
 bool xLimitTriggered = readLimitSwitchWithDebounce(limitSwitchX);
 bool yLimitTriggered = readLimitSwitchWithDebounce(limitSwitchY);
 bool zLimitTriggered = readLimitSwitchWithDebounce(limitSwitchZ);
 // Immediate check for limit switch triggers
```

```
 if (readLimitSwitchWithDebounce(limitSwitchX) || readLimitSwitchWithDebounce(limitSwitchY) ||
readLimitSwitchWithDebounce(limitSwitchZ)) {
      if (!systemLocked) {
         digitalWrite(red, HIGH);
         digitalWrite(yellow, LOW);
         digitalWrite(green, LOW);
         Serial.println("Axis crashed");
         step1.stop();
         step2.stop();
         step3.stop();
         systemLocked = true;  // Lock the system
         return;  // Exit the loop to handle the crash
      }
 }
 if (systemLocked) {
   // If system is locked, only check for homing command
   if (Serial.available() > 0) {
     String input = Serial.readStringUntil('\n');
     input.trim();
     if (input.equalsIgnoreCase("H")) {
       systemLocked = false; // Unlock the system
       // Perform homing for all axes
       homeStepper(step1, limitSwitchX, 'X');
       homeStepper(step2, limitSwitchY, 'Y');
       homeStepper(step3, limitSwitchZ, 'Z');
       digitalWrite(red, LOW);
       digitalWrite(yellow, LOW);
       digitalWrite(green, HIGH);
       Serial.println("Homing Complete");
     }
   }
   return; // Skip the rest of the loop code if locked
```

```
}
if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n'); // Read the input until newline
    input.trim(); // Remove any whitespace
    if (input.equalsIgnoreCase("H")) {
      // Perform homing for all axes
      homeStepper(step1, limitSwitchX, 'X');
      homeStepper(step2, limitSwitchY, 'Y');
      homeStepper(step3, limitSwitchZ, 'Z');
      Serial.println("Homing Complete");
      digitalWrite(red, LOW);
      digitalWrite(yellow, LOW);
      digitalWrite(green, HIGH);
    } else {
      // Process custom movement commands
      processCommand(input);
    }
}
  // Continuously run stepper motors
step1.run();
step2.run();
step3.run();
if (millis() - lastPrintTime >= printInterval) {
  float xPos = step1.currentPosition() / (float)stepsPerMillimeter;
  float yPos = step2.currentPosition() / (float)stepsPerMillimeter;
  float zPos = step3.currentPosition() / (float)stepsPerMillimeter;
  // Grab the latest blocks from Pixy2. Color Connected Components
  pixy.ccc.getBlocks();
  if (pixy.ccc.numBlocks) {
    for (int i = 0; i < pixy.ccc.numBlocks; i++) {
      // Convert Pixy2 coordinates to machine coordinates
      if (pixy.ccc.blocks[i].m_signature == signatureNumber) {
```

```
        float cameraX = (pixy.ccc.blocks[i].m_x - 157.5) / 157.5 * 150;  // Scale factor: 150 mm for half-width of the camera's view
        float cameraY = (pixy.ccc.blocks[i].m_y - 103.5) / 103.5 * 85 + 50;  // Scale factor: 85 mm for half-height of the camera's view, +70 for the offset
        float boundaryWidthMM = pixy.ccc.blocks[i].m_width * widthScaleFactor;
        float boundaryHeightMM = pixy.ccc.blocks[i].m_height * heightScaleFactor;
        Serial.print("Wound Detected - X: ");
        Serial.print(cameraX);
        Serial.print(" mm, Y: ");
        Serial.print(cameraY);
        Serial.print(" mm, Boundary Dimensions - w: ");
        Serial.print(boundaryWidthMM);
        Serial.print(" mm, h: ");
        Serial.print(boundaryHeightMM);
        Serial.println(" mm");
      }
    }
  }
  int zleadscrewratio = 4.271;
  Serial.print("Machine Position - X: ");
  Serial.print(xPos);
  Serial.print(" mm, Y: ");
  Serial.print(yPos);
  Serial.print(" mm, Z: ");
  Serial.print(zPos/2.155);
  Serial.println(" mm");
  lastPrintTime = millis();
 }
 // Dynamic speed adjustment for joystick control
 if (abs(joystickXValue) > deadZone) {
  int speedX = map(abs(joystickXValue), deadZone, joystickCenterValue - deadZone, 0, 3000);
  step1.setMaxSpeed(speedX);
```

```
  step1.setAcceleration(2000);
  step1.move(joystickXValue > 0 ? -pulsesPerRevolution : pulsesPerRevolution);
 } else {
  step1.stop(); // Stop motor 1 if within dead zone
 }
 if (abs(joystickYValue) > deadZone) {
  int speedY = map(abs(joystickYValue), deadZone, joystickCenterValue - deadZone, 0, 3000);
  step2.setMaxSpeed(speedY);
  step2.setAcceleration(2000);
  step2.move(joystickYValue > 0 ? pulsesPerRevolution : -pulsesPerRevolution);
 } else {
  step2.stop(); // Stop motor 2 if within dead zone
 }
 // Encoder handling for stepper motor direction
 bool clockwise = digitalRead(CLK_pin) != currentState;
 if (currentState != lastState) {
  if (clockwise) {
   step3.setMaxSpeed(3000);
   step3.setAcceleration(1200); // Set a suitable speed for your setup
   step3.move(pulsesPerRevolution); // Move one step forward
  } else {
   step3.setMaxSpeed(3000);
   step3.setAcceleration(1200); // Set a suitable speed for your setup
   step3.move(-pulsesPerRevolution); // Move one step backward
  }
  lastState = currentState;
 }
 if (digitalRead(SW_pin) == LOW) {
  encoderPosition = 0; // Reset encoder position if switch is pressed
 }
}
void moveToPosition(float x, float y) {
```

```
  // Convert to steps and move
  step1.moveTo(x * stepsPerMillimeter);
  step2.moveTo(y * stepsPerMillimeter);
  // Wait for the movement to finish
  while (step1.distanceToGo() != 0 || step2.distanceToGo() != 0) {
   step1.run();
   step2.run();
  }
}
void performToolpath(float boundaryWidthMM, float boundaryHeightMM) {
 // Current position will be used as the starting point for the toolpath
 float currentX = step1.currentPosition() / (float)stepsPerMillimeter;
 float currentY = step2.currentPosition() / (float)stepsPerMillimeter;
 step1.setMaxSpeed(2000);
 step1.setAcceleration(1500);
 step2.setMaxSpeed(2000);
 step2.setAcceleration(1500);
 // Calculate the number of lines needed based on the torch diameter for overlap
 float lineSpacing = torchRadius ; // The distance between each line path
 int lines = boundaryWidthMM / lineSpacing;
 // Define the movement boundaries within the overall boundary dimensions
 float minX = currentX - boundaryWidthMM / 2;
 float maxX = currentX + boundaryWidthMM / 2;
 float minY = currentY - boundaryHeightMM / 2;
 float maxY = currentY + boundaryHeightMM / 2;
 // Starting conditions
 bool movingRight = true; // Direction flag
 // Perform the raster toolpath
 for (int i = 0; i <= lines; i++) {
  float lineX = minX + (i * lineSpacing);
  // Adjust the line position to stay within the boundary
  lineX = constrain(lineX, minX, maxX);
```

```
    // Move to the beginning of the next line
    moveToPosition(lineX, currentY);
    // Perform the line path
    if (movingRight) {
      // Move from top to bottom along Y
      moveToPosition(lineX, maxY);
      moveToPosition(lineX, minY);
    } else {
      // Move from bottom to top along Y
      moveToPosition(lineX, minY);
      moveToPosition(lineX, maxY);
    }
    // Change direction
    movingRight = !movingRight;
  }
  // Return to the starting position
  moveToPosition(currentX, currentY);
}
```