Spring 2021

Automated Control System for the Remote-Controlled Torch Positioner

Christopher Ferguson
caf113@zips.uakron.edu

Anna Abate
ama349@zips.uakron.edu

Kyle Cramer
knc45@zips.uakron.edu

Alex Sabitsch
ass47@zips.uakron.edu

Jacob Stefanko
jjs217@zips.uakron.edu

AUTOMATED CONTROL SYSTEM FOR THE REMOTE-CONTROLLED TORCH POSITIONER AND FLIR CAMERA MOUNT

Anna Abate

Kyle Cramer

Christopher Ferguson

Alex Sabitsch

Jacob Stefanko

Final Report for 4600:471 Senior/Honor Design, Spring 2021

Faculty Advisor: Dr. Gregory Morscher

Faculty/Honors Advisor: Dr. Donald Quinn

Faculty/Honors Reader 1: Dr. Ajay Mahajan

Faculty/Honors Reader 2: Dr. Manigandan Kannan

03, May, 2021

Project No. 03

Table of Contents

Abstract

This project is comprised of two separate sections to assist Dr. Morscher's lab in the Gas Turbine Testing Facility. The first is a continuation of a 2019-2020 Senior Design Project called the "Remote-Controlled Positioner" which was completed by Matthew Evans, Donald Morrison, and Joshua Verdier. The goal of this project is to complete the controller for the positioner and install the positioner so it can be utilized by Dr. Morscher's lab. The completion of this project relied on understanding Arduino software and hardware, and electrical components to build the controller.

The second section of this project was designing a camera mount for the FLIR thermal camera which the lab group uses during testing. The purpose of the mount is to replace a tripod which the lab group used prior to the construction of the camera mount. To successfully build the mount multiple considerations took place including material properties, location of the mount, and adjustability. Designing the mount involved following the appropriate design process of initial concept, conceptual design, embodiment design and final assembly.

The success of this project is determined by the satisfaction of the lab members to make testing simpler. Having an automated positioner allows more minute adjustments to be made during testing making a more accurate test. In addition, the camera mount replaces the tripod which is not secure and in the way of testing with a mount that is off to the side and extremely steady.

Background

The original goal of this project, per the project proposal, was to design and implement a controls system for Dr. Morscher's burner rig using LabView. The scope of this project changed due to the amount of work that remained on the original manual controller and with the delays in procuring a LabView capable computer. Work shifted to a two-part project with one being the creation of a FLIR camera mount for the burner rig and the second being the completion of the remote controller for the torch. The burner rig, which this project is built on, serves to simulate jet turbine conditions and test how materials, specifically ceramic matrix composites, react under high temperatures and stresses. This is a complicated system that requires a safe environment for the operators of the machine. The high-level components of the system are the torch, a sediment feeder, two FLIR cameras and an MTS tensile machine. One issue that the operators noticed was with the front FLIR camera and how it was mounted. It used a tripod that had a large footprint in a relatively small room, and it was easy to trip over due to its size. This was a problem because the camera required a stable platform to accurately record the temperature of the sample. A camera mount was built out of extruded aluminum to solve these problems. Extruded aluminum was chosen as the material as it is light, easy to assemble and non-corrosive. It is a modular system with adjustments in the x, y and z axes allowing for greater flexibility in the position of the camera.

The other problem that needed to be solved in this project is the remote controller to allow for precise movement of the torch. Previously, the torch was located on an extruded aluminum frame that was heavy and hard to adjust accurately. A group last year created a two axis CNC mount for the torch to move in two dimensions. This created a stable platform where the torch can be lit and adjusted safely and accurately. The mount has two stepper motors that turn ball screws for precise movement of the torch in a variety of speeds. A controller was necessary to move the torch using this CNC mount while also displaying the location of the torch. The controller was modified heavily from the previous group to create a reliable platform that the operators can safely use. This system allows for the operators to move the hypersonic flame in fractions of millimeters at a safe distance. The improvement in the setup allows for more accurate results while also increasing the distance the operators are from the flame. The controller represents a quality-of-life improvement that will aid in the research of high temperature composites.

Materials/Cost

The primary material usage was for the FLIR Camera Mount. It was entirely built from T-Slotted aluminum which is rather costly. It was the best option to meet the design requirements. The Automated Torch Positioner and Controller utilized the main components purchased from the previous group. Below is a breakdown of the cost of the FLIR Camera mount and the Automated Torch Positioner.

FLIR Camera Mount	$890.25
Automated Torch Positioner	$13.35
Total	**$903.60**

Component	Price	Quantity(Packs)	Total Cost	Part Number(McMaster-Carr)
T-Slotted Framing(1"x1"x6')	$17.68	1	$17.68	47065T101
T-Slotted Framing(1"x1"x2')	$11.77	6	$70.62	47065T107
T-Slotted Framing(2"x2"x3')	$27.86	1	$27.86	47065T501
T-Slotted Framing(Bearing)	$63.07	5	$315.35	47065T964
Ball Joint Linkage	$5.23	4	$20.92	6058K31
Corner Bracket(1"x2" Al)	$7.62	10	$76.20	47065T237
Stainless Hex Nut(1/4-20)	$2.06	1	$2.06	92673A113
T-Slotted Framing(Diagonal Brace)	$16.95	3	$50.85	47065T189
Nut(1/4-20)	$5.44	1	$5.44	47065T905
Corner Bracket(1"x1" Al)	$5.21	18	$93.78	47065T236
Bolt/Nut(1/4-20)	$1.85	22	$40.70	47065T139
Corner Bracket(2"x1" Al)	$5.25	2	$10.50	47065T239
Long Hex Bolt(1/4-20, 1-1/4)	$6.35	1	$6.35	92240A544
Flat Washer(.281" ID)	$7.28	1	$7.28	98029A027
Hex Nut(1/4-20 Al)	$6.43	2	$12.86	99149A121
Hex Nut(1/1-13 Al)	$9.22	1	$9.22	90670A033
Flat Washer(..531" ID)	$6.02	1	$6.02	92141A033
Hex Bolt(1/2-13x2" Al)	$3.36	2	$6.72	93306A720
Corner Bracket(2"x2" Al)	$7.91	4	$31.64	47065T253
Socket Head Cap Screw(1/4-20x5/8)	$8.79	1	$8.79	91251A539
Plastic Hand Break	$10.77	6	$64.62	60585K31
Hex Bolt(1/4-20x7/8)	$4.79	1	$4.79	92240A541
Resistor Kit(1/4W (500 Total)	$7.95	1	$7.95	Resistor Kit - 1/4W (500 total) - COM-10969 - SparkFun Electronics
Jumper Wires - Connected 6" (M/M, 20 pack)	$1.95	2	$3.90	Jumper Wires - Connected 6" (M/M, 20 pack) - PRT-12795 - SparkFun Electronics
Insulated Wire Ferrules Connector - Red - 16AWG - Pack of 100	$1.50	1	$1.50	Terminal Block Insulated Wire Ferrules Connector - Red - 16AWG - Pack of 100 | eBay

FLIR Camera Mount

The FLIR Camera Mount was a concept that Dr. Morscher proposed for our team to design. The purpose is to design a structure to hold a FLIR thermal camera at a specific location to record the temperature of the test specimen. Prior to the installation of our team's mount, the lab crew used a tripod for the camera. The issue is that the tripod takes a significant amount of space in the already congested testing area. The tripod can be seen below in **Figure 1**. To acquire an accurate reading of the test specimen, the FLIR camera is set at a specific angle. This was initially done using a six-foot tripod which took up a significant amount of space. Due to the spatial limitations of the lab, the physical structure of the mount must be compact and not impede the technician trying to work in the testing area.

Figure 1: Tripod in Testing Location

Preliminary Design

Prior to being able to model and design a mount for the FLIR camera, the team first needed to understand the need for a new camera mount design, as well as what the benefits would be in doing so. To begin, the team was able to witness the test rig being used and noticed how the existing camera tripod was taking up a significant amount of space within the testing room. Additionally, if an adjustment in the camera position needed to be made, the only way to do so was to manually pick up and move the tripod.

As previously stated, the first problem to address was the footprint that a camera mount would take up on the floor. Dr. Morscher and the lab team desired the mount be as integrated as possible with the rest of the testing rig. The initial process for the design was to first understand the location of the FLIR thermal camera. To determine the placement of the camera, the group went to the lab and measured the distance of the FLIR camera to the test specimen. This gave the team a reference for where the FLIR camera must be located once the mount is built. This location later changed after discussing the location of the camera in further detail with a member of the lab.

Initially, the camera mount team was not given many constraints in terms of drilling or in any way machining into the MTS test rig. As a result, some initial designs included drilling into the existing test rig, which is mostly made from steel. Since there were few specifications at the beginning of the project, the team came up with several ideas, most of which were described verbally to other team members. Designs like the one shown in **Figure 2** served as rough sketches of preliminary designs. Once it was determined that the mount was able to be adjustable without machining the existing MTS system, the camera mount team was able to move into conceptualizing the design using SolidWorks as the 3D modeling software.

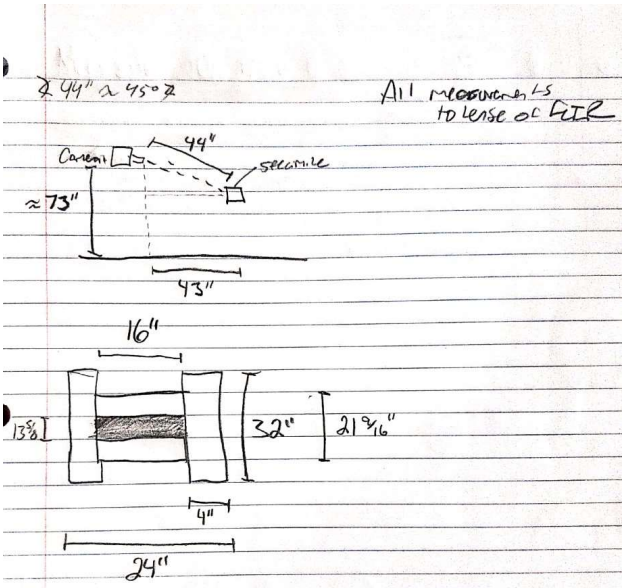

Figure 2: Initial Drawing of FLIR camera setup and MTS System base

Conceptual Design

As previously mentioned, the team primarily focused on using the computer aided design (CAD) software, SolidWorks for all conceptual and embodiment design phases. The two team members who worked on this part of the project were quite familiar with SolidWorks from previous internship experiences. Using those experiences, Anna and Alex created concepts of models that served the three primary functions; support the FLIR camera with adjustability, reduce the footprint of the camera mount, and resist rusting. Therefore, all designs that were modeled must conform to these key areas. In all the designs, corrosion resistant 80/20 aluminum t-slotted framing was utilized. The aluminum framing is relatively inexpensive, easily adjustable and compatible with linear bearings, and most importantly, corrosion resistant.

Another design requirement that was made clear by the lab members was needing to use non-corrosive materials. In the lab, the environment is very corrosive due to the humidity formed during testing. This prevented the use of steel due to its high corrosivity (rust) and is the reason why the team chose using aluminum. Aluminum is rust resistant which is demonstrated in the lab environment. In addition to not being corrosive, there is an aluminum product called 80/20 which is slotted aluminum allowing for brackets, connections, bearings, etc. to be used. The current mount for the torch is made of aluminum 80/20 and has no rust in comparison to the torch positioner frame which has already rusted in approximately one year. This can be seen in **Figure 3.**

Figure 3: Rust is clearly visible on steel

The team initially came up with two concepts. The key differentiation between each concept was the way each mounted to the ground/base. The team's first idea was to weld the base of the camera mount to the base of the MTS structure. However, Dr. Morscher and the lab group determined that it was not desirable to weld to the permanent structure. In addition to affecting the MTS framing, this concept was also found to be undesirable because it limited the amount of adjustability. Although grinding can be done to remove a weld between two materials, in this case other options presented better solutions. The second concept involved no machining or welding, but instead utilized the aluminum t-slotted framing, formed around the base of the MTS system, and clamped at designated contact points.

An important aspect of the conceptual design phase is to receive feedback from the project sponsor to make changes and improve the design. After several meetings with Dr. Morscher and his team, it was determined that the second concept, using 80/20 for the entire structure (not including hardware, clamps, etc.) would be the best option. This design would meet the three criteria set by supporting the FLIR camera with adjustability, reducing the footprint of the camera mount, and resisting rust. **Figure 4** shown below is a demonstration of this designed concept.

Figure 4: The chosen concept for the FLIR camera mount.

Along with meeting the three criteria, this design would also allow for three axes of adjustability. Another benefit of using the aluminum t-slotted framing is that there are many attachments that can be configured to the 80/20 frame. One such attachment that the team made use of was linear bearings. These linear bearings allowed one side of the frame to be held fixed, while another was free two move along an axis. On the adjustable side of the bearings, adjustable handles made it simple to loosen and tighten the framing for easy adjustability. By using three bearings, the mount was capable of being adjusted in all three axes. The linear bearings were able to be modeled into the CAD concept using SolidWorks and McMaster-Carr's downloadable part files. An example of one of the linear bearings that was used is shown in **Figure 5**. Once the specific design was chosen, the next step was to move into the embodiment design phase.

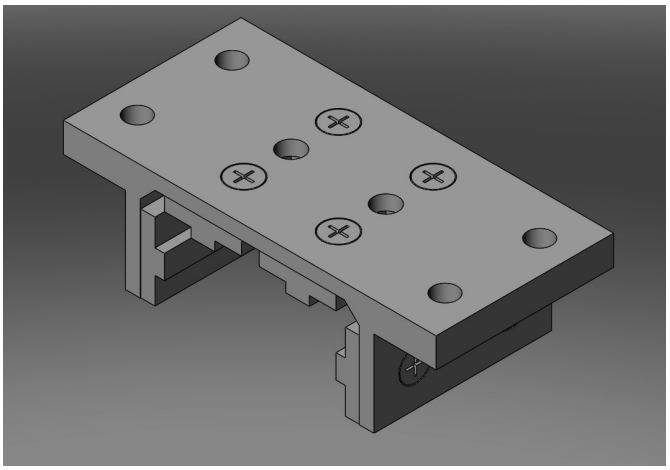

Figure 5: A linear bearing like what was used in the design.

Embodiment Design

Once the desired concept had been decided and agreed upon, the next steps were to create drawings, a bill of materials (BOM), order parts, complete any machining if necessary, and finally build the camera mount. Again, SolidWorks was utilized, except this time, for creating 2D drawings and a bill of materials. Since the senior design team would be the ones to build the final product, only an overall assembly with a BOM was necessary. Often, when work is being contracted out to other vendors, it is necessary to have detailed part and assembly drawings, but this was not necessary for this project. Instead, only one drawing was created and finetuned. A drawing like the one shown in **Figure 6** was created.

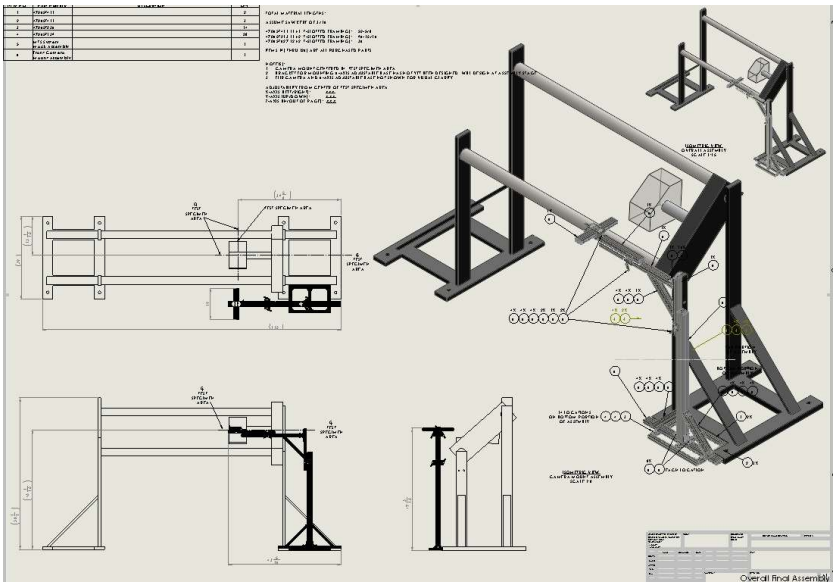

Figure 6: An example of a 2D drawing created.

The purpose of the drawing was not only to visualize the design and create a BOM, but a drawing also serves to be a guide when physically building the assembly. The balloon feature in SolidWorks is useful to concisely tell the builder what components go where and how many of each item goes to each location. Typically, the balloons are tethered to the BOM so all item numbers are unique and not repeated.

For the design the team chose, a combination of aluminum 80/20 t-slotted framing, 90° angle brackets, screws, gussets, and linear bearings were utilized to hold the structure together as well as make it fully adjustable. These items all had to be included within the drawing BOM, to serve as a guide of what to order and as instructions for building the camera mount. The team chose to simultaneously work on the drawing while still finalizing the design. Doing so allowed the team to notice possible issues on the 2D drawing that might go unnoticed in the 3D model, and vice versa.

The team went through several iterations of the model and had to adjust the drawing and BOM as necessary. Finalizing how the camera mount would mount to the ground was the key issue and what took the most time to decide. While it was determined during the conceptual design phase that a structure that was freestanding and clamped for support would work best, the team used the design embodiment phase to go through different options that would meet these criteria. One viable option was to build an 80/20 frame around the MTS base, then use sandbags or another heavy but removable item to keep the mount in place. Another option was to build the 80/20 frame around the MTS base, but to then have an extra piece of 80/20 spanning the length of the camera mount frame base. This design was found to not quite serve as a solid, but adjustable base so other options were considered. Finally, building the 80/20 frame base so that it would sit inside of the rectangular area of the MTS base was found to be the most desirable

option. Clamps are inexpensive and could be used to stabilize the camera mount to the MTS base if necessary.

Once the design and way of mounting to the ground was officially decided, the team needed to finalize the drawing and BOM. Often, the longer list of different materials on a BOM signifies increased cost for specialized items and machined parts. One important consideration in any engineering design is the cost. To reduce cost as much as feasibly possible, the team had at first hoped to reuse some of the 80/20 t-slotted framing that was already at the turbine building. A mistake that the team initially made was assuming the t-slotted framing that was the same size as what was available for free at the turbine building. It was not until the team was going to begin building the camera mount that it was discovered that the free 80/20 was 1”x1” but the design, drawing, and all hardware was modeled for 1.5”x1.5” framing. At this point, the camera mount team had to decide if it would be best to start over with the design to use the available framing, or if ordering new framing in the size that was designed was desirable and more cost effective. At this point, the team turned to Dr. Morscher to decide since financial cost was involved and it was determined that ordering new t-slotted framing would not affect the overall expense of the project significantly.

After gaining permission to order all parts, the team placed an order with McMaster-Carr for standard sizes of 80/20 framing, hardware, three linear bearings, and three custom cut angled 80/20 pieces to serve as gussets. Once the parts all arrived, it was time to begin building. The first step was to first confirm that the parts received were all correct sizes and quantities. An image of the team confirming part quantities and sizes can be seen in **Figure 7**. Once confirmed, the team then asked Brett Bell, an authorized member of the lab, if he would be willing to cut the standard sizes of 80/20 down to the correct lengths. Fortunately, Brett was able to cut all pieces the same day as parts were received, so significant time was saved by not going through a third-party machine shop.

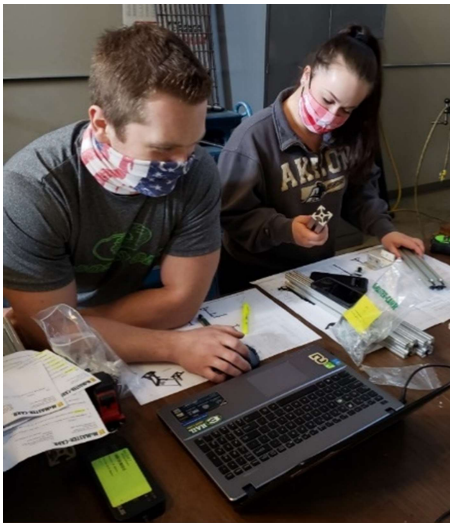

Figure 7: Alex and Anna beginning the assembly process

By following the BOM and assembly drawing, Anna and Alex were able to build most of the camera mount within a single day. **Figure 8** shows a progress picture of the camera mount being built. The following image, **Figure 9** displays the next stage in the building process, and one linear bearing had been installed. Since all of the parts ordered were designed to be used with 80/20, the team had relatively few issues installing and building the camera mount, but like any project, there was bound to be at least one issue or added requirement.

Figure 8: Progress picture during the first building day

Figure 9: Partially assembled Camera Mount. At this stage the mount was not stable, and more parts were required for stability

In the later stages of the embodiment design phase, project leaders and advisors asked if the team would be able to utilize a hole that goes through the MTS structure as a contact point to stabilize and hold the camera mount steady. A requirement of doing so was that the design must not permanently alter the MTS structure and the support piece must be removable. The team initially had planned on using a flat piece of steel that would be bolted on one side to the MTS hole and on the other side connected to the camera mount. However, using steel that had no corrosion resist properties was undesirable, so the team chose to use another piece of 80/20 to serve this purpose. A custom bracket needed to be modeled for this added piece, but the team was able to handle doing so without any issues.

Once the camera mount was built, the team analyzed the assembly to see if any improvements needed to be made. One area of improvement was that there was visible tilt where the linear bearings were located. In two locations the linear bearings had enough play in the connection that tilt was clear along two axes. The team decided that the best way to solve this issue was to order two more linear bearings and place them strategically to serve as additional supports. This can be seen in **Figure 10**, which is a side view of the completed camera mount design. **Figure 11** is another view of the completed camera mount.

Figure 10: Camera Mount Side View

Figure 11: Camera Mount Front View

The images shown **in Figures 12**, **13**, **14**, and **15** display the camera mount once it was installed in its final, but removable location. While the team did not have enough time to finalize the design for the mounting plate between the FLIR camera itself and the camera mount, members of this team will be continuing to work on this during graduate school to finalize the design for Dr. Morscher and his team. Once the camera is installed, it will be determined if c-clamps are necessary to hold the mount in place with the added weight on the t-slotted beam.

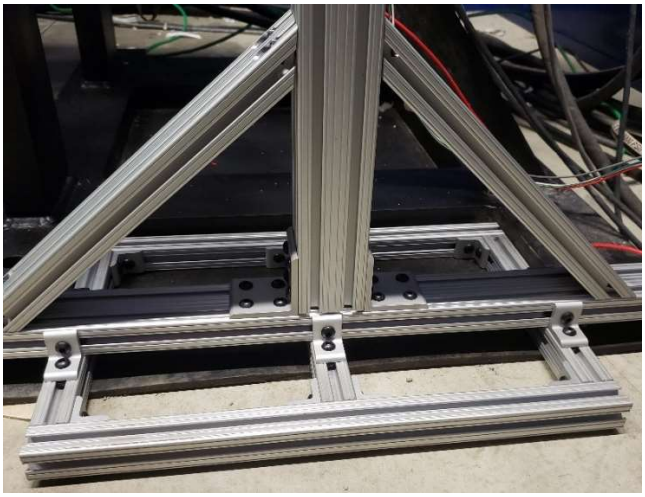

Figure 12: Camera Mount Base. Notice how the mount is stable and does not require mounting at the base. If mounting is wanted, the use of C-Clamps would work great.

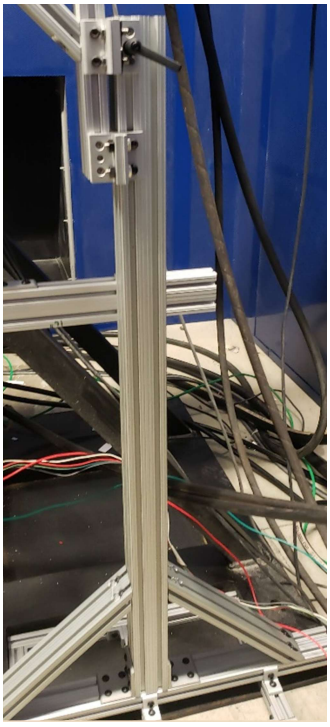

Figure 13: Vertical Component with gussets to provide support. Also, notice the linear bearings at the top of the vertical member to provide adjustability in the Y-Direction.

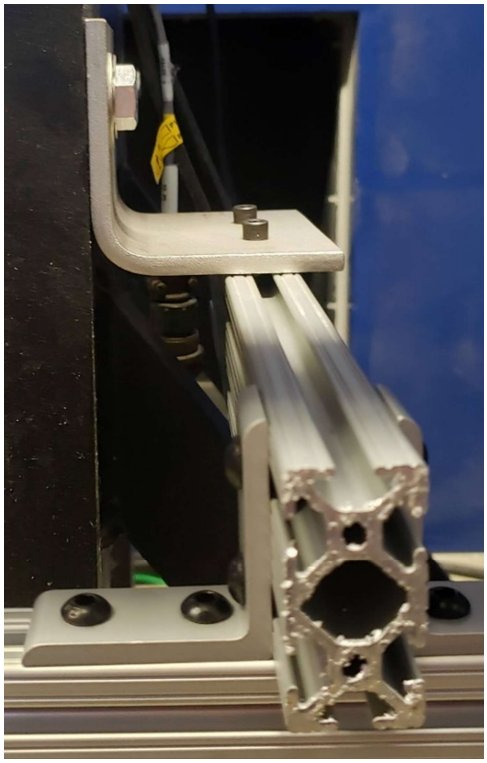

Figure 14: Mounting to MTS to provide additional support.

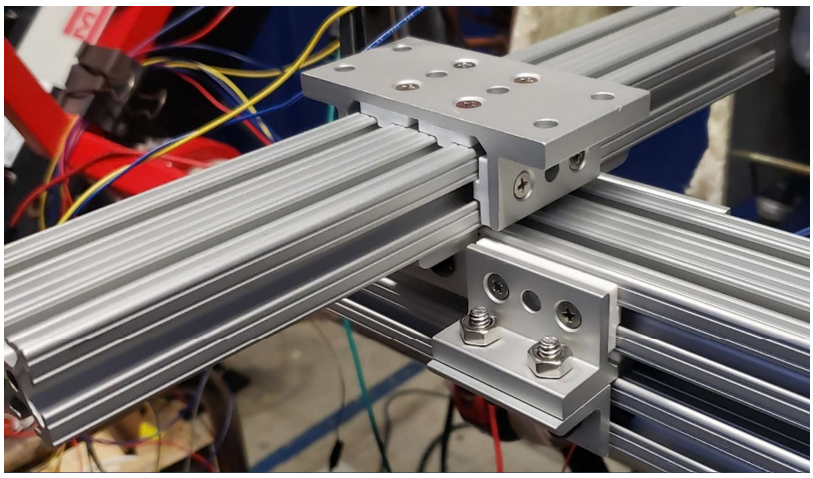

Figure 15: Image displaying the use of two linear bearings in two axes.

Lessons Learned

During all phases of the design process for the camera mount, the team learned about non-corrosive materials and their properties. The team also had the opportunity to improve on project management skills for a project that spanned two semesters. Most of the time working on the camera mount design was spent using SolidWorks to model the design. To have accurate models of the camera mount, the team first needed to complete on-site investigations and measurements of the MTS structure. Since having those measurements correct was critical to a functional camera mount, team members confirmed measurements by going back to the lab a second time to re-measure everything to confirm accuracy. As previously mentioned, a mistake that was made was that team members did not measure the existing and available 80/20 that was at the lab. While this mistake was able to be remedied, it certainly would have saved money by being able to utilize the available materials at the lab instead of ordering new parts.

While the team knew early on that aluminum 80/20 t-slotted framing was most desirable, other materials were investigated as well. Easily machinable and weldable A36 carbon steel was investigated as an option. The team learned that this material is a fraction of the cost of stainless steel and is easier to machine and weld. However, it is corrosive and would likely rust within months of installation. If this material would have been utilized, we would have had to investigate coatings to prevent corrosion. Some options included zinc plating, yellow chromate plating, or a rust-oleum brush on paint. However, 80/20 served as a better option for its corrosion resistance, in addition to compatible accessories for easy adjustment such as linear bearings and handles.

A major concern we initially had was the ability to weld the aluminum camera mount to the steel frame that is currently in place. After some research, the team realized it is not easy to weld the materials together and it requires special techniques that are outside of the University of Akron's machine shop capabilities. None of the team members had any background in welding

so alternative means were used to secure the aluminum camera frame to the steel frame that was already in place.

For adjustability, replaceability, and easy installation we designed the camera mount frame out of 80/20 T-Slotted extruded aluminum. This allowed the team to order mostly off the shelf parts, that had very little lead times, and required very little machining. There were concerns about corrosion between and around the aluminum and steel joint where the two frames would be bolted together. However, it was found that a bolted steel and aluminum joint will not have significant corrosion since the existing frame is not made from stainless steel. If the frame was made from stainless steel, localized corrosion could occur from the oxidation of aluminum during the anodizing process which makes it passive.

After realizing the possibility of corrosion between our stainless-steel bolts, extruded aluminum, and existing steel frame, aluminum replacement bolts were ordered to replace the stainless-steel bolts we initially planned to use. While waiting for the replacement hardware to arrive, a mock-up was made with the stainless-steel hardware to make sure the camera mount would be functional. For the mock-up, the team used 18-8 alloy stainless-steel bolts and nuts and tightened them to secure it in place. When we tried to remove the hardware, it was stuck, teaching us about thread galling the hard way. Galling occurs when sliding surfaces that are in contact with each other cause wear. The combination of friction between surfaces along with adhesion causes material to become stuck.

Overall, having the opportunity to complete the camera mount design for Dr. Morscher and his team was a great learning experience. The team became more familiar with project management, field measurements, SolidWorks, constructing a BOM, vendor relations and management, and hands-on building work. This portion of the project was a great opportunity to improve our engineering skills.

Remote Controlled Torch Positioner

A remote controller was created for the burner rig so graduate students could complete their research in a safe and efficient manner. A major problem with the previous method of torch adjustment is the crude nature of lifting and moving the entire base for large adjustments. Fine adjustments were done with a small dial that was attached to the top of the base which worked well but could be improved upon. The entire adjustment process was usable but not ideal for precise, reliable movement. A two-axis torch mount was created by the previous group to accomplish the hardware side of this project, but the controller aspect needed to be finished for full operation. The controller utilizes two Arduinos to control six buttons, two toggle switches, a potentiometer, two stepper motors and two I2C LCDs. The first Arduino (Nano) controls the stepper motors with the buttons on the face of the controller working as inputs to control them. The second Arduino (Uno) takes the step and direction signals (used for the stepper motor control) from the Nano and uses it to calculate position and speed. These calculations are then displayed on the two LCDs.

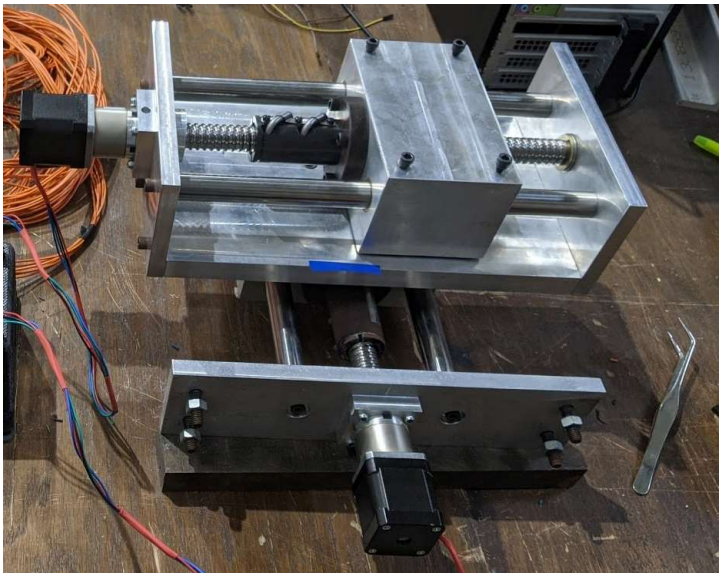

Figure 16: Torch Positioner to be controlled

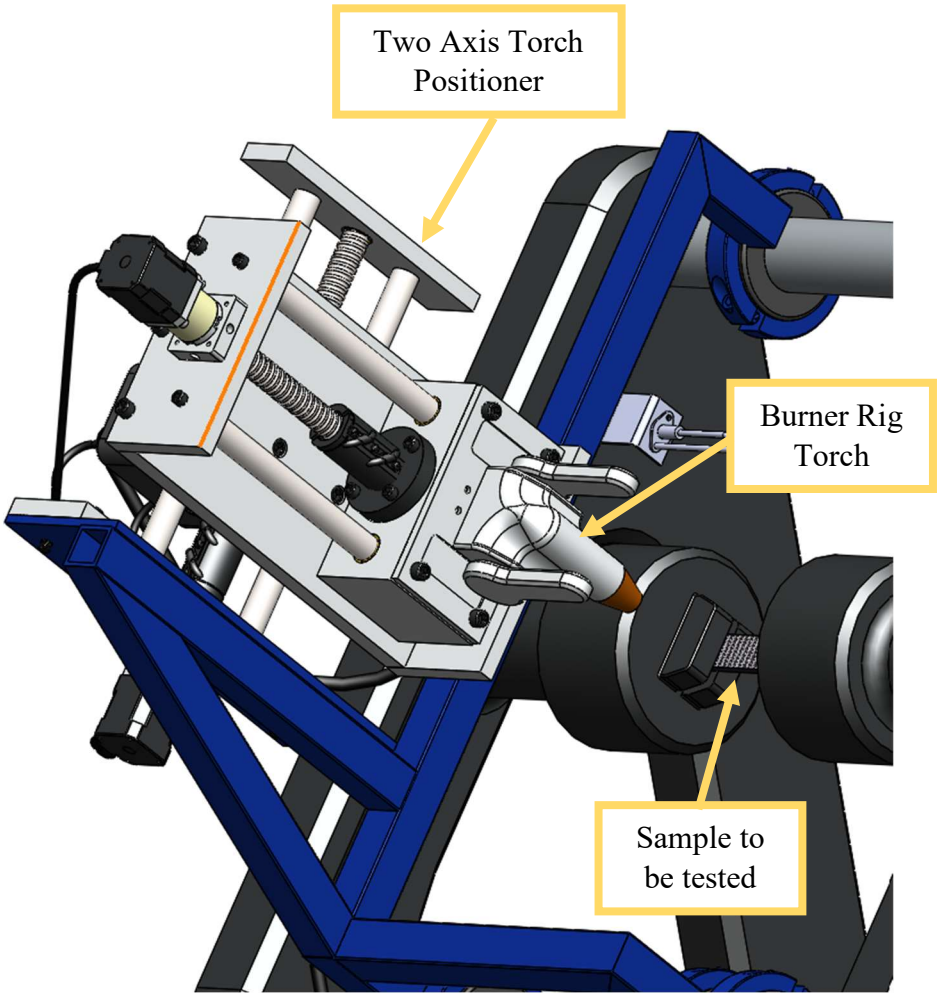

Figure 17: SolidWorks model of the Torch Positioner without the controller

Controller Hardware and Design

The controller was partially designed and created before we started, but it did not have in-depth documentation to explain all the design decisions that were made. Two PCB boards from the last group were partially wired up to the Arduino Nano and pushbuttons without an explanation on what each wire did. Our group is made up entirely of Mechanical Engineering students, so we did not have the experience to know what we were looking at initially. Due to the lack of knowledge in electronics and the lack of documentation, the project was slow going at first.

The first step for the project was to create our own documentation and reverse engineer the work that was already complete. This stage took approximately one month to determine what was done, why it was done and how we could continue the project. Overall, the why was the most difficult to understand but paid dividends as the project progressed and we became more confident working with electronics. Below in **Figure 18** is the Excel table created to map out the Arduino Nano's connections with the PCB, push buttons, potentiometer, motor control boards

and the Arduino Uno. As a basic description, an Arduino is a microcontroller that can accomplish small computational tasks. There are rows of digital and analog pins that can be connected to different devices like pushbuttons or potentiometers. Each pin on the board can be connected to a device to either power it, receive data, or output data. Individual pins can be operated using code from Arduino IDE which is a software created by Arduino and based off the C programming language.

Summary of Electronics

The first major device used in this controller is the Arduino Nano as it sends out the step and direction signals to the motor control boards. It controls the movement of the positioner above in **Figure 19** and is the brain of the controller.

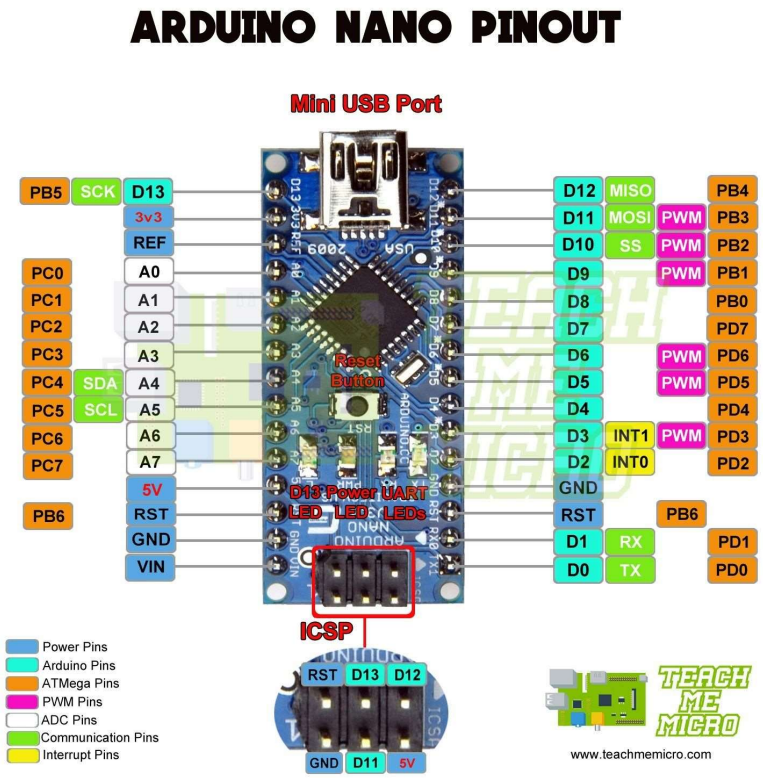

Figure 19: Arduino Nano pin layout

An Arduino Uno was added to the controller late in the process because of the inability for the single Nano to complete all the computations without a noticeable lag. This complicated the project due to splitting the workload between two devices. Here is the pin layout of an Arduino Uno below.

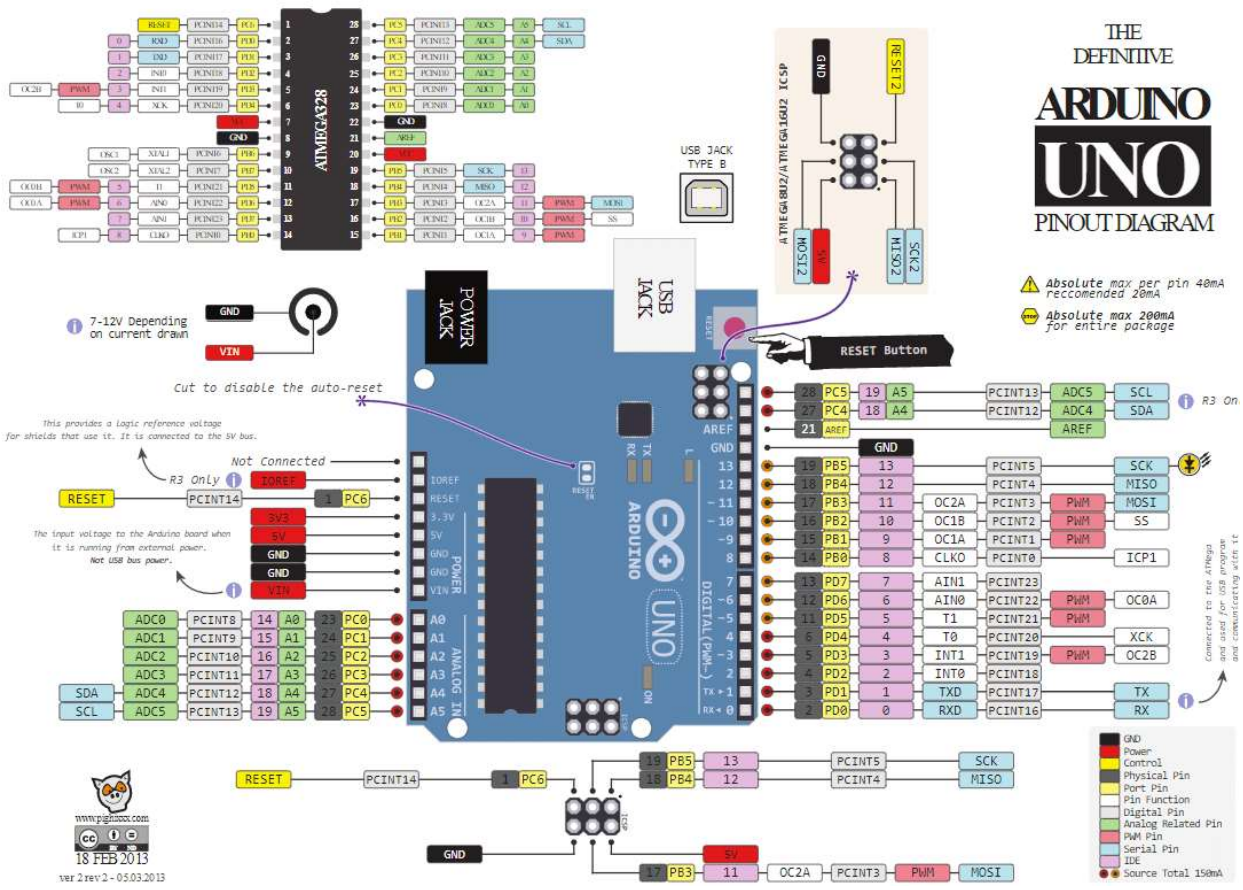

Figure 20: Arduino Uno Pin Layout

The two Arduinos described above connect to PCB boards for ease of assembly. A PCB board is used to simplify and miniaturize prototyping. Electronics are initially tested on prototyping board or breadboards which allow for easy disassembly and alteration during the design phase. PCB's act as built-in wiring paths so only the devices themselves are needed to be soldered onto the board and the board takes care of most of the wiring. The schematic for the controller PCB can be seen in below and this was used to document where each connection on the board leads to. This schematic was used as a launching pad in understanding the controller and how to modify it to our needs.

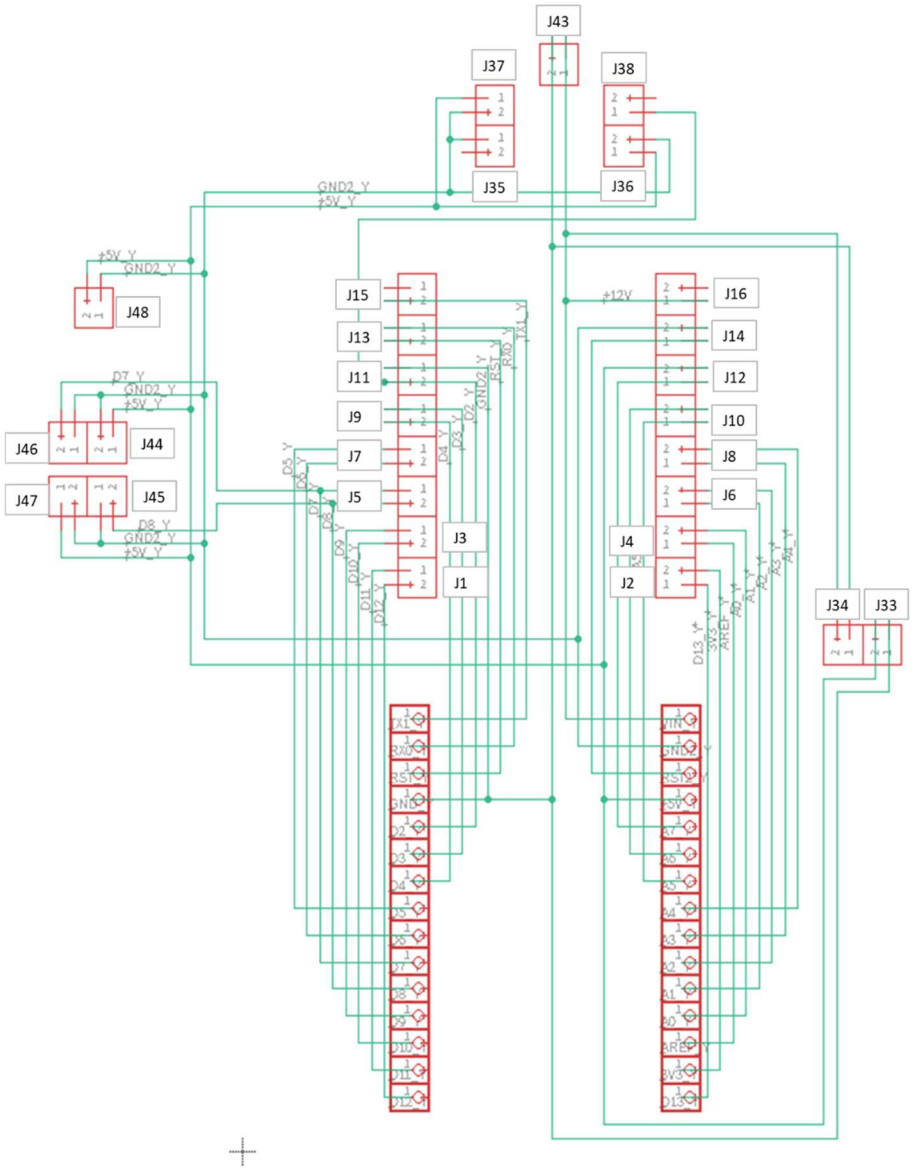

Figure 21: Schematic for the PCB used to connect the Arduino Nano to every other component (Designed by the previous group and updated for clarification)

Every junction defined in the above schematic connects to a pin on the Arduino Nano where a through hole is located on the PCB. Terminal blocks were soldered to these through holes as shown in **Figure 21**. Terminal blocks are parts that can be soldered to a PCB board while allowing for non-permanent attachment to wires. This allows for adjustment of the wiring without having to remove solder. The Arduino Nano, PCB board and terminal blocks can be seen in **Figure 22** below.

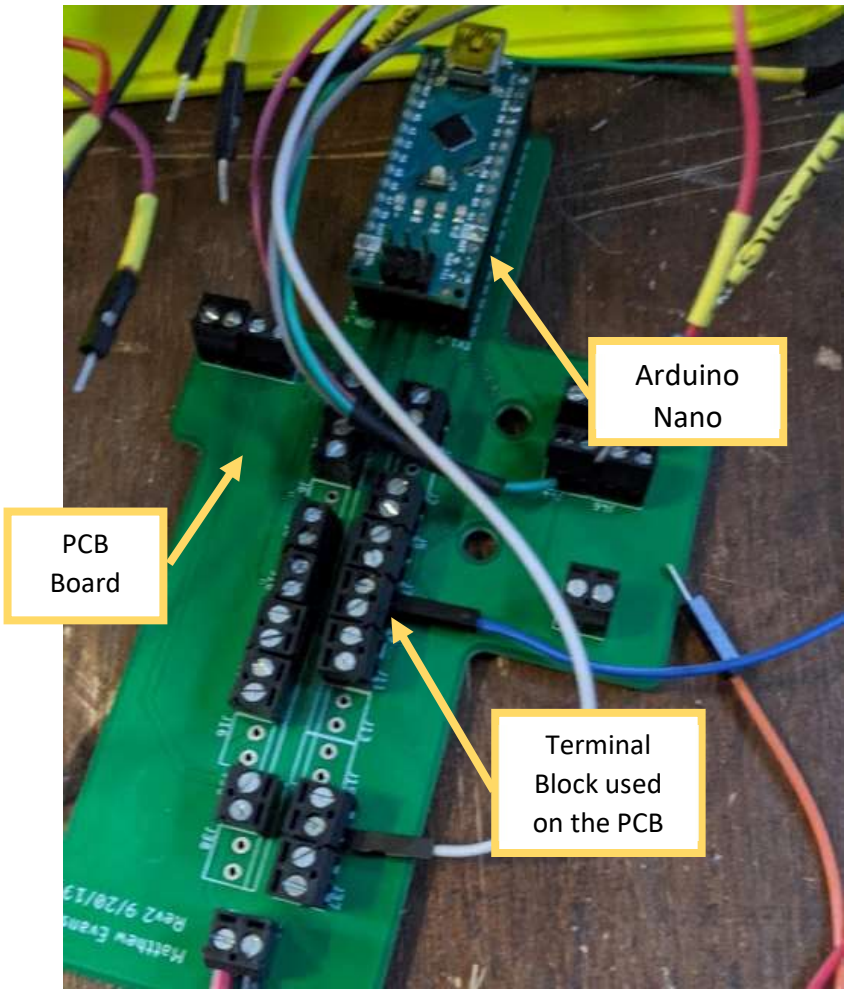

Figure 22: Initial soldered PCB with mounted Nano

A second PCB board was made for the twin motor control boards used in this project. The motor control board that was chosen was the Big Easy Driver because of its robust documentation and overall affordability. Each board controls one stepper motor and are mounted in the breakout box. The breakout box will be discussed later, but this is an enclosure that is separate from the controller and acts as a housing for the motor controllers. The Arduino Nano sends the Big Easy Drivers signals on how many steps and direction it should rotate. These step and direction signals are translated into actions by the Big Easy Driver turning on and off different motor coils.

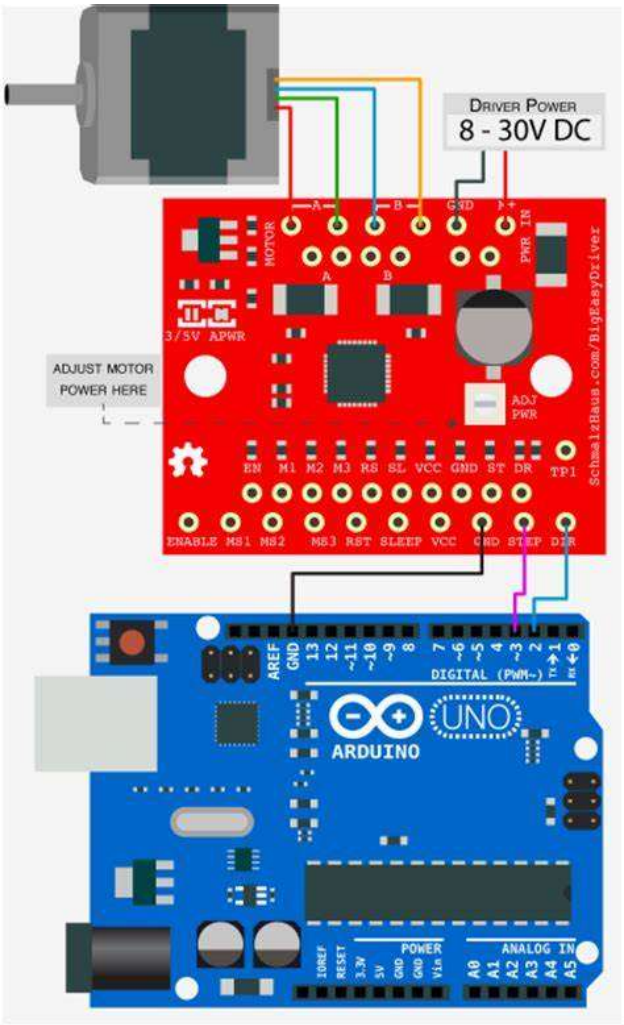

Figure 23: Big Easy Drive acting as an intermediate for the Arduino and stepper motor

Two of these motor control boards were attached to the PCB board below. Each pin on the board corresponds to a different task and the PCB board creates an easy-to-use platform to solder on components.

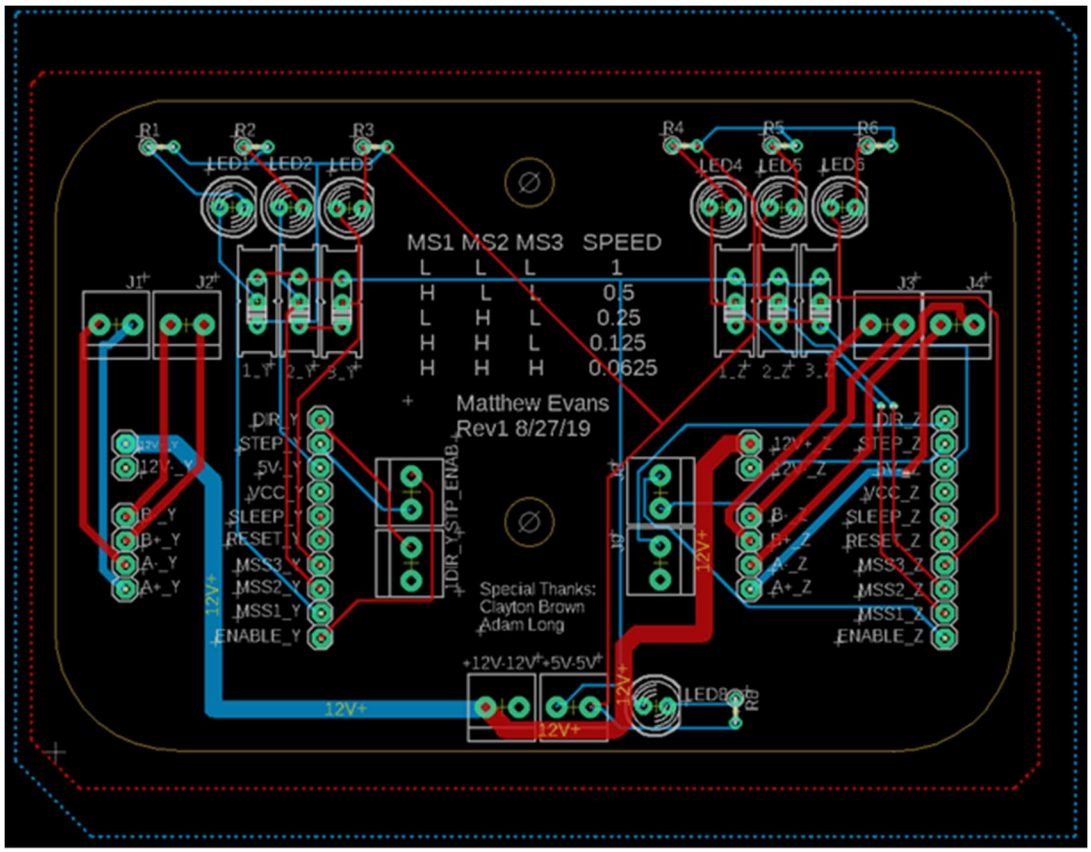

Figure 24: Schematic of the Motor Control PCB (Designed by the previous group)

Documentation and Planning

Figure 18 has the entire list of connections that are made from the PCB board to the Arduino Nano. It was created by our group to better organize the internals of the controller. The PCB board is broken up into a series of junctions made up of two through holes each used to organize the pins of the Nano. The original wiring was recorded on this excel sheet and was extensively revised and built upon to create the final wiring schematic. This table was integral in understanding and designing the controller internals. One look at the table clarifies what each wire in the controller does and where it should be located. Coding the Arduino, which will be touched on later, is simplified when each Arduino pin is defined with its purpose. This organization benefits both the hardware and software side of the controller.

Color Table	
Digital Pin	Power
Analog Pin	GND

Junction	Pin Num	Connecting Pin	Description	Hardware Connection
J1	1	D11	Arduino: Digital pin	Y Stepper Motor - Direction
	2	D12	Arduino: Digital pin	Y Stepper Motor - Step
J2	1	D13	Arduino: Digital pin	Y Toggle Switch
	2	3.3V	Arduino: Power output for 3.3V	
J3	1	D9	Arduino: Digital pin	Z+ Push Button
	2	D10	Arduino: Digital pin	Z- Push Button
J4	1	REF	Arduino: Reference pin for analog voltage comparison	
	2	A0	Arduino: Analog pin for potentiometer	Potentiometer
J5	1	D7	Arduino: Digital pin	
	2	D8	Arduino: Digital pin	
J6	1	A1	Arduino: Analog pin	Y Step to Uno
	2	A2	Arduino: Analog pin	Z Step to Uno
J7	1	D5	Arduino: Digital pin	Z Stepper Motor - Direction
	2	D6	Arduino: Digital pin	Z Stepper Motor - Step
J8	1	A3	Arduino: Analog pin	Y Direction to Uno
	2	A4	Arduino: Analog pin	
J9	1	D3	Arduino: Digital pin	
	2	D4	Arduino: Digital pin	
J10	1	A5	Arduino: Analog pin	
	2	A6	Arduino: Analog pin	
J11	1	GND - D	Arduino: Ground digital side	Y Toggle Switch, Z Toggle Switch
	2	D2	Arduino: Digital pin	Z Toggle Switch
J12	1	A7	Arduino: Analog pin	Z Direction to Uno
	2	5V	Arduino: Power output for 5V	Y Toggle Switch, Z Toggle Switch
J13	1	RX	Arduino: Serial RX pin (D0)	
	2	RESET1	Arduino: Reset pin	
J14	1	RESET2	Arduino: Reset pin	
	2	GND - A	Arduino: Ground analog side	Z- Push Button, Z+ Push Button
J15	1	EMPTY	EMPTY	EMPTY
	2	TX	Arduino: Serial RX pin (D1)	
J16	1	VIN	Arduino: Power input - 12V	VIN on Uno
	2	EMPTY	EMPTY	EMPTY
J33	1	GND - D	Arduino: Ground digital side	Potentiometer
	2	5V	Arduino: Power output for 5V	Potentiometer
J34	1	GND - D	Arduino: Ground digital side	GND to BED
	2	VIN	Arduino: Power input - 12V	12V Power to BED
J35	1	GND - A	Arduino: Ground analog side	Y Zero with LED ring
	2	EMPTY	EMPTY	EMPTY
J36	1	5V	Arduino: Power output for 5V	Y+ Push Button, Y- Push Button, Z- Push Button, Z- Push Button, Z+ Push Button, Z+ Push Button
	2	GND - A	Arduino: Ground analog side	Z Zero with LED ring
J37	1	5V	Arduino: Power output for 5V	Motor Control Board
	2	GND - A	Arduino: Ground analog side	Motor Control Board
J38	1	D2	Arduino: Digital pin	
	2	EMPTY	EMPTY	EMPTY
J43	1	VIN	Arduino: Power input - 12V	Connects to 12V power supply
	2	GND - D	Arduino: Ground digital side	Connects to 12V ground
J44	1	5V	Arduino: Power output for 5V	Y+ Push Button
	2	GND - A	Arduino: Ground analog side	Y+ Push Button
J45	1	GND - A	Arduino: Ground analog side	Y- Push Button
	2	D8	Arduino: Digital pin	Y- Push Button
J46	1	GND - A	Arduino: Ground analog side	Ground to Uno
	2	D7	Arduino: Digital pin	Y+ Push Button
J47	1	5V	Arduino: Power output for 5V	Y- Push Button
	2	GND - A	Arduino: Ground analog side	LCD Bottom Panel
J48	1	5V	Arduino: Power output for 5V	Y Zero, Z Zero
	2	GND - A	Arduino: Ground analog side	LCD Top Panel

Figure 18: Documentation of each pin on the motor control Arduino

Figure 25 is the documentation we created for the Arduino Uno connections. Because the Uno only receives data from the Nano and then outputs that data to the LCDs, it is a much simpler system. The inputs for the Uno are the step and direction of each motor, the potentiometer, and the two buttons. The outputs are simply the two LCD displays.

The two blue brackets labeled with 2KΩ are the records on which pins need a resistor of resistance 2KΩ. This resistance was chosen as it is the recommended resistor used with the purchased push buttons. A push button requires three wires to complete the connection and they are: the power wire, the ground wire, and the digital signal wire. The digital signal wire must be connected to the ground through a resistor to "pull down" the signal thus stabilizing it. Without this resistor, the signal will fluctuate creating a button that no longer turns on when pressed.

Arduino Uno Connections			
Uno Pin	**Connecting Pin**	**Nano Junction**	**Description**
RX	-	-	-
TX	-	-	-
D2	-	-	-
D3	-	-	-
D4	-	-	-
D5	-	-	-
D6	-	-	Y Zero
D7	-	-	Z Zero
D8	-	-	-
D9	A3	J8(1)	Y Direction
D10	A7	J10(2)	Z Direction
D11	A1	J6(1)	Y Step
D12	A2	J6(2)	Z Step
D13	-	-	-
A0	-	-	Potentiometer
A1	-	-	-
A2	-	-	-
A3	-	-	-
A4	-	-	I2C Data Out, LCD Top and Bottom Panels, SDA
A5	-	-	I2C Clock, LCD Top and Bottom Panels, SCL
VIN	VIN on Nano	J16(1)	12V Power
GND	GND on Nano	J46(1)	Ground
GND	-	-	Y Zero
GND	-	-	Z Zero
5V	-	-	Top and Bottom LCDs, Y Zero, Z Zero
3.3V	-	-	-
Reset	-	-	-
IOREF	-	-	-
AREF	-	-	-

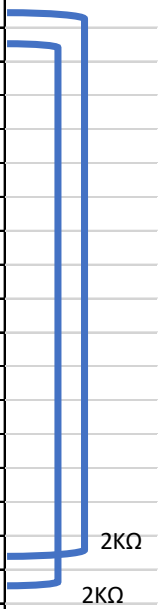

Figure 25: Documentation of Each Pin on the LCD Control Arduino

Construction of the Controller

After the initial setup and documentation steps, the parts of the controller already designed could be expanded and built off. One of the first steps was to recreate the controller on a prototyping breadboard. A breadboard allows for non-permanent construction of circuits allowing us to create designs that can be easily changed. After we had a working breadboard design, we took that and worked to improve the wiring paths. The original controller PCB was designed to have each component ground in a single terminal. This was not ideal and led to grounding issues throughout the initial tests. Because of these issues, we took a new controller PCB and reassigned where each component connected to. This allowed for the component grounds to connect to a set of thirteen different grounding pins which was much more stable for us during our preliminary tests.

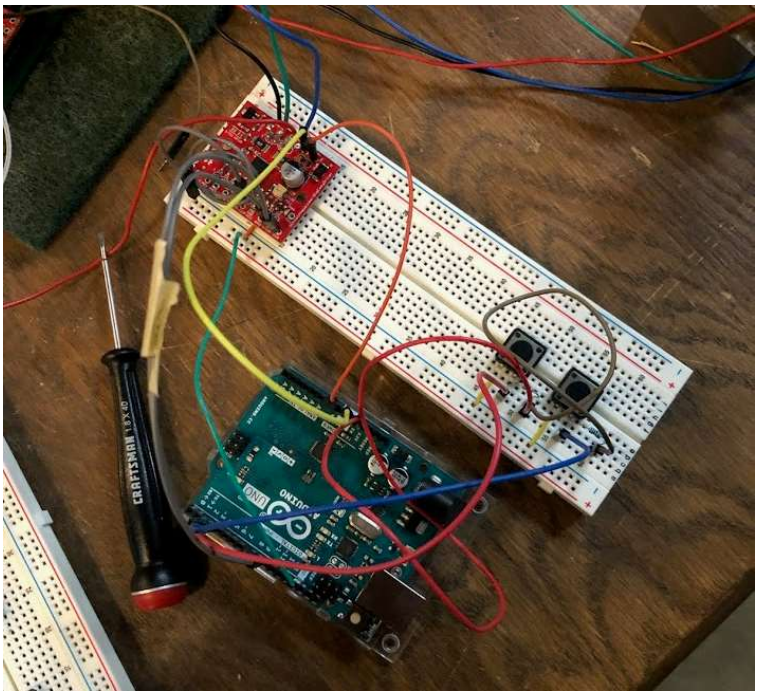

Figure 26: Breadboard controller

Due to the redistribution of the wiring in the PCB, each terminal block, header pin and resistor had to be resoldered. Soldering is a skill no one in the group possessed so learning to solder small electronics proved challenging at first. Two separate PCBs had to be soldered due to connection issues that was eventually solved by rearranging the ground and power pins to include fewer components per pin. One major issue with the PCB was that it did not include through holes for resistors on the main face. Since it was designed without dedicated resistor locations, the only way to add resistors to the push buttons was to solder resistors to the bottom of the PCB on the joints. The soldered resistors are seen in **Figure 27**, and even though this

method was not ideal, it worked for the purposes of this controller. We were too far in the design phase to be able to redesign the PCB leading to these work arounds.

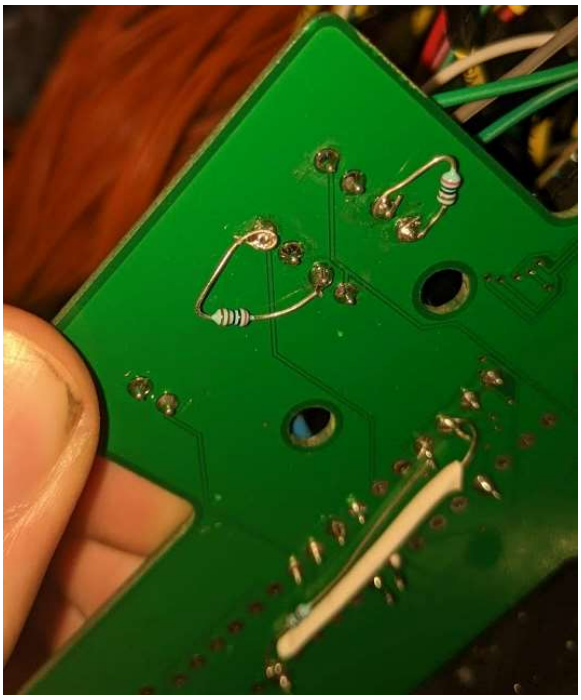

Figure 27: 2KΩ Resistors soldered to the underside of the PCB

Each component on the face of the controller lid had to be connected to the PCB. Since the wiring was redesigned, the Excel table created above was used extensively to organize the hardware. There are six pushbuttons with four wires each. One wire is for the digital signal on the button's state, two for 5-volt power and the other for the ground. There is one potentiometer to control the speed with three wires. One for power, one for ground and one that outputs the signal. Next is the two toggle switches which enable the motor. They have three wires each with one ground, one 5-volt power, and one digital signal. The last devices that need to attach to the PCB are the two LCDs. The LCDs use I2C protocol meaning it only has four wires to connect to the PCB. One is the power; one is ground, and the other two are SCL and SDA. The only wires that connect to the controller PCB are the power and ground as the two data pins connect directly to the Arduino Uno. All components are labeled on **Figure 28.**

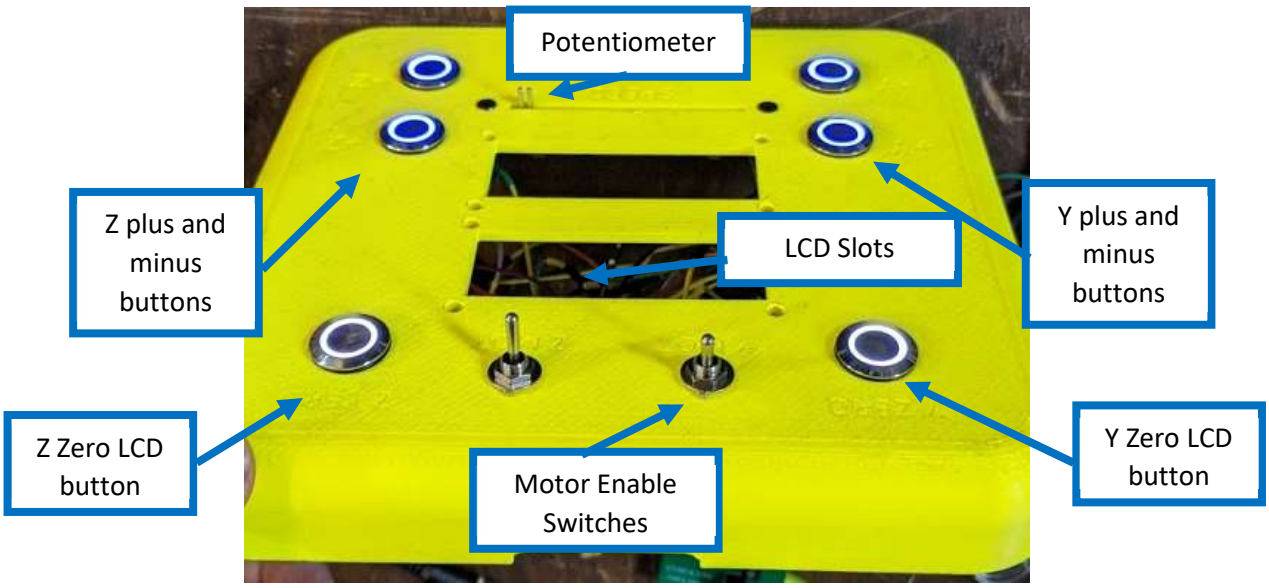

Figure 28: Controller face and input components

All components are wired up to the two Arduinos below in **Figure 29**. The wires that signal the motors to move are routed out the bottom through the Deutsch connector. The orange pair of wires carry 12V power along with the ground while the blue and green send the step and direction signals to the motor controllers. All six of these wires are necessary for the operation of the torch positioner and allow for the decisions that the Arduino make to be communicated to the motors.

Each of the components were wired using jumper wires. Jumper wires allow for two wires to connect easily for prototyping. One problem we had was that the jumper wires would separate since the connection was not fully secure. The solution to this problem was to use heat shrink over each connection. After heat was applied the heat shrink provided a secure connection that prevented movement in the wires. This step solved the issue of wires disconnecting and led to a more stable controller. The heat shrink is pointed out in the figure below.

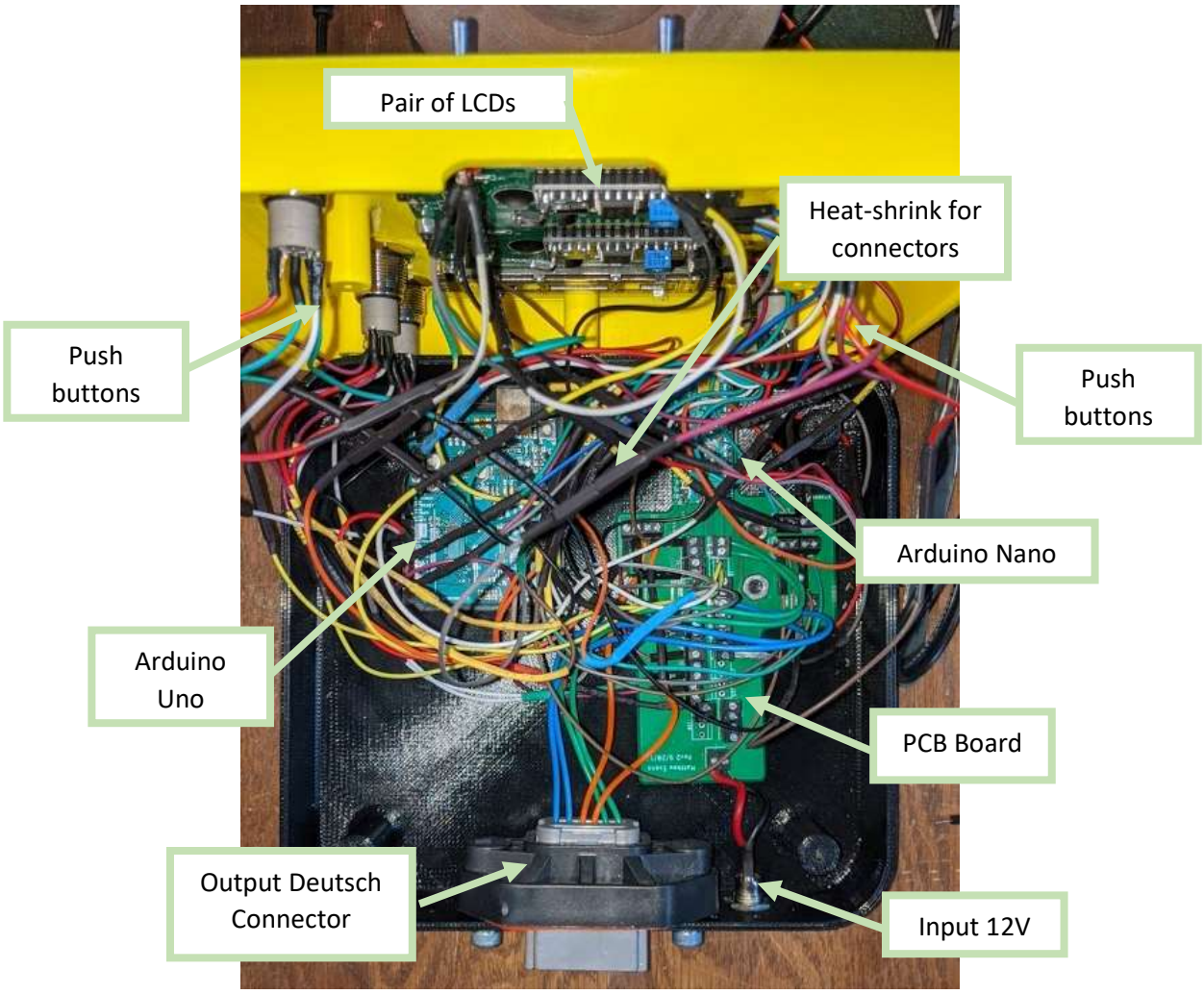

Figure 29: Wired final controller

Breakout Box Design

The breakout box is an additional component that mounts to the torch positioner frame. The function of the breakout box is to encapsulate the stepper motor control boards. The control boards must be kept cool and contain a filter to remove debris which can damage the control boards. It is important to note that the Breakout Box is located close to the Stepper motors because this increases the efficiency of the system (Evans et.al., 2020).

The design of this component was initiated by the previous group who designed the initial breakout box design which can be seen below in **Figure 30**. This was a great initial design, however cracked and was no longer usable.

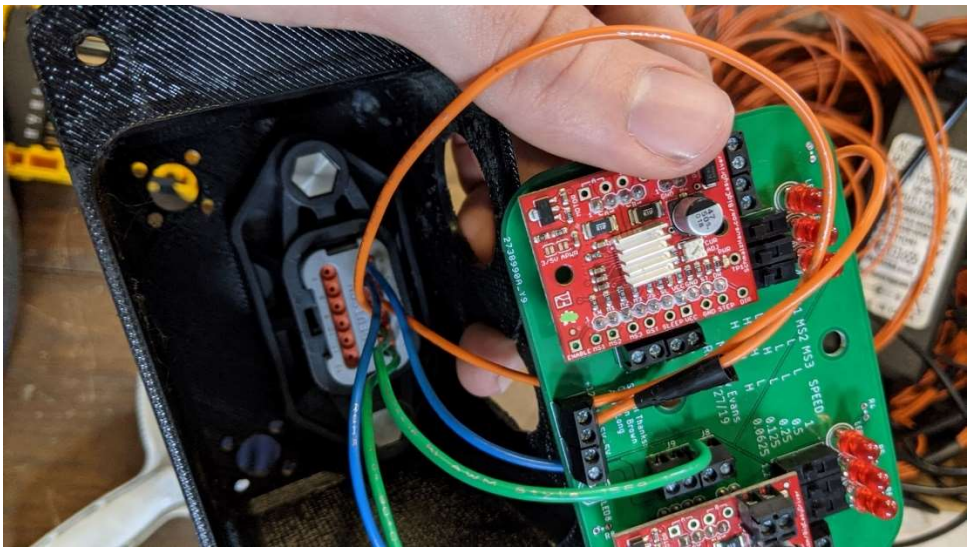

Figure 30: An early iteration of the breakout box with the Big Easy Drivers attached to the PCB

To update this design, our group made a few changes to increase the functionality of the breakout box. The first is changing the fan hole locations so the fans can be mounted to provide a crossflow of air through the breakout box. Having the crossflow would also increase the amount of cooling into the breakout box allowing for smoother operation of the motor control boards. Another consideration for this box was the addition of the intake funnel which would allow us to filter the air entering the system. Heavy particulates are generated by the torch and the filter would prevent it from accumulating in the box.

This led to the design of an additional component that serves as an inlet vent. The concept is to have the capability to put a filter (part of a mask) on the end of the vent. Having the filter removes the issue of having the heavy aluminum powder dust to enter the controls damaging them. The controls that mount in the breakout box can be seen in **Figure 31** and the assembled breakout box on the positioner can be seen in **Figure 32**.

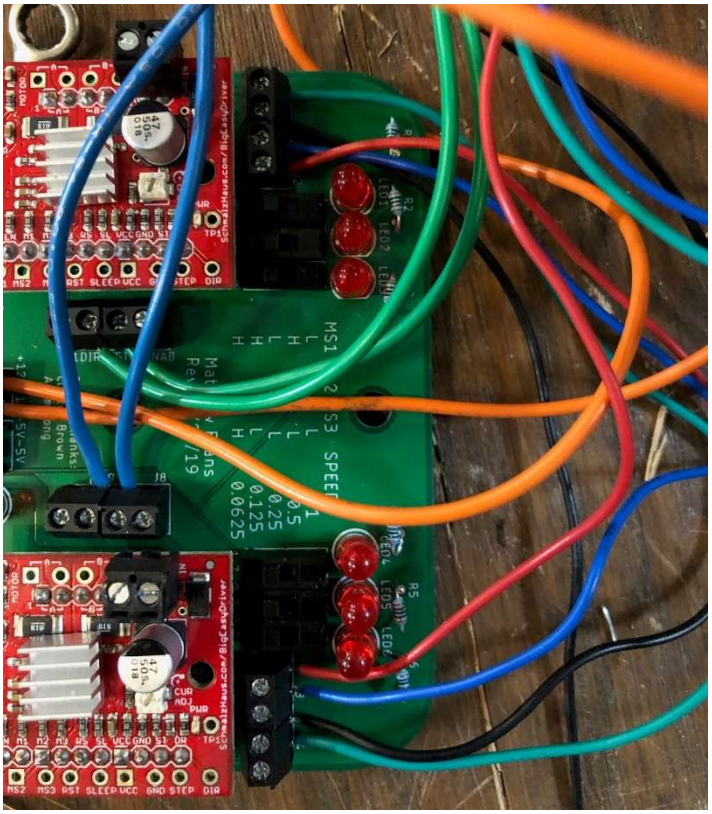

Figure 31: Assembled motor control PCB that fits in the breakout box

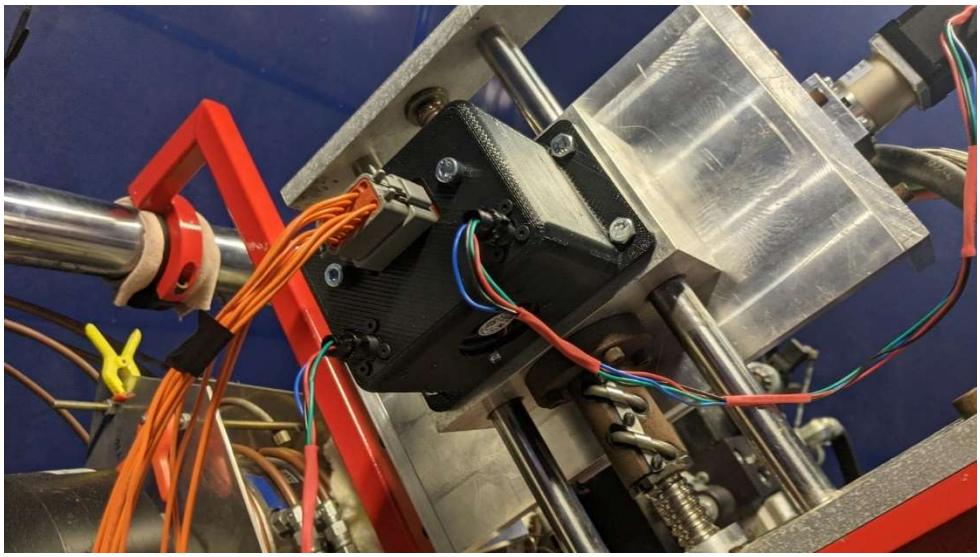

Figure 32: Assembled breakout box mounted on the positioner with the two motors connected

Final Assembly of Controller

The controller lid was attached to the base and closed. All wires are concealed except for the wires running between the controller and breakout box. The layout can be seen below in **Figure 33.**

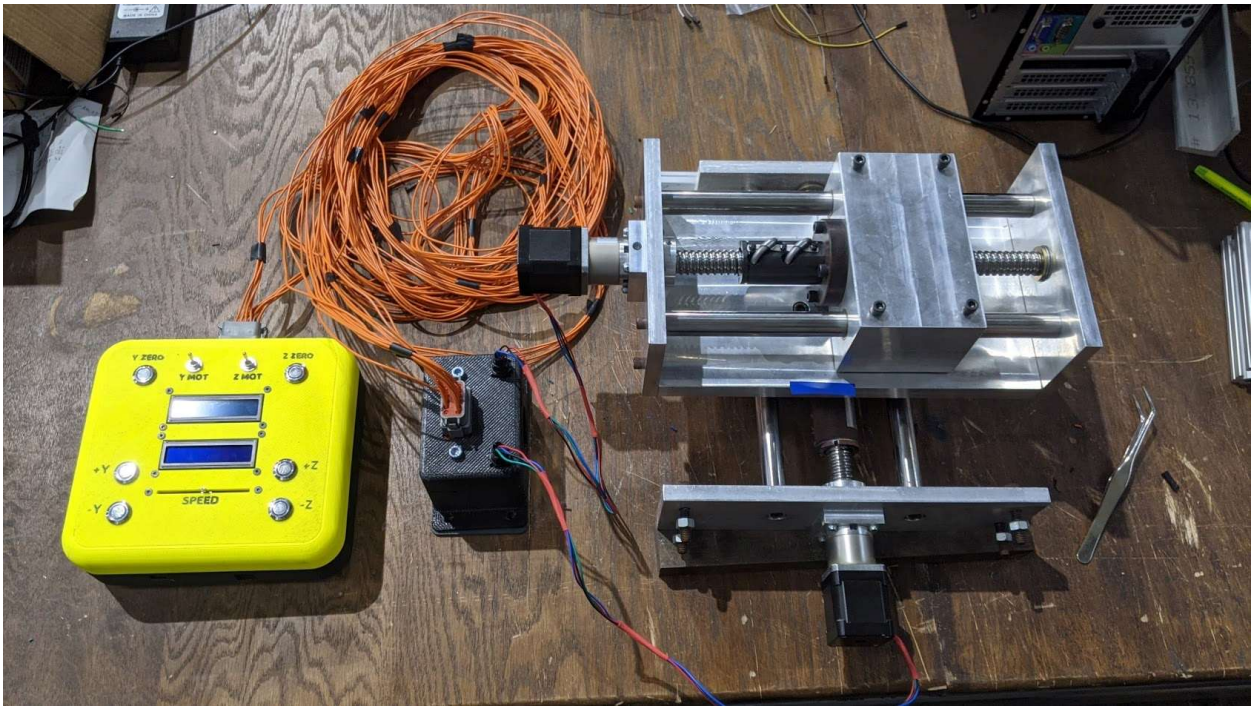

Figure 33: Final Assembly of the Controller and Breakout Box when connected to the torch positioner

Arduino Coding

The Arduino Nano and Uno were chosen for this project for two main reasons. Firstly, Arduino is one of the simplest programming languages to learn and use, and secondly, Arduinos are great for engineering projects and modeling new ideas. All types of Arduinos can be coded on Mac, Linux, and Windows, making it available to everyone. Arduino, the company that designs and manufactures these single-board microcontrollers, is an open-source company that allows users across the world to interact with one another and share different codes for various home or work-related projects. The way an Arduino operates is simple because all it does is take inputs and converts them into outputs.

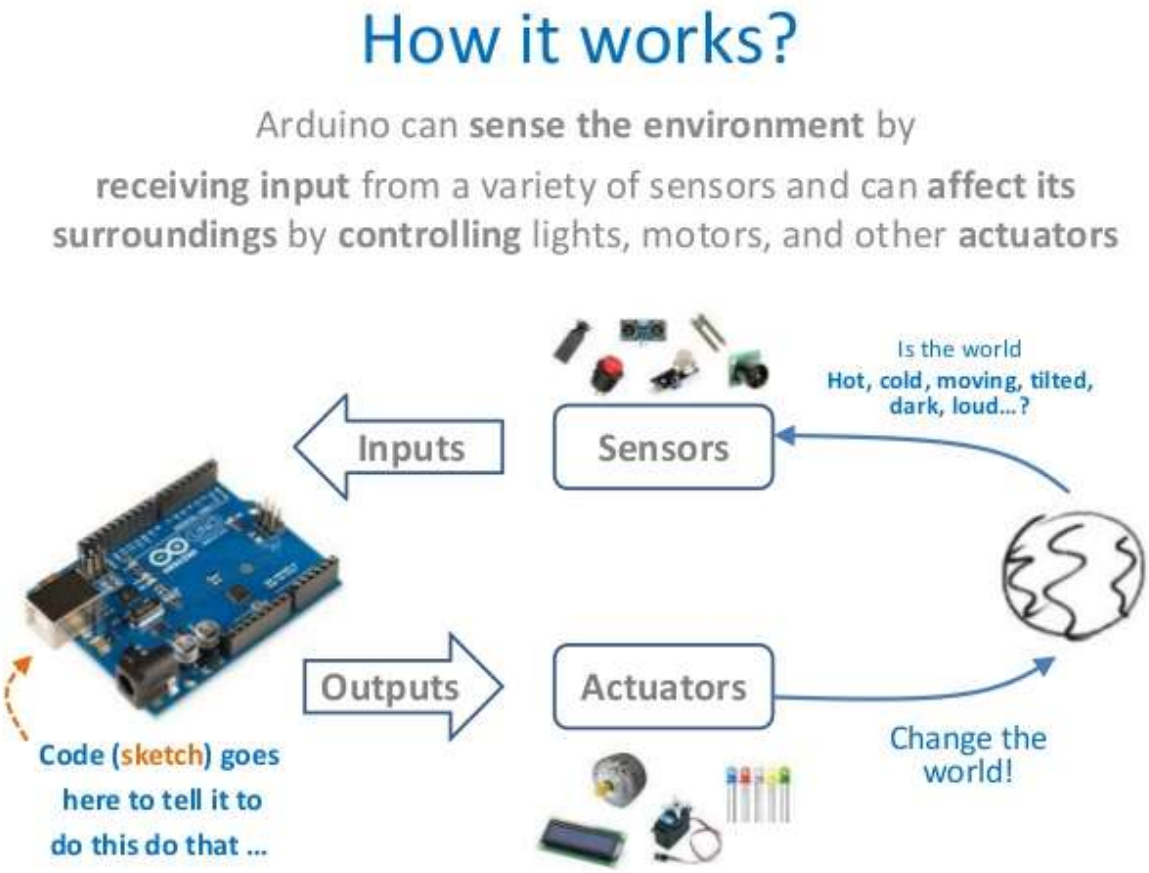

Figure 34: Arduino coding explanation

The most used Arduino, the Uno, has the following major components: 14 digital pins, 6 analog pins, a USB connector, USB interface chip, power port, microcontroller, reset switch, and TX/RX LEDs. Other components on an Uno can be seen in **Figure 35** below.

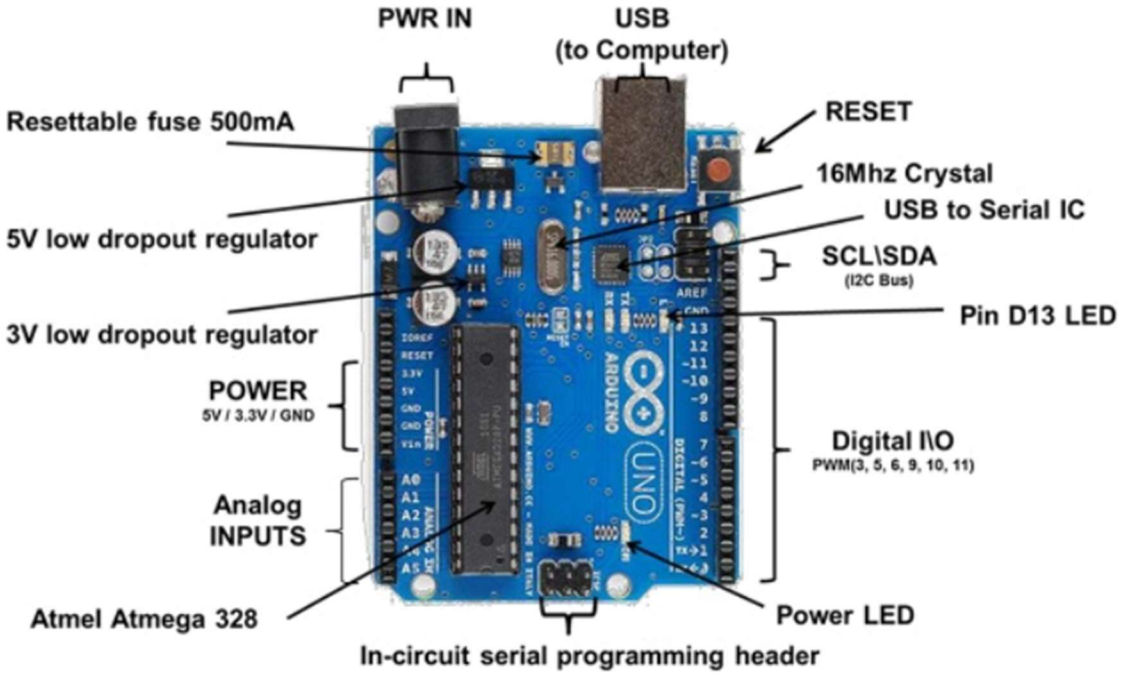

Figure 35: Arduino Uno components

Button Code

The portion of code, shown below in **Figure 36**, is formatted as a series of if statements to control the motors with pushbuttons. If the button that controls the desired motion of the stepper motors is on (HIGH state) and all others are off (LOW state), then the stepper motor will start the stepping process. This works by switching the stepping state from high to low, and this continues to repeat throughout loop until a push button is no longer pressed. There is also a qualifier for the enable switch for each if statement. If the enable switch is off, then the stepper motor will not turn even if the correct button is pushed. The enable switches were used for this project if a push button gets stuck, and the controller operator needs to stop the torch from moving.

A potentiometer is used to control the rotational speed of the motors, and this speed is converted to a translational speed that is displayed on an LCD located on the controller box. The potentiometer is connected to the Nano's analog pin A0. In order to change the switch rate between the high and low states of the motor's step, a step delay is placed between these two states. When this delay is decreased, using the *delayMicroseconds()* function, the motor's RPM will increase, and the opposite will happen when the delay is increased. The *map()* function is used to re-map a number from one range to another. For this project, we re-map the range of the potentiometer (0 to 1023, corresponding to 0-5V) to the step delay range (200 to 1000 μs).

```
void loop() {
 val = analogRead(A0);
 stepdelay = map(val,0,1023,maxdelay,mindelay);

  // Y up push button
  if(digitalRead(7) == HIGH && digitalRead(8) == LOW && digitalRead(9) == LOW && digitalRead(10) == LOW && digitalRead(13) == LOW) {
    digitalWrite(11,HIGH);   // Y direction
    digitalWrite(A3,HIGH);   // Y direction to Uno
    digitalWrite(12,HIGH);  // Y step
    digitalWrite(A1,HIGH);  // Y step to Uno
    delayMicroseconds(stepdelay);
    digitalWrite(12,LOW);   // Y step
    digitalWrite(A1,LOW);   // Y step to Uno
    delayMicroseconds(stepdelay);
  }
  // Y down push button
  if(digitalRead(8) == HIGH && digitalRead(7) == LOW && digitalRead(9) == LOW && digitalRead(10) == LOW && digitalRead(13) == LOW) {
    digitalWrite(11,LOW);  // Y direction
    digitalWrite(A3,LOW);  // Y direction to Uno
    digitalWrite(12,HIGH);  // Y step
    digitalWrite(A1,HIGH);  // Y step to Uno
    delayMicroseconds(stepdelay);
    digitalWrite(12,LOW);   // Y step
    digitalWrite(A1,LOW);   // Y step to Uno
    delayMicroseconds(stepdelay);
  }
  // Z up push button
  if(digitalRead(9) == HIGH && digitalRead(7) == LOW && digitalRead(8) == LOW && digitalRead(10) == LOW && digitalRead(2) == LOW) {
```

Figure 36: Push Button Code

Motor Setup

Below in **Figure 37** is a picture of each of the pins on the Arduino being assigned to the correct buttons. There is also a section that allows for the step and direction to be outputted to the Arduino Uno. Using the *pinMode()* function is simple. It is used to identify the pin number that is being called on, and then designating it as an input or output. The push buttons for the Y and Z directions are inputs because they tell the Nano to send signals to the stepper motor controllers. The step and direction for each motor are the outputs because the motor controllers are receiving the signals and they execute the task at hand: enabling the stepper motors to start the stepping process.

Motor_Control_Nano

```
int stepdelay = 0;
int mindelay = 450;
int maxdelay = 800;
int val;

void setup() {

  pinMode(11,OUTPUT);      // y direction
  pinMode(12,OUTPUT);      // y step
  digitalWrite(11,LOW);
  digitalWrite(12,LOW);
  pinMode(5,OUTPUT);       // z direction
  pinMode(6,OUTPUT);       // z step
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  pinMode(7,INPUT);        // y up
  pinMode(8,INPUT);        // y down
  pinMode(9,INPUT);        // z up
  pinMode(10,INPUT);       // z down

  // Arduino Uno Outputs for the LCDs
  pinMode(A1,OUTPUT);      // y step output to Uno
  pinMode(A2,OUTPUT);      // z step output to Uno
  digitalWrite(A1,LOW);
  digitalWrite(A2,LOW);
  pinMode(A3, OUTPUT);     // y direction output to Uno
  pinMode(A6, OUTPUT);     // z direction output to Uno
  digitalWrite(A3,LOW);
  digitalWrite(A6,LOW);
```

Figure 37: Assigning Pins for the Arduino Nano

Use of Interrupts for Motors

The LCDs are coded to record every time a step or direction signal is sent by breaking out of the loop. If a signal is sent, then it will be picked up using interrupt logic. The LCD code is below in **Figure 38** with the final LCD display in **Figure 39**.

```
//interruption to detect steps for Y and Z
ISR(PCINT0_vect){

  if(PINB & B00000010 ){                          //If D9 is HIGH, motor turns right so we increase steps
      if(PINB & B00001000 ){                        //If D11 is high we have one step detected
        if(step_y_state == 0){
          step_y_state = 1;
          step_count_y = step_count_y + 1;          //Increase steps by 1
        }
      }

      else if(step_y_state == 1){                   //If D11 is LOW we have another step detected
        step_y_state = 0;
        step_count_y = step_count_y + 1;            //Increase steps by 1
        }
    }//Y_up D9

    if (!(PINB & B00000010 )){                       //If D9 is LOW, motor turns left so we decrease steps
      if(PINB & B00001000 ){
        if(step_y_state == 0){
          step_y_state = 1;
          step_count_y = step_count_y - 1;           //Decrease steps by 1
        }
      }

      else if(step_y_state == 1){
        step_y_state = 0;
        step_count_y = step_count_y - 1;             //Decrease steps by 1
```

Figure 38: Using Interrupts for Arduino Uno

LCD Setup

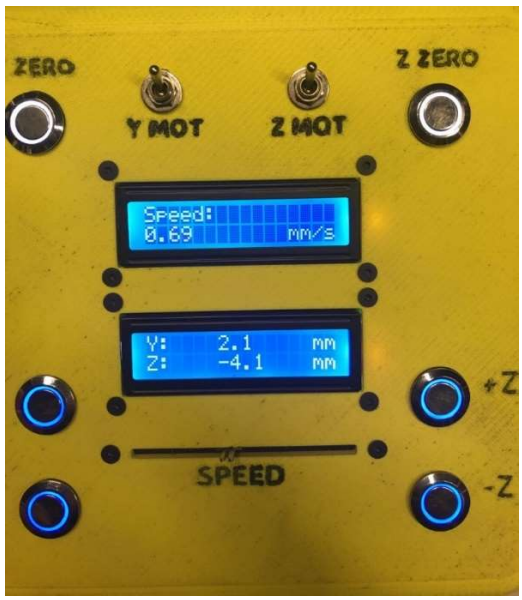

Figure 39: LCD Display on controller

For our design, we assigned the top LCD to display the speed of the stepper motors and the bottom LCD to display the distances each motors travel. As shown below in **Figure 40**, the LCDs are connected to the Arduino Uno by bridging the wires coming from both LCDs. This was done by soldering the wires together and placing heat shrink overtop of them to eliminate any wire from showing. The data from the Uno is relayed to the LCDs via the SDA line (serial data line), and the SCL line (serial clock line) which is used to synchronize all the devices with a clock signal. The SDA and SCL pins must connect to the analog pins A4 and A5 on the board for proper communication between the two components.

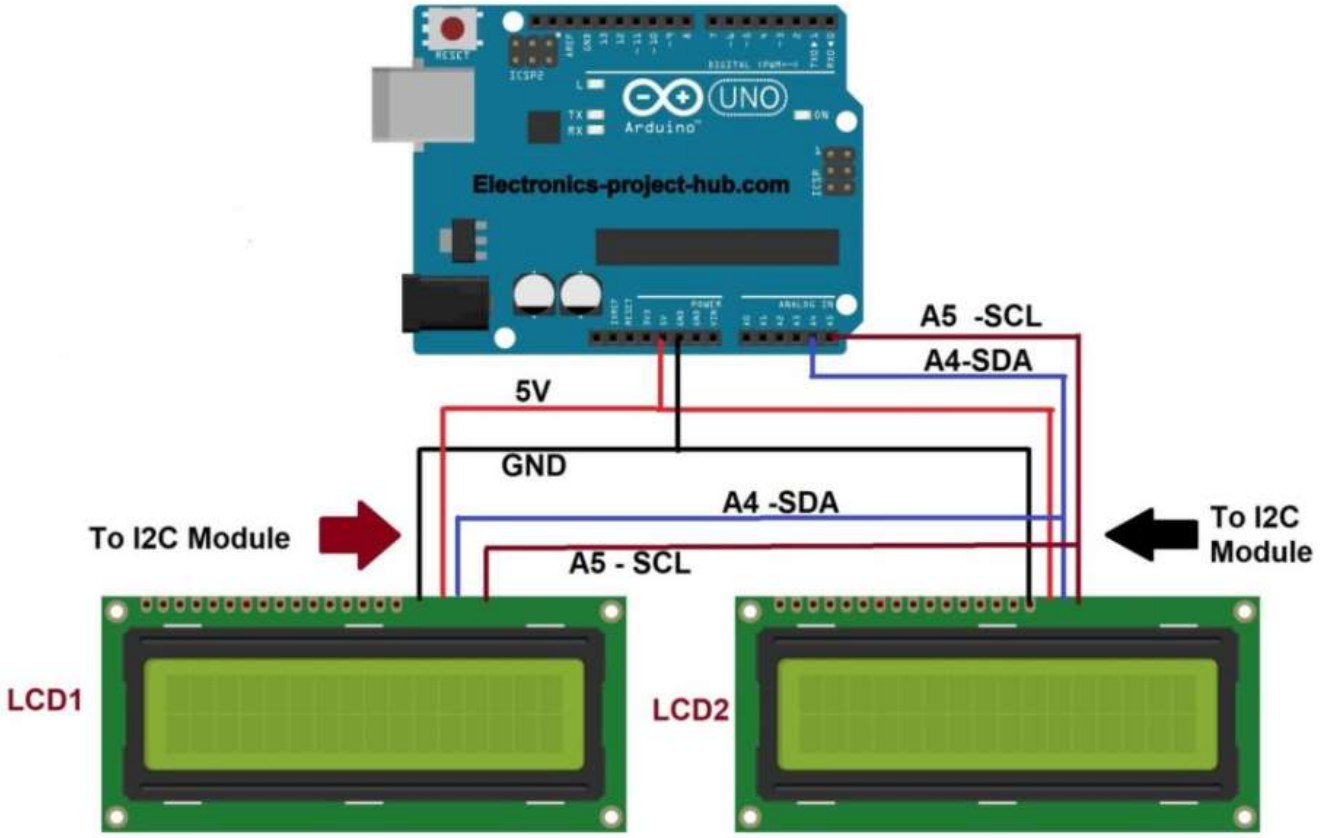

Figure 40: LCD Wiring

The addresses of both LCDs need to be identified to ensure that they are communicating properly with the Uno. To do this, a code (found on Arduino's website) can be uploaded to the Uno, and the Serial Monitor is used to tell the user the address of the LCD. If nothing is being displayed on the screens, then the addresses of the LCDs may be incorrect, and the code below, in **Figure 41**, can be used to identify them.

```
#include <Wire.h> //include Wire.h library

void setup()
{
  Wire.begin(); // Wire communication begin
  Serial.begin(9600); // The baudrate of Serial monitor is set in 9600
  while (!Serial); // Waiting for Serial Monitor
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address; //variable for error and I2C address
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for (address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println("  !");
      nDevices++;
    }
    else if (error == 4)
    {
      Serial.print("Unknown error at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000); // wait 5 seconds for the next I2C scan
}
```

Figure 41: Code to identify I2C addresses

Figure 42 shown below is the code used to display the speed the stepper motors move at and the distances that they travel. Using the LiquidCrystal I2C library in the Arduino IDE, it was simple to write the code. First, both LCDs are turned on using the *init()* and *backlight()* commands. The *setCursor()* command is then used to identify where the text is going to be

printed on the LCD. The first number in the parentheses indicates the column where the character will be placed, and the second number is for the row. For example, *setCursor(12,1)* will tell the LCD to set its cursor at the twelfth column in the second row (row and column numbers begin at zero). The *print* command is then used to tell the LCD what to print where the cursor is currently set at.

```
void setup() {

  lcd1.init();
  lcd2.init();
  lcd1.backlight();
  lcd2.backlight();

  lcd1.setCursor(0,0);              //
  lcd1.print("Speed:          ");   //
  lcd1.setCursor(0,1);              // speed
  lcd1.print(RealSpeed);            //
  lcd1.setCursor(12,1);
  lcd1.print("mm/s");               //

  lcd2.setCursor(0,0);              //
  lcd2.print("Y:              ");   //
  lcd2.setCursor(6,0);              // y distance
  lcd2.print(mm_y);                 //
  lcd2.setCursor(14,0);             //
  lcd2.print("mm");                 //

  lcd2.setCursor(0,1);              //
  lcd2.print("Z:              ");   //
  lcd2.setCursor(6,1);              // z distance
  lcd2.print(mm_z);                 //
  lcd2.setCursor(14,1);             //
  lcd2.print("mm");                 //
```

Figure 42: LCD setup code

Troubleshooting

1. Push Buttons

   If a stepper motor is not functioning it may be due to the input connections coming loose. First, check that all input connections are connected properly inside the Controller Box via the Wiring diagram. Also, check that all output connections are connected inside the Breakout Box via the schematic show in **Figure 24**. If all connections are secured and in their correct locations, then run the code below to ensure that the push buttons are working. The code in **Figure 43** should be uploaded to the Nano for the push buttons controlling the Y and Z movement. If the reset buttons are not zeroing out the distances displayed on the LCDs, then the code in **Figure 44** should be uploaded to the Uno.

Button_Troubleshoot

```
int Y_UP = 7;
int Y_DOWN = 8;
int Z_UP = 9;
int Z_DOWN = 10;

void setup() {
  Serial.begin(9600);
  pinMode(Y_UP,INPUT);
  digitalWrite(7,LOW);
  pinMode(Y_DOWN,INPUT);
  digitalWrite(8,LOW);
  pinMode(Z_UP,INPUT);
  digitalWrite(9,LOW);
  pinMode(Z_DOWN,INPUT);
  digitalWrite(10,LOW);
}

void loop() {
  int y_upState = digitalRead(Y_UP);
  int y_downState = digitalRead(Y_DOWN);
  int z_upState = digitalRead(Z_UP);
  int z_downState = digitalRead(Z_DOWN);

  Serial.print(y_upState);
  Serial.print(",");
  Serial.print(y_downState);
  Serial.print(",");
  Serial.println(z_upState);
  Serial.print(",");
  Serial.print(z_downState);
  Serial.print(",");
  delay(1);
}
```

Figure 43: Button Troubleshooting Code

Button_Troubleshoot_ZEROS

```
int Y_ZERO = 6;
int Z_ZERO = 7;

void setup() {
  Serial.begin(9600);
  pinMode(Y_ZERO,INPUT);
  digitalWrite(6,LOW);
  pinMode(Z_ZERO,INPUT);
  digitalWrite(7,LOW);
}

void loop() {
  int y_zeroState = digitalRead(Y_ZERO);
  int z_zeroState = digitalRead(Z_ZERO);

  Serial.print(y_zeroState);
  Serial.print(",");
  Serial.println(z_zeroState);
  Serial.print(",");
  delay(1);
}
```

Figure 44: Troubleshooting Code (reset buttons)

After uploading the code shown above, press Ctrl+Shift+L to open the Serial Plotter that reads the state of the push buttons. If the buttons are working properly, then the window will appear as shown below in **Figure 45**. The signals will appear clean without any presence of electronic noise. If electronic noise is present, then the resistors connecting the buttons to ground may be disconnected or burnt out, in which they will need to be replaced.

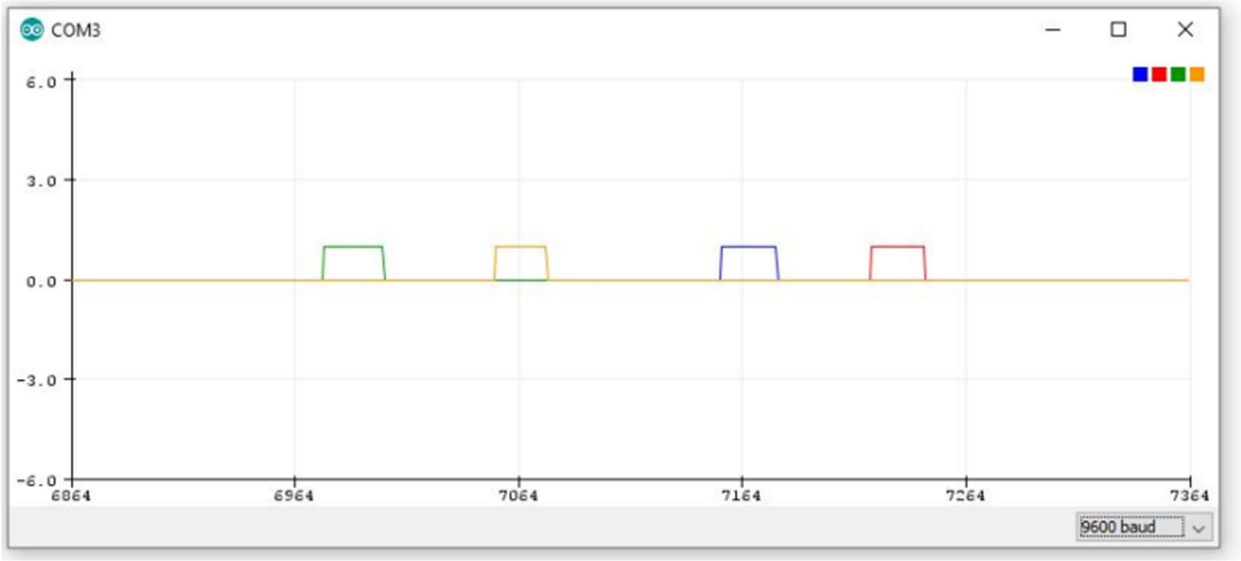

Figure 45: Serial Plotter with a plot of each button being pressed

2. Potentiometer

To check that the potentiometer is working correctly, use the code shown below in **Figure 46** for the nano. The potentiometer should in the analog pin A0.

ReadAnalogVoltage §

```
/*
  ReadAnalogVoltage

  Reads an analog input on pin 0, converts it to voltage, and prints the result to the Serial Monitor.
  Graphical representation is available using Serial Plotter (Tools > Serial Plotter menu).
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/ReadAnalogVoltage
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

Figure 46: Analog Potentiometer Troubleshooting

Once again, press Ctrl+Shift+L to open the Serial Plotter that reads the state of the potentiometer. The window shown below in **Figure 47** shows the trace of the voltage as the slide on the potentiometer is moved from min. to max. and then back to min. The trace of the voltage should appear smooth. If the potentiometer is not working properly, then the signal will appear unstable at various positions along the voltage trace.

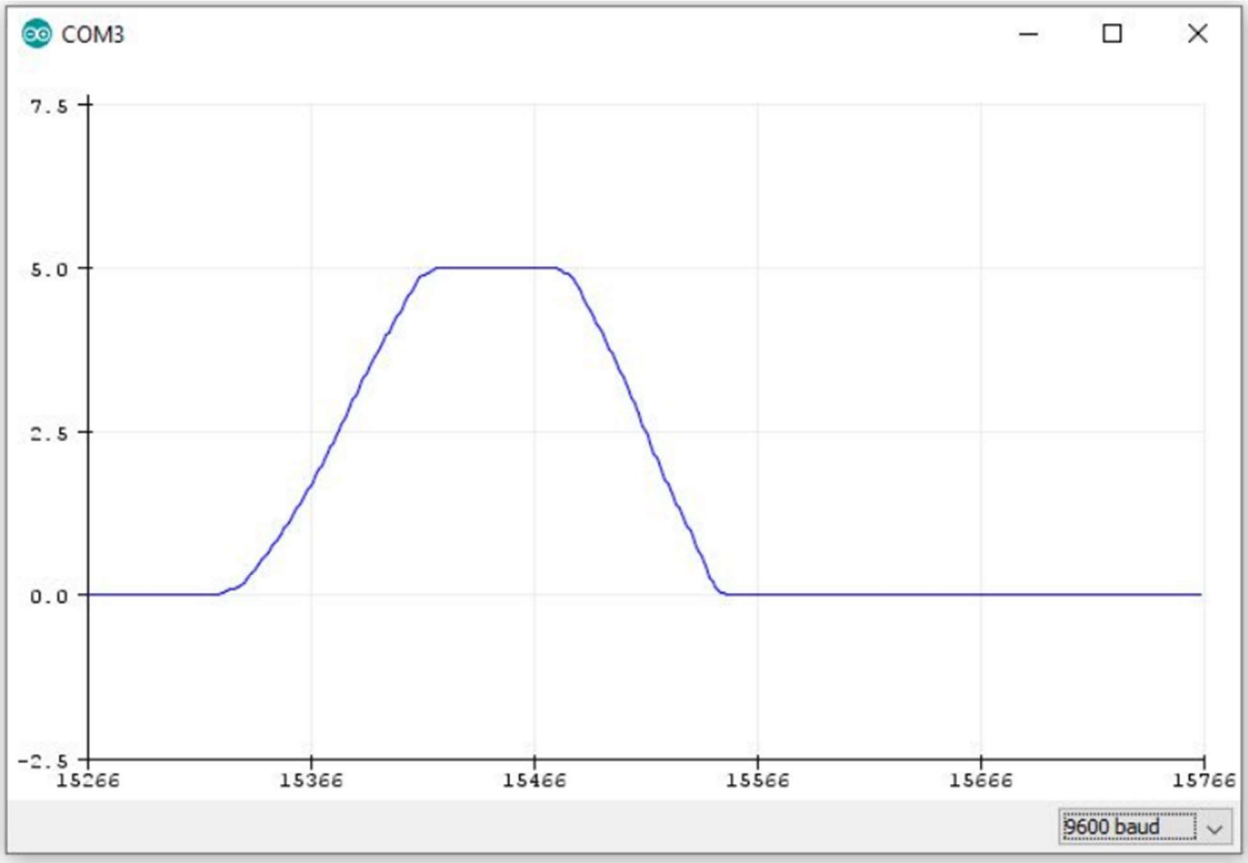

Figure 47: Serial Plotter display on potentiometer ramp up and ramp down

3. Stepper Motors

If the stepper motors are not functioning properly, and all wires are connected in their correct pin locations and the troubleshooting code shows they are functioning, then try the following:

a. Using a multimeter and the motor hookup guide shown in **Figure 48**, the correct wire pairs for each coil can be distinguished. Sometimes when stepper motors are manufactured the coil pairs are not identical to what is in the datasheet, so it is good to check to make sure. A low resistance on the multimeter indicates the wire pair is connected to the same coil and a high resistance indicates they are not the correct pair.

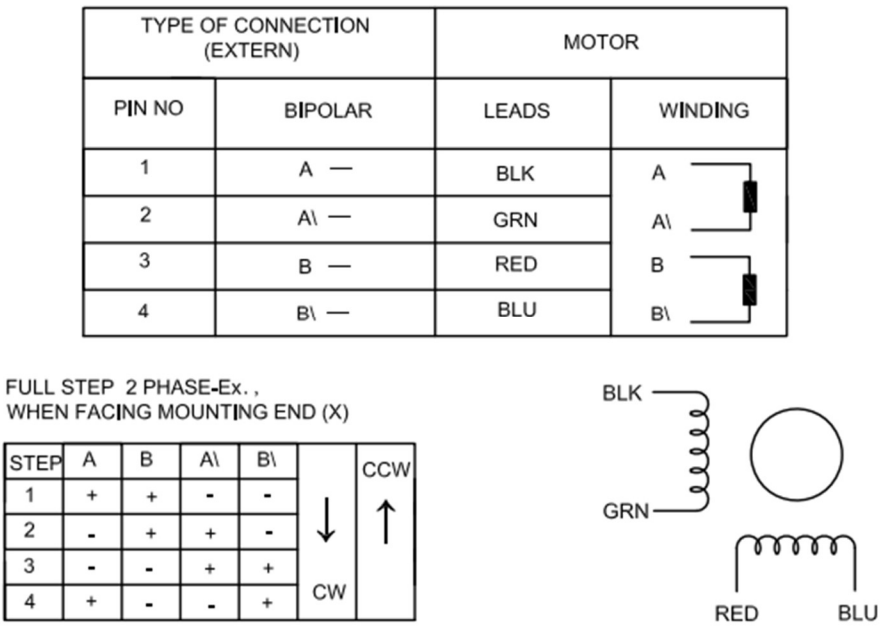

Figure 48: Bipolar stepper motor diagram

b. Check to see if the receptacles located on the Breakout Box are fully connected (**Figure 49**). Twist the ring connecting the male and female receptacles together until a click is heard, meaning they are fully connected. When the receptacles are not fully connected the motor may not move in both directions and/or will not move at all.

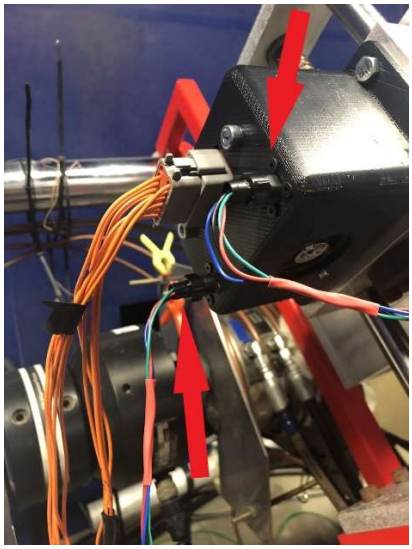

Figure 49: Location of Receptacles on Breakout Box

c. Check the pins that are attached to the receptacles. These pins connect the motor's wires to the housing of the receptacle. Compare the depths of the pins to each other, if one is shorter than the others it is noticeable. This means a pin is not fully inserted into the receptacle housing, so gently push it in until a click is heard. Use caution when doing this to prevent the connection between the wires and the pins from severing.

Figure 50: Close-up of Receptacle Pins

d. Make sure that the step and direction wires are fully connected to the terminal blocks (and in the correct locations) located in the Controller Box and Breakout Box. If these become loose then the motor will not perform correctly. If everything is connected

correctly, then check to make sure the Nano is sending the signals to the motor controllers by uploading the code in **Figure 51**.

Step_Direction_Troubleshoot

```
int Y_dir = 11;
int Y_step = 12;
int Z_dir = 5;
int Z_step = 6;

void setup() {
  Serial.begin(9600);
  pinMode(Y_dir,INPUT);
  digitalWrite(9,LOW);
  pinMode(Y_step,INPUT);
  digitalWrite(11,LOW);
  pinMode(Z_dir,INPUT);
  digitalWrite(10,LOW);
  pinMode(Z_step,INPUT);
  digitalWrite(12,LOW);
}

void loop() {
  int y_dirState = digitalRead(Y_dir);
  int y_stepState = digitalRead(Y_step);
  int z_dirState = digitalRead(Z_dir);
  int z_stepState = digitalRead(Z_step);

  Serial.print(y_dirState);
  Serial.print(",");
  Serial.print(y_stepState);
  Serial.print(",");
  Serial.println(z_dirState);
  Serial.print(",");
  Serial.print(z_stepState);
  Serial.print(",");
  delay(1);
}
```

Figure 51: Troubleshooting Code for Step and Direction

Again, press Ctrl+Shift+L to open the Serial Plotter that reads the state of the step and direction signals. Press the Y and Z buttons on the Controller Box (not at the same time). If the Nano is sending the signals to the motor controllers, then a signal will appear in the Serial Plotter window when a button is pressed.

Final Assembly

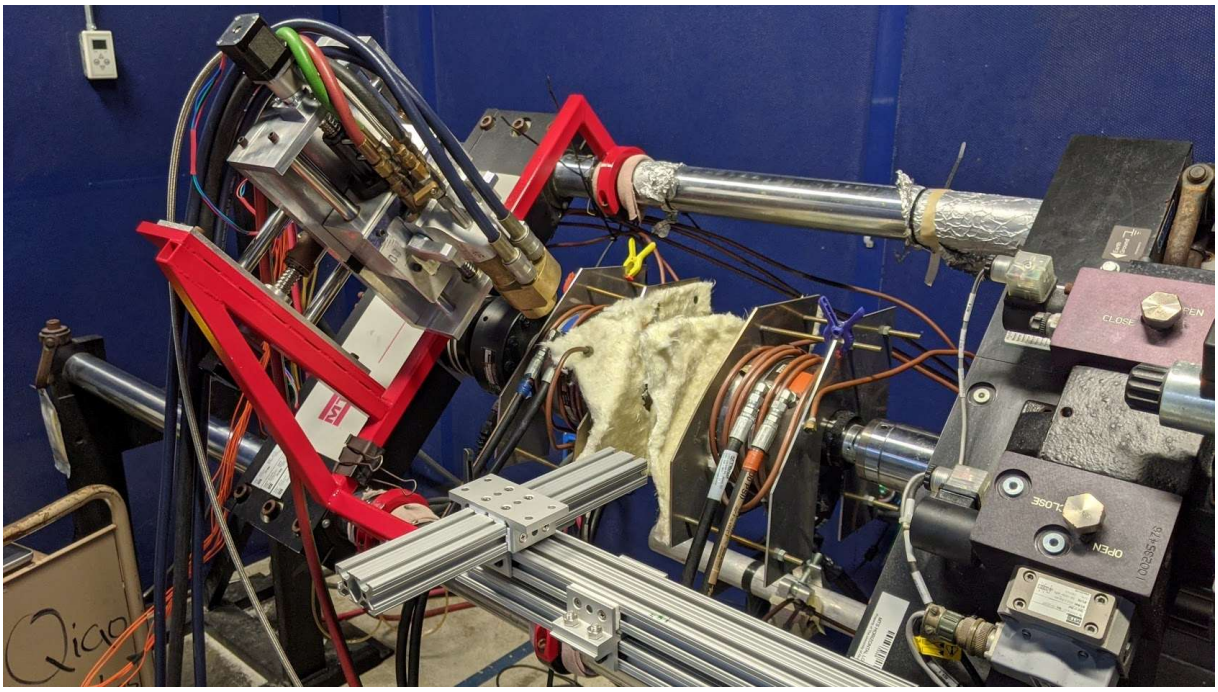

Figure 52: Final Assembly of the positioner and camera mount

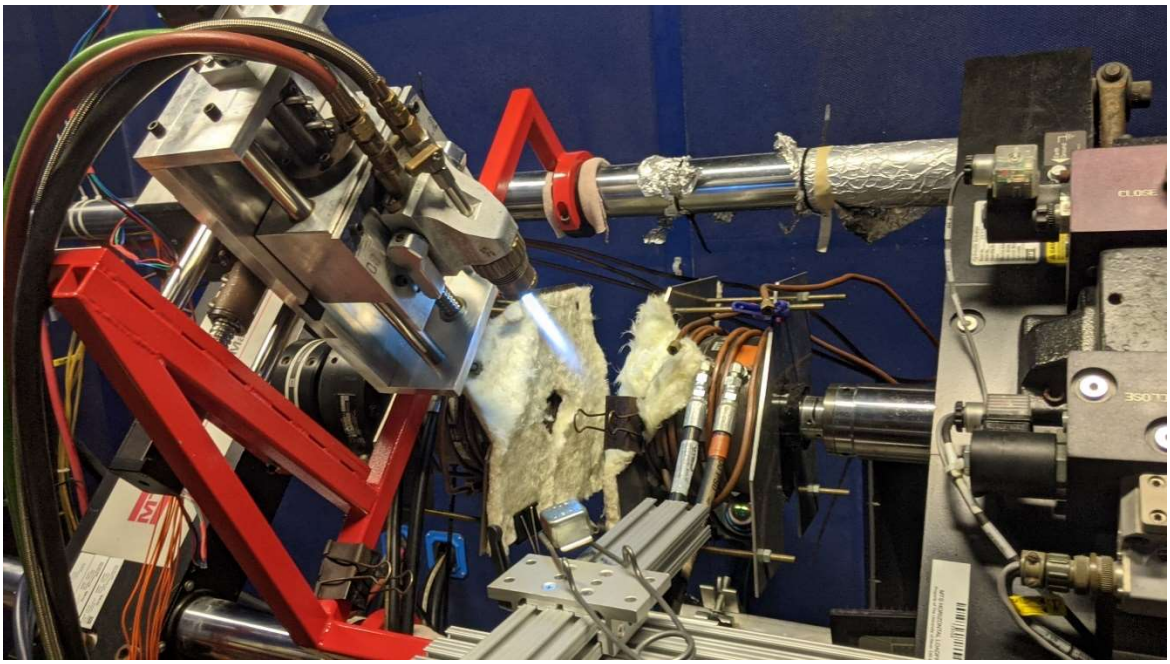

Figure 53: Operation of the torch with the working positioner

Lessons Learned

Hardware:

A major improvement that could have been made to this project is a redesigned controller PCB. The current PCB was designed by the previous group under the assumption that the project could work with one Arduino. The internals of the controller had to be redesigned late to include a second Arduino using its header pins instead of the neater way of using PCB pins. A new PCB would clean up the internal wiring of the controller and would allow for a smaller case.

In addition to the new PCB, we would have liked to change the microcontroller used in the controller box. If we could redo this project, we would use a Raspberry Pi Pico or a Teensy 4.1 which are competitors to the Arduino Nano. The Pico has six times the processing power, five times the memory and it is half the cost. This board could control both the motors and the LCDs without an issue. A Pico would simplify the controller by not needing to communicate motor signals to another Arduino.

One issue with the controller PCB was it initially had all ground wires from each component fit into one junction. This caused a insecure connection and grounding issues throughout the controller. A quick redesign and a freshly soldered board fixed that issues, but we learned how connection issues could become large issues on our project. Connection issues were a problem throughout and showed up in the circular connectors on the breakout box. These connect the motor to the motor control board using a quick connect. The four wires from the motor had to be crimped into ferrules and inserted into these connectors. The problem was that

these ferrules would be pulled out easily causing connection issues. The solution was to use slightly bigger wires with more mass to crimp onto.

Arduinos have a certain current threshold that it can support using the 5V pins. With six pushbuttons, two toggle switches, potentiometer and two LCDs drew too much of a load and prevented the Arduino from working correctly. We had to split this current load into two Arduinos to correctly power every component correctly and if we knew that before, we would have designed a new PCB board to accommodate that fact.

An important lesson when working with Arduinos is that each pin has different roles. We were stuck on a problem for over a week where the y-location was showing up correctly on the LCD but not the z-location. The reason was that the z-direction wires were plugged into the Arduino Nano's A6 and A7 pins. These pins are input only which prevented the signal from being sent like we hoped. Knowledge on the Arduino pins would have benefitted us early on due to problems like this.

A problem that came up during assembly was the jumper wires becoming disconnected as we moved the lid. After the lid of the controller was closed its hard to access the wires without opening the lid again. We did not want the wires to become disconnected during assembly, so we applied heat shrink on each connection to prevent movement. This solved our issue with jumper wires.

Soldering skills were a big learning curve for the team as no one had these skills prior to this project. One problem with soldering initially was that we had the temperature of the gun set too low which did not melt the solder cleanly. It was burning the PCB board because the hot tip had to be on the board longer to melt the solder and create a good connection. The lesson learned was to use a hotter temperature, around 700F. In the same vein of soldering, we learned to not solder jumper wires directly to the PCB board. The final product has directly soldered wires but the major problem with them is that the wires can easily come out of the connector's crimp. It is hard to get the wires back in after it breaks free from the crimp.

During the building phase of the controller, the electronic parts did not always do what we wanted them to do. A multimeter was used to track down voltage, resistance, and current from different points of interest in, and was invaluable for diagnosing problems.

The last hardware lesson learned is that resistors are required for pushbuttons and that without them, the button will fluctuate between High and Low. We thought that resistors were optional at the beginning, so we did not take the proper steps to include them in our designs.

Coding:

Learning how to use Arduino IDE code has helped further our knowledge about how to initiate inputs and outputs in an electronics system. At the beginning of this project, a few of us had little experience with Arduinos but had a basic understanding of what are considered inputs and outputs in a mechanical/electrical system.

We also learned how fast some software can process the information in the code. For example, when we first wrote the code used for the motors, we did not use any delays between the high and low states on the step. This resulted in the motor not rotating consistently because the software was going through the code at a pace that was too fast to process the information. Adding in delays between some of the coding allowed the software to have more time to execute the commands properly.

Another lesson learned was the use of interrupts and Boolean logic. This allowed for the Uno to keep track of the number of steps the motors were taking. We needed to do this to calculate the distance the motors are traveling when in use. Keeping track of the stepper motors position is beneficial to the operators of the Burner Rig. With an accurate location of the motors the operators know where to place the torch for a desired temperature of the test specimen being heated.

The final lesson learned when it comes to coding is how to print text onto an LCD. This is pertinent to the overall design of the Controller Box. It is useful for the operators to be able to see the speed that the motors are moving at. While observing the temperature readings from the FLIR camera the operators can determine how fast they want to torch to move in order to reach a desired temperature within a specified amount of time.

Conclusions

Accomplishments

In this project, the team built both the mount to hold the FLIR thermal camera in a compact and stable package, and the torch positioner controller to create a safe work environment. These accomplishments will allow for the burner rig to be used more efficiently and more accurately leading to better research. Dr. Morscher's team uses the burner rig extensively for their work on ceramic matrix composites so this will be a large help for them.

As a team we learned several concepts during the creation of our project. We learned how to work with extruded aluminum to add an adjustable layer to the camera mount in case the researchers wanted to change the camera's angle. The controller pushed us to learn how to use and code an Arduino, how to solder and how to read basic electrical schematics. This project pushed us in directions we never explored before and has broadened our knowledge of engineering.

Future Work

This project will continue being used by the Burner rig team during their upcoming research. It will aid them by increasing the precision of adjustment to the oxygen flame torch. The controller will be built upon with Chris Ferguson's Master thesis over the next two years. His work will include creating a controls system using LabView and the existing positioner. The

scope of his thesis is similar to the proposed work in this project but will include material testing as well. Because of the automated system that will be created in LabView, the controller will only be used for short testing that does not require constant adjustment of the temperature.

The automated system will utilize the existing torch positioner, camera mount and breakout box to control the temperature of the sample. The main goal is to automate the movement of the torch using a temperature input from the FLIR camera. Once the temperature is set, the positioner can adjust itself for the correct setpoint. A relationship between temperature and torch distance will have to be researched to find the correct method of positioning the torch. The LabView control system will expand the capabilities of the burner rig by allowing for thermal cycle testing. Aircraft fly in a set cycle of startup, takeoff, cruise and then landing. The temperature that the ceramic within the jet turbine undergoes changes during each one of these cycles. The LabView controller will allow for the burner rig to test ceramic samples under the conditions of a typical flight or multiple flights. The work completed in this senior design project is useful to complete this future endeavor and acts as a steppingstone along the way.

Engineering Standards

To follow correct engineering practice, the group followed multiple Engineering Standards to make our work more official. The first is following Engineering Drawing Practices(Y14:100-2017) which was used in the final camera mount drawing. The benefit to following a standard drawing practice is to ensure uniformity. Although the members of the group were the only ones reading the final drawing, if the drawing were to be sent to an outside machine shop or manufacturing firm, they would be able to easily read the drawing and design the product.

Another standard followed is ISO29.060 which standardizes the size and rating of electrical wires. Having this standardization assisted the group with purchasing additional materials so we knew the size, voltage, amperage, etc. the wire could handle. Although this is not a standard which the group had to follow, learning about this standard has allowed us to know what to search for when working with electrical components.

Acknowledgments

Without the help of our advisor, the lab group, faculty, students and others, this project would not have been as successful. The team wants to thank our advisor Dr. Morscher, the lab group consisting of Ragavendra Prasad Panakarajupally, Leland Hoffman, Farhan Mirza, Andrew Wenzel, and Joseph El Rassi, the previous group that worked on the torch positioner consisting of Matthew Evans, Donald Morrison, and Joshua Verdier, Engineering Technician

Brett Bell, Jeff Slone who painted the positioner arm, and Zack Owen who assisted the group with questions about Arduino coding and hardware.